File created: 2-Nov-93 13:53:08 **{PELE:MV:ENVOS}<LISPCORE>SOURCES>COURIER.;5**

changes to: (FNS \MAKE.EXPEDITED.STREAM \COURIER.BROADCAST.ON.NET)

previous date: 28-Apr-92 17:35:17 {PELE:MV:ENVOS}<LISPCORE>SOURCES>COURIER.;4

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;
;; Copyright (c) 1983, 1984, 1985, 1986, 1987, 1988, 1989, 1990, 1991, 1992, 1993 by Venue & Xerox Corporation. All rights reserved.

```
(RPAQQ COURIERCOMS
       [(COMS                                                    ; COURIER Protocol
              (DECLARE%: EVAL@COMPILE DONTCOPY (COMS * COURIERDECLS))
              (COMS                                              ; COURIERPROGRAMS type
                    (INITVARS (\COURIERPROGRAM (HARRAY 20)))
                    (GLOBALVARS \COURIERPROGRAM)
                    (FILEPKGCOMS COURIERPROGRAMS)
                    (FNS COURIER.VERSION# COURIERPROGRAM \COURIER.CHECKDEF \COURIER.CHECK.PROCEDURES
                         \COURIER.CHECK.ERRORS \COURIER.DELDEF \COURIER.GETDEF \COURIER.PUTDEF
                         \DUMP.COURIERPROGRAMS)
                    (FNS \GET.COURIER.TYPE \GET.COURIER.DEFINITION))
              (COMS                                              ; COURIER record access
                    (MACROS COURIER.FETCH COURIER.CREATE)
                    (PROP INFO COURIER.FETCH COURIER.CREATE)
                    (FNS \COURIER.RECORDTRAN))
              (COMS                                              ; COURIER calls and returns
                    (FUNCTIONS STREAMTYPECASE)
                    (FNS COURIER.OPEN \COURIER.WHENCLOSED COURIER.CALL COURIER.EXECUTE.CALL \COURIER.RESULTS
                         COURIER.SIGNAL.ERROR \COURIER.HANDLE.BULKDATA \COURIER.HANDLE.ERROR \BULK.DATA.STREAM
                         \COURIER.ATTENTIONFN \COURIER.OUTPUT.ABORTED \BULK.DATA.CLOSE \ABORT.BULK.DATA))
              (FNS COURIER.EXPEDITED.CALL COURIER.EXECUTE.EXPEDITED.CALL \BUILD.EXPEDITED.XIP
                   \SEND.EXPEDITED.XIP \COURIER.EXPEDITED.ARGS \MAKE.EXPEDITED.STREAM \COURIER.EOF
                   \COURIER.EXPEDITED.OVERFLOW)
              (FNS COURIER.BROADCAST.CALL \COURIER.BROADCAST.ON.NET)
              (FNS COURIER.READ \COURIER.UNKNOWN.TYPE COURIER.READ.SEQUENCE COURIER.READ.STRING COURIER.WRITE
                   COURIER.WRITE.SEQUENCE COURIER.WRITE.STRING COURIER.WRITE.FAT.STRING COURIER.SKIP
                   COURIER.SKIP.SEQUENCE \COURIER.TYPE.ERROR DECODE-NS-STRING)
              (FNS COURIER.READ.BULKDATA BULKDATA.GENERATOR BULKDATA.GENERATE.NEXT COURIER.WRITE.BULKDATA
                   COURIER.ABORT.BULKDATA)
              (COMS                                              ; Reading/writing sequence unspecified in an interesting way
                    (FNS COURIER.READ.REP COURIER.WRITE.REP COURIER.WRITE.SEQUENCE.UNSPECIFIED \CWSU.DEFAULT
                         COURIER.REP.LENGTH \MAKE.COURIER.REP.STREAM \COURIER.REP.BIN \COURIER.REP.BOUT)
                    (INITVARS \COURIER.REP.DEVICE))
              (COMS (FNS COURIER.READ.NSADDRESS COURIER.WRITE.NSADDRESS)
                    (PROP COURIERDEF NSADDRESS)))
        (COMS                                                    ; Debugging
              (INITVARS (COURIERTRACEFILE)
                     (COURIERTRACEFLG)
                     (COURIERPRINTLEVEL '(2 . 4))
                     (NSWIZARDFLG))
              (GLOBALVARS COURIERTRACEFLG COURIERTRACEFILE COURIERPRINTLEVEL NSWIZARDFLG)
              (FNS COURIERTRACE \COURIER.TRACE))
        (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA \DUMP.COURIERPROGRAMS
                                                                                 COURIERPROGRAM)
                                                                           (NLAML)
                                                                           (LAMA COURIER.EXPEDITED.CALL
                                                                                 COURIER.CALL])
```

;; COURIER Protocol

```
(DECLARE%: EVAL@COMPILE DONTCOPY

(RPAQQ COURIERDECLS
       ((DECLARE%: EVAL@COMPILE (FILES (SOURCE)
                                       LLNSDECLS SPPDECLS))
        (CONSTANTS (COURIER.VERSION# 3))
        (CONSTANTS (\COURIERMSG.CALL 0)
               (\COURIERMSG.REJECT 1)
               (\COURIERMSG.RETURN 2)
               (\COURIERMSG.ABORT 3))
        (CONSTANTS (\NS.WKS.Courier 5))
        (MACROS \GET.COURIERPROGRAM \COURIER.QUALIFIED.NAMEP NULLORLISTP)
        (RECORDS COURIERPGM COURIERFN COURIERERR \BULK.DATA.CONTINUATION COURIERREPSTREAM BULKDATAGENERATOR)
        (GLOBALVARS LCASEFLG \COURIER.REP.DEVICE \BASEBYTESDEVICE)
        (COMS (CONSTANTS (\EXPEDITED.LENGTH (IPLUS \XIPOVLEN 6 4))
                      \EXTYPE.EXPEDITED.COURIER)
              (RECORDS EXPEDITEDXIP))))

(DECLARE%: EVAL@COMPILE

(FILESLOAD (SOURCE)
```

```
       LLNSDECLS SPPDECLS)
)

(DECLARE%: EVAL@COMPILE

(RPAQQ COURIER.VERSION# 3)

(CONSTANTS (COURIER.VERSION# 3))
)

(DECLARE%: EVAL@COMPILE

(RPAQQ \COURIERMSG.CALL 0)

(RPAQQ \COURIERMSG.REJECT 1)

(RPAQQ \COURIERMSG.RETURN 2)

(RPAQQ \COURIERMSG.ABORT 3)

(CONSTANTS (\COURIERMSG.CALL 0)
       (\COURIERMSG.REJECT 1)
       (\COURIERMSG.RETURN 2)
       (\COURIERMSG.ABORT 3))
)

(DECLARE%: EVAL@COMPILE

(RPAQQ \NS.WKS.Courier 5)

(CONSTANTS (\NS.WKS.Courier 5))
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS \GET.COURIERPROGRAM MACRO ((PROGRAM)
                                     (GETHASH PROGRAM \COURIERPROGRAM)))

(PUTPROPS \COURIER.QUALIFIED.NAMEP MACRO [OPENLAMBDA (X)
                                           (AND (LISTP X)
                                                (LITATOM (CDR X))
                                                (LITATOM (CAR X])

(PUTPROPS NULLORLISTP MACRO (OPENLAMBDA (X)
                              (OR (NULL X)
                                  (LISTP X))))
)

(DECLARE%: EVAL@COMPILE

(RECORD COURIERPGM (VERSIONPAIR . COURIERDEFS)
       (RECORD VERSIONPAIR (PROGRAM# VERSION#))
       (PROPRECORD COURIERDEFS (TYPES PROCEDURES ERRORS INHERITS)))

(RECORD COURIERFN (FN# ARGS RETURNSNOISE RESULTS REPORTSNOISE ERRORS))

(RECORD COURIERERR (ERR# ARGS))

(RECORD \BULK.DATA.CONTINUATION (PROGRAM PROCEDURE PGMDEF PROCDEF NOERRORFLG INTERNALFLG))

[ACCESSFNS COURIERREPSTREAM ((CRWORDLIST (fetch F1 of DATUM)
                                        (replace F1 of DATUM with NEWVALUE))
                            (CRNEXTBYTE (fetch F2 of DATUM)
                                        (replace F2 of DATUM with NEWVALUE))
                            (CRLASTWORD (fetch F3 of DATUM)
                                        (replace F3 of DATUM with NEWVALUE]

(RECORD BULKDATAGENERATOR (BGITEMSLEFT BGSTREAM (BGPROGRAM . BGTYPE) . BGLASTSEGMENT?))
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS LCASEFLG \COURIER.REP.DEVICE \BASEBYTESDEVICE)
)

(DECLARE%: EVAL@COMPILE

(RPAQ \EXPEDITED.LENGTH (IPLUS \XIPOVLEN 6 4))

(RPAQQ \EXTYPE.EXPEDITED.COURIER 2)

(CONSTANTS (\EXPEDITED.LENGTH (IPLUS \XIPOVLEN 6 4))
       \EXTYPE.EXPEDITED.COURIER)
)

(DECLARE%: EVAL@COMPILE
```

```
[ACCESSFNS EXPEDITEDXIP ((EXPEDITEDBASE (fetch (PACKETEXCHANGEXIP PACKETEXCHANGEBODY) of DATUM)))
       (BLOCKRECORD EXPEDITEDBASE ((LOWVERSION WORD)
                                   (HIGHVERSION WORD)
                                   (MSGTYPE WORD)
                                   (TRANSACTIONID WORD)
                                   (PROGRAM# FIXP)
                                   (VERSION# WORD)
                                   (PROCEDURE# WORD)
                                   (ARG0 WORD)))
       (ACCESSFNS EXPEDITEDXIP ((EXPEDITEDMSGBODY (LOCF (fetch (EXPEDITEDXIP MSGTYPE) of DATUM)))
                                (EXPEDITEDARGBASE (LOCF (fetch (EXPEDITEDXIP ARG0) of DATUM]
)
)
```

;; COURIERPROGRAMS type

```
(RPAQ? \COURIERPROGRAM (HARRAY 20))

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \COURIERPROGRAM)
)

[PUTDEF 'COURIERPROGRAMS 'FILEPKGCOMS '((COM MACRO (X (E (\DUMP.COURIERPROGRAMS . X)))
                                             CONTENTS NILL)
                                        (TYPE DESCRIPTION "Courier programs" GETDEF \COURIER.GETDEF DELDEF
                                              \COURIER.DELDEF PUTDEF \COURIER.PUTDEF]

(DEFINEQ
```

(COURIER.VERSION#

```
  [LAMBDA NIL                                                    (* bvm%: " 2-May-84 12:27")
```

;;; Returns number of the version of Courier we are running

```
    COURIER.VERSION#])
```

(COURIERPROGRAM

```
  [NLAMBDA X                                                     (* bvm%: "10-Jun-84 23:02")
```

;; Define a Courier program and its associated types, constants, procedures, and errors. Syntax is (COURIERPROGRAM programName
;; (programNumber versionNumber) TYPES (typeDeclarations ...) PROCEDURES (procedureDeclarations ...) ERRORS (errorDeclarations ...)) The
;; TYPES, PROCEDURES, and ERRORS may appear in any order after the program number/version number pair.

```
    (PUTDEF (CAR X)
           'COURIERPROGRAMS
           (CDR X])
```

(\COURIER.CHECKDEF

```
  [LAMBDA (NAME DEF)                                             (* bvm%: "16-Jul-84 15:36")
    (COND
       ([OR (NLISTP (fetch (COURIERPGM VERSIONPAIR) of DEF))
            (NOT (FIXP (fetch (COURIERPGM PROGRAM#) of DEF)))
            (NOT (FIXP (fetch (COURIERPGM VERSION#) of DEF]
        (ERROR "Bad version specification in Courier def" NAME))
       (T (for TAIL on (fetch COURIERDEFS of DEF) by (CDDR TAIL) do (SELECTQ (CAR TAIL)
                                                                            ((TYPES INHERITS))
                                                                            (PROCEDURES (\COURIER.CHECK.PROCEDURES
                                                                                          (CADR TAIL)))
                                                                            (ERRORS (\COURIER.CHECK.ERRORS
                                                                                      (CADR TAIL)))
                                                                            (ERROR "Courier definition not
                                                                                   understood" (CAR TAIL])
```

(\COURIER.CHECK.PROCEDURES

```
  [LAMBDA (DEFS)                                                 (* bvm%: "12-Oct-84 11:24")
    (for FNDEF in DEFS bind INFO
       unless
       [COND
          ((NLISTP FNDEF)
           NIL)
          ((EQ (CAR FNDEF)
               COMMENTFLG)                                       ; Comments ok
           T)
          (T (SETQ INFO (CDR FNDEF))
             (COND
                ((AND (FIXP (fetch (COURIERFN FN#) of INFO))
                      (NULLORLISTP (fetch (COURIERFN ARGS) of INFO))
                      (LITATOM (fetch (COURIERFN RETURNSNOISE) of INFO))
                      (NULLORLISTP (fetch (COURIERFN RESULTS) of INFO))
                      (LITATOM (fetch (COURIERFN REPORTSNOISE) of INFO))
                      (NULLORLISTP (fetch (COURIERFN ERRORS) of INFO)))
                                                                 ; nice new format
                 T)
                (T (PROG (ARGS RESULTS ERRORS N)
```

```
                        (RETURN (COND
                                   ([while INFO do (COND
                                                      [(NULL (CDR INFO))
                                                       (RETURN (FIXP (SETQ N (CAR INFO]
                                                      (T (SELECTQ (CAR INFO)
                                                              (ARGS (OR (NULLORLISTP (SETQ ARGS (CADR INFO)))
                                                                        (RETURN)))
                                                              (RESULTS (OR (NULLORLISTP (SETQ RESULTS
                                                                                              (CADR INFO)))
                                                                           (RETURN)))
                                                              (ERRORS (OR (NULLORLISTP (SETQ ERRORS (CADR INFO))
                                                                                       )
                                                                          (RETURN)))
                                                              (RETURN))
                                                         (SETQ INFO (CDDR INFO]
                                    (/RPLACD FNDEF (create COURIERFN
                                                           FN# _ N
                                                           ARGS _ ARGS
                                                           RETURNSNOISE _ 'RETURNS
                                                           RESULTS _ RESULTS
                                                           REPORTSNOISE _ 'REPORTS
                                                           ERRORS _ ERRORS))
                                    T]
      do (ERROR "Bad Courier Procedure definition" FNDEF])
```

(**COURIER.CHECK.ERRORS**

```
  [LAMBDA (DEFS)                                          (* bvm%: "12-Oct-84 11:24")
    (for ERRDEF in DEFS bind INFO unless [COND
                                          ((NLISTP ERRDEF)
                                           NIL)
                                          ((EQ (CAR ERRDEF)
                                               COMMENTFLG)        ; Comments ok
                                           T)
                                          (T (SETQ INFO (CDR ERRDEF))
                                             (COND
                                                ((AND (FIXP (fetch (COURIERERR ERR#) of INFO))
                                                      (NULLORLISTP (fetch (COURIERERR ARGS) of INFO)))
                                                                   ; nice new format
                                                 T)
                                                (T (COND
                                                      ((AND (EQ (CAR INFO)
                                                                'ARGS)
                                                            (NULLORLISTP (CADR INFO))
                                                            (FIXP (CADDR INFO)))
                                                                   ; Old format
                                                       (/RPLACD ERRDEF (create COURIERERR
                                                                               ERR# _ (CADDR INFO)
                                                                               ARGS _ (CADR INFO)))
                                                       T]
      do (ERROR "Bad Courier Error definition" ERRDEF])
```

(**COURIER.DELDEF**

```
  [LAMBDA (NAME TYPE)                                     (* bvm%: "15-Jun-84 15:34")
    (AND (EQ TYPE 'COURIERPROGRAMS)
         (PUTHASH NAME NIL \COURIERPROGRAM])
```

(**COURIER.GETDEF**

```
  [LAMBDA (NAME TYPE OPTIONS)                             (* bvm%: " 4-Jul-84 15:44")
    (AND (EQ TYPE 'COURIERPROGRAMS)
         (\GET.COURIERPROGRAM NAME])
```

(**COURIER.PUTDEF**

```
  [LAMBDA (NAME TYPE DEFINITION)                          ; Edited 15-Jun-88 12:13 by drc:
```

;;; PUTDEF for type COURIERPROGRAMS -- also called by COURIERPROGRAM

```
    (PROG (OLDINFO)
          (SETQ OLDINFO (GETHASH (SETQ NAME (\DTEST NAME 'LITATOM))
                                 \COURIERPROGRAM))
          [COND
             ((NULL OLDINFO)
              (MARKASCHANGED NAME TYPE 'DEFINED))
             ((AND OLDINFO (NOT (EQUAL OLDINFO DEFINITION)))
              (COND
                 ((NEQ DFNFLG T)
                  (EXEC-FORMAT "(Courier program ~S redefined)~%%" NAME)))
              (MARKASCHANGED NAME TYPE 'CHANGED]
          (/PUTHASH NAME DEFINITION \COURIERPROGRAM)
          (RETURN NAME])
```

(**DUMP.COURIERPROGRAMS**

```
  [NLAMBDA NAMES                                          (* bvm%: " 3-Oct-86 14:20")
```

```
    ;; Used by the COURIERPROGRAMS filepkgcom
    (for PROGRAM in NAMES bind PGMDEF do (COND
                                          ((SETQ PGMDEF (\GET.COURIERPROGRAM PROGRAM))
                                           (TERPRI)
                                           ;; because if you have a really bold font, it lines up the bottoms, but you can get crowded into
                                           ;; the line above.
                                           (printout NIL "(" .P2 'COURIERPROGRAM %, .FONT PRETTYCOMFONT .P2
                                                  PROGRAM .FONT DEFAULTFONT %, .P2 (CAR PGMDEF))
                                                                  ; Version pair
                                           (for TAIL on (CDR PGMDEF) by (CDDR TAIL)
                                              do (TAB 4)
                                                 (CHANGEFONT PRETTYCOMFONT)
                                                 (PRIN2 (CAR TAIL))
                                                                  ; Property name
                                                 (CHANGEFONT DEFAULTFONT)
                                                 (TAB 6)
                                                 (PRINTDEF (CADR TAIL)
                                                        6))
                                           (PRIN1 '%))
                                           (TERPRI))
                                          (T (CL:FORMAT T "(no COURIER definition for ~S)~%%" PROGRAM])

)

(DEFINEQ

(\GET.COURIER.TYPE
  [LAMBDA (PROGRAMNAME TYPENAME)                                    (* ecc " 7-JUL-83 14:34")
    (CAR (\GET.COURIER.DEFINITION PROGRAMNAME TYPENAME 'TYPES])

(\GET.COURIER.DEFINITION
  [LAMBDA (PROGRAM NAME TYPE PGMDEF)                                (* bvm%: "16-Jul-84 15:35")
    (COND
       ((\COURIER.QUALIFIED.NAMEP NAME)
        (\GET.COURIER.DEFINITION (CAR NAME)
               (CDR NAME)
               TYPE))
       (T (OR (CDR (ASSOC NAME (LISTGET [CDR (OR PGMDEF (SETQ PGMDEF (\GET.COURIERPROGRAM PROGRAM]
                                        TYPE)))
              (for OTHERPROGRAM in (LISTGET (CDR (OR PGMDEF (\GET.COURIERPROGRAM PROGRAM)))
                                            'INHERITS)
                 when [SETQ $$VAL (CDR (ASSOC NAME (LISTGET (CDR (\GET.COURIERPROGRAM OTHERPROGRAM))
                                                            TYPE]
                 do                                                 ; Is defined in an inherited program
                     (RETURN $$VAL))
              (ERROR (CONCAT "No " TYPE " definition for")
                     (LIST PROGRAM NAME])

)

;; COURIER record access

(DECLARE%: EVAL@COMPILE

(PUTPROPS COURIER.FETCH MACRO (ARGS (\COURIER.RECORDTRAN ARGS 'FETCH)))

(PUTPROPS COURIER.CREATE MACRO (ARGS (\COURIER.RECORDTRAN ARGS 'CREATE)))
)

(PUTPROPS COURIER.FETCH INFO NOEVAL)

(PUTPROPS COURIER.CREATE INFO NOEVAL)

(DEFINEQ

(\COURIER.RECORDTRAN
  [LAMBDA (ARGS OP)                                                 (* bvm%: " 4-Jul-84 15:42")
    (PROG ((PROGRAM (CAR ARGS))
           (REST (CDR ARGS))
           TYPEDEF)
          [SETQ TYPEDEF (COND
                           ((NLISTP PROGRAM)
                            (\GET.COURIER.TYPE PROGRAM (pop REST)))
                           ((\COURIER.QUALIFIED.NAMEP PROGRAM)
                            (SETQ TYPEDEF (CDR PROGRAM))
                            (\GET.COURIER.TYPE (SETQ PROGRAM (CAR PROGRAM))
                                   TYPEDEF))
                           (T (GO ERROR]
      LP  (COND
             ((NLISTP TYPEDEF)
              (SETQ TYPEDEF (\GET.COURIER.TYPE PROGRAM TYPEDEF))
              (GO LP))
             [(NEQ (CAR TYPEDEF)
                   'RECORD)
```

```
            (COND
               ((\COURIER.QUALIFIED.NAMEP TYPEDEF)
                (SETQ TYPEDEF (\GET.COURIER.TYPE (SETQ PROGRAM (CAR TYPEDEF))
                                  (CDR TYPEDEF)))
                (GO LP))
               (T (GO ERROR]
            (T (pop TYPEDEF)))
         (RETURN (SELECTQ OP
                     (FETCH                                    ; FETCH FIELD of DATUM --- DATUM is a list of values, one for
                                                               ; each field
                            (bind (FIELD _ (pop REST))
                                  (FORM _ (CAR REST)) first [SELECTQ FORM
                                                                    ((OF of)
                                                                     ; Noise word
                                                                     [COND
                                                                        ((AND (EQ FORM 'OF)
                                                                              LCASEFLG)
                                                                         (/RPLACA REST 'of]
                                                                     [SETQ FORM (CAR (SETQ REST (CDR REST])
                                                                    (COND
                                                                       ((EQ FORM 'of)
                                                                     ; Noise word
                                                                        (SETQ FORM (CAR (SETQ REST (CDR REST]
                                                                  (COND
                                                                     ((CDR REST)
                                                                     ; Too many args
                                                                      (GO ERROR)))
                                    while TYPEDEF do [COND
                                                        ((EQ (CAAR TYPEDEF)
                                                             FIELD)
                                                         (RETURN (LIST 'CAR FORM]
                                                       (SETQ FORM (LIST 'CDR FORM))
                                                       (SETQ TYPEDEF (CDR TYPEDEF))
                                    finally (GO ERROR)))
                     (CREATE                                   ; CREATE Field1 Value1 ... FieldN ValueN
                             [CONS 'LIST (bind (TAIL _ REST)
                                           X while TAIL
                                        collect [COND
                                                   ((NEQ (CAR TAIL)
                                                         (CAR (pop TYPEDEF)))
                                                    ; Fields not in order
                                                    (GO ERROR))
                                                   (T (PROG1 (COND
                                                                [(EQ [SETQ X (CAR (SETQ TAIL (CDR TAIL]
                                                                     '_)
                                                                 ; Noise token
                                                                 (CAR (SETQ TAIL (CDR TAIL]
                                                                (T X))
                                                             (SETQ TAIL (CDR TAIL)))]
                                        finally (COND
                                                   (TYPEDEF (GO ERROR])
                     (GO ERROR)))
      ERROR
          (ERROR "Invalid Courier Record Access form" (CONS OP ARGS])

)

;; COURIER calls and returns

(DEFMACRO STREAMTYPECASE (STREAM &BODY FORMS)
   `(AND (STREAMP ,STREAM)
         (SELECTQ (fetch (FDEV DEVICENAME) of (fetch (STREAM DEVICE) of ,STREAM))
             ,@FORMS)))

(DEFINEQ

(COURIER.OPEN
  [LAMBDA (HOSTNAME OBSOLETE NOERRORFLG NAME WHENCLOSEDFN OTHERPROPS)
                                                               ; Edited 28-Apr-92 17:34 by jds
                                                               ; Open a Courier connection to the specified host.
    (RESETLST
        (PROG (ADDRESS STREAM LOW.VERSION HIGH.VERSION)
              [COND
                 [(NOT (SETQ ADDRESS (\COERCE.TO.NSADDRESS HOSTNAME)))
                  (RETURN (AND (NOT NOERRORFLG)
                               (ERROR "Unknown host" HOSTNAME]
                 ([NULL (SETQ STREAM (SPP.OPEN ADDRESS \NS.WKS.Courier T NAME
                                               `(CLOSEFN ,(CONS (FUNCTION \COURIER.WHENCLOSED)
                                                                (MKLIST WHENCLOSEDFN))
                                                         ,@OTHERPROPS]
                  (RETURN (AND (NOT NOERRORFLG)
                               (ERROR "Host not responding" HOSTNAME]
              (RESETSAVE NIL (LIST (FUNCTION \SPP.CLOSE.IF.ERROR)
                                   STREAM))
              (replace ENDOFSTREAMOP of STREAM with (FUNCTION \COURIER.EOF))
              (SPP.DSTYPE STREAM \SPPDSTYPE.COURIER)
```

```
            [COND
               (COURIERTRACEFLG (printout COURIERTRACEFILE T "Opened " (OR NAME "")
                                          " with "
                                          (SPP.DESTADDRESS STREAM]
            (PUTWORD STREAM (SUB1 COURIER.VERSION#))          ; Lie about knowing an older version so as to demand a reply
                                                              ; immediately
            (PUTWORD STREAM COURIER.VERSION#)
            (SPP.SENDEOM STREAM)
            (SETQ LOW.VERSION (GETWORD STREAM))
            (SETQ HIGH.VERSION (GETWORD STREAM))
            [COND
               ((NOT (AND (ILEQ LOW.VERSION COURIER.VERSION#)
                          (ILEQ COURIER.VERSION# HIGH.VERSION)))
                (SPP.CLOSE STREAM)
                (RETURN (AND (NOT NOERRORFLG)
                             (ERROR "Server supports wrong version of Courier" (LIST HOSTNAME LOW.VERSION
                                                                                     HIGH.VERSION]
            (if (EQ (\SPP.PREPARE.INPUT STREAM 0)
                    'EOM)
                then ;; Clear the EOM with which the Alpine Sun server ends the version exchange (absent in Xerox D-machine servers).
                     ;; THIS SEEMS TO BE PART OF XNS NEW PROTOCOLS.
                     (SPP.CLEAREOM STREAM))
            (RETURN STREAM)))])
```

(\COURIER.WHENCLOSED

```
  [LAMBDA (STREAM CON)                                        (* ejs%: "27-May-86 11:24")
    (COND
       (COURIERTRACEFLG (printout COURIERTRACEFILE .TAB0 0 "Closed with " (STREAMTYPECASE STREAM
                                                                                (SPP (SPP.DESTADDRESS STREAM))
                                                                                (TCP (TCP.DESTADDRESS STREAM))
                                                                                "remote host")
                                  T])
```

(COURIER.CALL

```
  [LAMBDA ARGS                                                ; Edited 31-Jul-87 13:48 by bvm:
    ;; Call a Courier procedure: (COURIER.CALL stream program-name procedure-name arg1 ... argN)
    ;; Returns the result of the remote procedure, or a list of such results if it returns more than one.  A single flag NoErrorFlg can be optionally
    ;; appended to the arglist -- If NoErrorFlg is NOERROR, return NIL if the Courier program aborts with an error;  if RETURNERRORS, then return an
    ;; expression (ERROR ERRNAME . args) on error.  If the Courier procedure takes a Bulk Data parameter, then the result of COURIER.CALL is a
    ;; stream for the transfer.  When the stream is closed, the results will be read and the functional argument that was supplied in the call, if any, will
    ;; be applied to the results.
    (LET ((STREAM (ARG ARGS 1))
          (PROGRAM (ARG ARGS 2))
          (PROCEDURE (ARG ARGS 3))
          NARGS ARGLIST NOERRORFLG PGMDEF PROCDEF ARGTYPES)
         (SETQ PGMDEF (OR (\GET.COURIERPROGRAM PROGRAM)
                          (ERROR "No such Courier program" PROGRAM)))
         (SETQ PROCDEF (\GET.COURIER.DEFINITION PROGRAM PROCEDURE 'PROCEDURES PGMDEF))
         [SETQ NARGS (LENGTH (SETQ ARGTYPES (fetch (COURIERFN ARGS) of PROCDEF]
         (OR (SELECTQ (- ARGS NARGS)
                  (3                                          ; Exactly right
                     T)
                  (4                                          ; Extra arg is errorflg
                     (SELECTQ (SETQ NOERRORFLG (ARG ARGS (+ NARGS 4)))
                          ((NOERROR RETURNERRORS T)           ; The only valid values
                              T)
                          NIL))
                  NIL)
             (ERROR "Wrong number of arguments to Courier procedure" (CONS PROGRAM PROCEDURE)))
         (SETQ ARGLIST (for I from 4 to (+ NARGS 3) collect (ARG ARGS I)))
         (COND
            ((type? STREAM STREAM)
             (COURIER.EXECUTE.CALL STREAM PROGRAM PGMDEF PROCEDURE PROCDEF ARGLIST ARGTYPES NOERRORFLG))
            [(type? NSADDRESS STREAM)                         ; Means to make a single call to this address
             (RESETLST
                 [LET ((STREAM (COURIER.OPEN STREAM NIL NOERRORFLG)))
                      (COND
                         (STREAM (RESETSAVE NIL (LIST (STREAMTYPECASE STREAM (SPP (FUNCTION \SPP.RESETCLOSE))
                                                                             (TCP (FUNCTION \TCP.RESETCLOSE))
                                                                             (FUNCTION CLOSEF))
                                                      STREAM))
                                 (COURIER.EXECUTE.CALL STREAM PROGRAM PGMDEF PROCEDURE PROCDEF ARGLIST ARGTYPES
                                        NOERRORFLG))
                         ((EQ NOERRORFLG 'RETURNERRORS)
                          '(ERROR CONNECTION.PROBLEM NoResponse])]
            ((NEQ NOERRORFLG 'NOERROR)
             (\ILLEGAL.ARG STREAM])
```

(COURIER.EXECUTE.CALL

```
  [LAMBDA (STREAM PROGRAM PGMDEF PROCEDURE PROCDEF ARGLIST ARGTYPES NOERRORFLG)
                                                              ; Edited 21-Jul-87 14:44 by bvm:
```

```
;; Send the arguments for a Courier call to the remote program.  Returns NIL if none of the formal parameters are of type BULK.DATA.SOURCE or
;; BULK.DATA.SINK, otherwise returns a stream for the Bulk Data transfer.

(COND
   (COURIERTRACEFLG (\COURIER.TRACE 'CALL PROGRAM PROCEDURE ARGLIST)))
(PROG ((OUTSTREAM STREAM)
       SOURCEFLG SINKFLG BULKDATAFN DATASTREAM)
      (STREAMTYPECASE STREAM (SPP (SPP.DSTYPE (SETQ OUTSTREAM (SPPOUTPUTSTREAM STREAM))
                                              \SPPDSTYPE.COURIER))
            NIL)
      (PUTWORD OUTSTREAM \COURIERMSG.CALL)
      (PUTWORD OUTSTREAM 0)                                    ; Transaction ID, ignored for now.
      (PUTLONG OUTSTREAM (fetch (COURIERPGM PROGRAM#) of PGMDEF))
      (PUTWORD OUTSTREAM (fetch (COURIERPGM VERSION#) of PGMDEF))
      (PUTWORD OUTSTREAM (fetch (COURIERFN FN#) of PROCDEF))
      (for VALUE in ARGLIST as TYPE in ARGTYPES do (SELECTQ TYPE
                                                           (BULK.DATA.SOURCE
                                                                (SETQ SOURCEFLG T)
                                                                (SETQ BULKDATAFN VALUE)
                                                                (PUTWORD OUTSTREAM 1))
                                                           (BULK.DATA.SINK
                                                                (SETQ SINKFLG T)
                                                                (SETQ BULKDATAFN VALUE)
                                                                (PUTWORD OUTSTREAM 1))
                                                           (COURIER.WRITE OUTSTREAM VALUE PROGRAM TYPE)))
      (STREAMTYPECASE OUTSTREAM (SPP (SPP.SENDEOM OUTSTREAM))
            (TCP (\TCP.FORCEOUTPUT OUTSTREAM))
            (FORCEOUTPUT OUTSTREAM))
      (CHECK (NOT (AND SOURCEFLG SINKFLG)))
      (RETURN (COND
                 ((AND (OR SOURCEFLG SINKFLG)
                       (SETQ DATASTREAM (\BULK.DATA.STREAM STREAM (COND
                                                                     (SINKFLG 'INPUT)
                                                                     (T 'OUTPUT))
                                                          PROGRAM PROCEDURE PGMDEF PROCDEF NOERRORFLG BULKDATAFN)))
                  (COND
                     (BULKDATAFN (\COURIER.HANDLE.BULKDATA DATASTREAM BULKDATAFN NOERRORFLG))
                     (T                                    ; Return the stream to caller
                        DATASTREAM)))
                 (T (\COURIER.RESULTS STREAM PROGRAM PGMDEF PROCEDURE PROCDEF NOERRORFLG])
```

(\COURIER.RESULTS

```
  [LAMBDA (STREAM PROGRAM PGMDEF PROCEDURE PROCDEF NOERRORFLG EXPEDITEDFLG)
                                                           ; Edited  1-May-87 11:39 by bvm:
    (LET (MSGTYPE RESULT)
         (SETQ RESULT (SELECTC (SETQ MSGTYPE (GETWORD STREAM))
                               (\COURIERMSG.RETURN
                                    [LET ((RESULTTYPES (fetch (COURIERFN RESULTS) of PROCDEF)))
                                         (GETWORD STREAM)          ; Skip the Transaction ID.
                                         (COND
                                            ((AND RESULTTYPES (NOT (CDR RESULTTYPES)))
                                                                   ; Single-valued procedures return conventionally
                                             (COURIER.READ STREAM PROGRAM (CAR RESULTTYPES)))
                                            (T (for TYPE in RESULTTYPES collect (COURIER.READ STREAM PROGRAM TYPE]
                               (\COURIERMSG.ABORT
                                    (GETWORD STREAM)               ; Skip the Transaction ID.
                                    [LET ((NUMBER (GETWORD STREAM))
                                          ERRORDEF)
                                         (CONS 'ERROR (COND
                                                         [(SETQ ERRORDEF
                                                            (find ERR
                                                               in (OR (fetch (COURIERPGM ERRORS) of PGMDEF)
                                                                      (for OTHER in (fetch (COURIERPGM INHERITS)
                                                                                       of PGMDEF)
                                                                         when (SETQ $$VAL (fetch (COURIERPGM ERRORS)
                                                                                             of (\GET.COURIERPROGRAM
                                                                                                   OTHER)))
                                                                         do (RETURN $$VAL)))
                                                               suchthat (IEQP (fetch (COURIERERR ERR#)
                                                                                 of (CDR ERR))
                                                                              NUMBER)))
                                                          (CONS (CAR ERRORDEF)
                                                                (for TYPE in (fetch (COURIERERR ARGS)
                                                                                of (CDR ERRORDEF))
                                                                   collect (COURIER.READ STREAM PROGRAM TYPE]
                                                         (T (LIST NUMBER])
                               (\COURIERMSG.REJECT
                                    (GETWORD STREAM)               ; Skip the Transaction ID.
                                    [LIST 'ERROR 'REJECT (COURIER.READ STREAM PROGRAM
                                                                '(CHOICE (NoSuchService 0)
                                                                         (WrongVersionOfService 1 (RECORD (lowest
                                                                                                            CARDINAL)
                                                                                                          (highest
                                                                                                            CARDINAL)))
                                                                         (NoSuchProcedure 2)
                                                                         (invalidArguments 3)
                                                                         (unspecifiedError 65535])
```

```
                        (LIST 'ERROR 'UnknownResponseType MSGTYPE)))
        (COND
           ((NOT EXPEDITEDFLG)
            (STREAMTYPECASE STREAM (SPP (SPP.CLEAREOM STREAM))
                   NIL)))
        (COND
           (COURIERTRACEFLG (\COURIER.TRACE 'RETURN PROGRAM PROCEDURE RESULT)))
        (COND
           ((EQ MSGTYPE \COURIERMSG.RETURN)                    ; Normal return
            RESULT)
           ((AND EXPEDITEDFLG (EQ (CADDR RESULT)
                                  'USE.COURIER))               ; Special flag on expedited courier call saying to use regular
                                                               ; Courier
            'USE.COURIER)
           (T (SELECTQ NOERRORFLG
                  (RETURNERRORS                                ; Caller wants to handle errors
                                RESULT)
                  (NIL                                         ; Default--signal the error
                       (COURIER.SIGNAL.ERROR PROGRAM PROCEDURE RESULT))
                  (PROGN                                       ; Caller wants no errors
                         (\COURIER.HANDLE.ERROR PROGRAM PROCEDURE RESULT)
                         NIL])
```

(COURIER.SIGNAL.ERROR

```
[LAMBDA (PROGRAM PROCEDURE ERRORFORM)                          ; Edited  1-May-87 11:33 by bvm:
```

;; Signals the error returned from PROCEDURE of PROGRAM. ERRORFORM is a form starting with the symbol ERROR from a Courier result.

```
(LET ((ARGS (CDR ERRORFORM)))
     (ERROR (CONCAT (COND
                       ((EQ (CAR ARGS)
                            'REJECT)                           ; Reject errors of form (ERROR REJECT reason)
                        (SETQ ARGS (CADR ARGS))
                        "Courier rejected call to ")
                       (T                                      ; Other errors of form (ERROR type . args)
                          [COND
                             ((NULL (CDR ARGS))                ; For errors with no arguments, make the error call slighly prettier
                                                               ; by just naming the error
                              (SETQ ARGS (CAR ARGS]
                             "Error in Courier procedure "))
                    PROGRAM "." PROCEDURE)
            ARGS])
```

(\COURIER.HANDLE.BULKDATA

```
[LAMBDA (DATASTREAM BULKDATAFN NOERRORFLG)                     ; Edited 27-Aug-87 11:26 by bvm:
```

;;; Called when a Courier call has a bulkdata argument. BULKDATAFN is a function to apply to the bulk data stream. If it returns a non-NIL result, that ;;; is returned as the value of the Courier call, ignoring the Courier results, if any. As a special case, a BULKDATAFN of (Program . Type) interprets the ;;; bulk data stream as a 'Stream of Program.Type'

```
    (CL:UNWIND-PROTECT
        (CL:MULTIPLE-VALUE-BIND (BULKRESULTS ERROR)
            (CL:CATCH :BULKDATA
                (COND
                   ((AND (LISTP BULKDATAFN)
                         (SELECTQ (CAR BULKDATAFN)
                              ([LAMBDA CL:LAMBDA]              ; Handler is not a type, just an interpreted fn
                                 NIL)
                              T))                              ; Special case, interpret as a type
                    (COURIER.READ.BULKDATA DATASTREAM (CAR BULKDATAFN)
                           (CDR BULKDATAFN)
                           T))
                   (T (CL:FUNCALL BULKDATAFN DATASTREAM))))
```

;; Bulk data handled now. If handler wanted to abort, then BULKRESULTS is :ABORT, in which case we send an abort packet (if ;; necessary), and the second value ERROR is optional error value to return.

```
            (LET [(MAINRESULTS (\BULK.DATA.CLOSE DATASTREAM (AND (EQ BULKRESULTS :ABORT)
                                                                 (OR NOERRORFLG T]
                 (OR (AND (NEQ BULKRESULTS :ABORT)
                          BULKRESULTS)
                     ERROR MAINRESULTS)))
```

;; Be sure bulk stream is closed on exit. This is a no-op on normal exit, since the stream has already been closed. On error exit, we send an ;; abort.

```
            (\BULK.DATA.CLOSE DATASTREAM T))])
```

(\COURIER.HANDLE.ERROR

```
[LAMBDA (PROGRAM PROCEDURE ERRORARGS)                          (* bvm%: "27-Jun-84 23:05")
  (COND
     (NSWIZARDFLG (printout PROMPTWINDOW .TAB0 0 "Error in Courier program " PROGRAM ", procedure " PROCEDURE
                    ": " ERRORARGS])
```

(\BULK.DATA.STREAM

```
[LAMBDA (STREAM MODE PROGRAM PROCEDURE PGMDEF PROCDEF NOERRORFLG INTERNALFLG)
```

```
                                                                ; Edited 20-May-87 12:33 by bvm:
    ;; Return a specialized version of an SPP stream suitable for sending or receiving a Bulk Data object.  Uses the Bulk Data device, which redefines
    ;; the EOFP and CLOSE functions.  Save the program, procedure, and result function in the stream record for use by \BULK.DATA.CLOSE.
    (STREAMTYPECASE STREAM
           (SPP (PROG ((CON (GETSPPCON STREAM))
                       SUBSTREAM NEXTPKT)
                      [COND
                         ((EQ MODE 'INPUT)                      ; Preview the incoming stream to see if there's any data there
                          (COND
                             ((NOT (SETQ NEXTPKT (\GETSPP CON NIL T)))
                                                                ; Connection died
                              (RETURN NIL))
                             ((NEQ (fetch (SPPXIP DSTYPE) of NEXTPKT)
                                   \SPPDSTYPE.BULKDATA)          ; Bulkdata not coming, must be error
                              (RETURN NIL))
                             ((fetch (SPPXIP ATTENTION) of NEXTPKT)
                                                                ; Immediately aborted, must be nothing coming
                              (\GETSPP CON)                     ; Eat the packet
                              (RETURN NIL]
                      (COND
                         ((type? STREAM (SETQ SUBSTREAM (fetch F10 of STREAM)))
                                                                ; reuse old substream
                          (replace F10 of STREAM with NIL)
                          (replace SPPFILEPTRHI of SUBSTREAM with 0)
                          (replace SPPFILEPTRLO of SUBSTREAM with 0)
                          (replace SPPEOFP of SUBSTREAM with NIL))
                         (T (SETQ SUBSTREAM (create STREAM
                                                    DEVICE _ \SPP.BULKDATA.DEVICE))
                            (replace SPP.CONNECTION of SUBSTREAM with CON)))
                      (replace BULK.DATA.CONTINUATION of SUBSTREAM
                         with (create \BULK.DATA.CONTINUATION
                                      PROGRAM _ PROGRAM
                                      PROCEDURE _ PROCEDURE
                                      PGMDEF _ PGMDEF
                                      PROCDEF _ PROCDEF
                                      NOERRORFLG _ NOERRORFLG
                                      INTERNALFLG _ INTERNALFLG))
                      (replace (STREAM ACCESS) of SUBSTREAM with MODE)
                      (replace SPPSUBSTREAM of CON with SUBSTREAM)
                      (replace SPPATTENTIONFN of CON with (FUNCTION \COURIER.ATTENTIONFN))
                      (COND
                         (COURIERTRACEFLG (\COURIER.TRACE 'BEGIN.BULK.DATA PROGRAM PROCEDURE)))
                      (SPP.DSTYPE SUBSTREAM \SPPDSTYPE.BULKDATA)
                      (RETURN SUBSTREAM)))
           (ERROR "Courier bulk data not supported on stream of type" (fetch (FDEV DEVICENAME)
                                                                             of (fetch (STREAM DEVICE) of STREAM])
```

(**COURIER.ATTENTIONFN**

```
  [LAMBDA (STREAM BYTE DSTYPE)                                  (* bvm%: "12-Oct-84 16:16")
```

;;; Called when attention packet received on input STREAM. If we are currently writing bulkdata, this is an abort, so arrange to kill the writer

```
    (COND
       ((AND (EQ BYTE 1)
             (EQ DSTYPE \SPPDSTYPE.BULKDATA))                   ; Bulk data stream truncation signal
        (LET (CON)
             (COND
                ((AND (SETQ CON (GETSPPCON STREAM))
                       (SETQ STREAM (fetch SPPSUBSTREAM of CON))
                       (WRITEABLE STREAM))
                 (replace SPPOUTPUTABORTEDFN of CON with (FUNCTION \COURIER.OUTPUT.ABORTED))
                 (replace SPPOUTPUTABORTEDP of CON with T)))
             (COND
                (NSWIZARDFLG (printout PROMPTWINDOW .TAB0 0 "[Remote host aborted data transfer]")))
             T])
```

(**COURIER.OUTPUT.ABORTED**

```
  [LAMBDA (STREAM)                                              ; Edited 18-May-87 17:07 by bvm:
```

;; Called when attempt is made to write data on STREAM when output has been aborted, or to read from a stream that is at ATTN (bulk data
;; abort).

```
    (LET (FILENAME CONTINUATION RESULT)
         (COND
            [(AND (SETQ CONTINUATION (fetch BULK.DATA.CONTINUATION of STREAM))
                  (NOT (fetch INTERNALFLG of CONTINUATION)))    ; This was a standalone bulkdata stream
             (SETQ RESULT (\BULK.DATA.CLOSE STREAM 'RETURNERRORS))
             (COND
                ((AND (SETQ FILENAME (fetch FULLFILENAME of STREAM))
                       (EQ (CADR RESULT)
                           'SPACE.ERROR))
                 (LISPERROR "FILE SYSTEM RESOURCES EXCEEDED" FILENAME))
                (T (ERROR (CONCAT (COND
                                     ((DIRTYABLE STREAM)
                                      "Output")
```

```
                                    (T "Input"))
                                " aborted: "
                                (CADR RESULT)
                                " -- "
                                (CADDR RESULT))
                            (OR FILENAME STREAM]
          (T                                                    ; Inside of \COURIER.HANDLE.BULKDATA
             (CL:THROW :BULKDATA :ABORT])
```

(**BULK.DATA.CLOSE**

```
[LAMBDA (STREAM ABORTFLG)                                       ; Edited 27-Aug-87 11:29 by bvm:
```

;; Close a Bulk Data stream after the transfer has taken place. If a result function was specified in COURIER.CALL, call it on the stream and the ;; result or list of results.

```
(PROG ((CON (GETSPPCON STREAM))
       (CONTINUATION (fetch BULK.DATA.CONTINUATION of STREAM)))
      (replace SPPATTENTIONFN of CON with NIL)
      (COND
         ((NULL (fetch SPPSUBSTREAM of CON))                    ; This stream has already been closed.  We don't want to try to
                                                                ; read the Courier results twice
          (RETURN)))
      [COND
         (COURIERTRACEFLG (\COURIER.TRACE 'END.BULK.DATA (fetch PROGRAM of CONTINUATION)
                                  (fetch PROCEDURE of CONTINUATION]
      (COND
         [(WRITEABLE STREAM)
          (COND
             (ABORTFLG (SPP.SENDATTENTION STREAM 1))
             (T (SPP.SENDEOM STREAM]
         ((NOT (\EOFP STREAM))                                  ; Closing before all the data has been read -- abort the transfer.
          (OR ABORTFLG (SETQ ABORTFLG T))
          (\ABORT.BULK.DATA STREAM)))
      (replace BULK.DATA.CONTINUATION of STREAM with NIL)       ; Tell SPP handler not to take any more bulk data packets.
      (replace SPPINPKT of CON with NIL)
```

;; This stream is closing; make sure there aren't any dangling pointers into the middle of ether packets.

```
      (replace CBUFPTR of STREAM with NIL)
      (replace CBUFSIZE of STREAM with 0)
      (RETURN (CAR (ERSETQ (RESETLST
```

;; The result of the Courier call may be an error which the user should see; however, we still need to ;; clean up the substream, so we wrap it in this RESETLST.

```
                [LET ((COURIERSTREAM (fetch SPPINPUTSTREAM of CON)))
                     (RESETSAVE NIL (LIST [FUNCTION (LAMBDA (STRM ABORTFLG)
                                                            [COND
                                                               (ABORTFLG
                                                                (replace ENDOFSTREAMOP of STRM
                                                                   with (FUNCTION \COURIER.EOF]
                                                            (COND
                                                               (RESETSTATE (SPP.CLOSE STRM T]
                                          COURIERSTREAM ABORTFLG))
                     [COND
                        (ABORTFLG (replace ENDOFSTREAMOP of COURIERSTREAM
                                     with (FUNCTION ERROR!]
                     (replace SPPSUBSTREAM of CON with NIL)
                     (PROG1 (\COURIER.RESULTS COURIERSTREAM (fetch PROGRAM of CONTINUATION)
                                  (fetch PGMDEF of CONTINUATION)
                                  (fetch PROCEDURE of CONTINUATION)
                                  (fetch PROCDEF of CONTINUATION)
                                  (OR ABORTFLG (fetch NOERRORFLG of CONTINUATION)))
                         (COND
                            ((NOT (fetch FULLFILENAME of STREAM))
                                                            ; On normal exit, save the substream for later reuse.
                             (replace F10 of COURIERSTREAM with STREAM))))])])
```

(**ABORT.BULK.DATA**

```
[LAMBDA (STREAM)                                                (* ejs%: "18-Dec-84 17:32")
  (PROG (EPKT)
        (do                                                     ; Empty queue of waiting packets without blocking.
            (replace COFFSET of STREAM with (fetch CBUFSIZE of STREAM)) repeatwhile (NOT (\SPP.PREPARE.INPUT STREAM
                                                                                             0)))
        (COND
           ((fetch SPPEOFP of STREAM)                           ; We've already received the last packet of the Bulk Data
                                                                ; transfer.
            )
           (T ;; Abort the bulk data stream by sending an Attention packet with a 1 in it.  WARNING: if the EOM bit is set in this packet, the NS
              ;; fileserver will crash.
              (SPP.SENDATTENTION STREAM 1)
              (if NIL
                  then ;; Ignore any remaining bulk data packets -- there shouldn't be many if the other end is obeying the protocol.
                       (while (\SPP.PREPARE.INPUT STREAM SPP.USER.TIMEOUT])
```

```
)

(DEFINEQ
```

(COURIER.EXPEDITED.CALL

```
  [LAMBDA ARGS                                                    (* bvm%: "16-Jul-84 15:39")
```

;;; Like COURIER.CALL but tries to use 'expedited' calls. The first two args are the address and socket# to talk to, rather than a single open Courier ;;; stream. Remaining args are identical. If expedited version fails, a regular courier call is executed. Bulk data is prohibited

```
    (PROG ((ADDRESS (ARG ARGS 1))
           (SOCKET# (ARG ARGS 2))
           (PROGRAM (ARG ARGS 3))
           (PROCEDURE (ARG ARGS 4))
           %#ARGS ARGLIST NOERRORFLG PGMDEF PROCDEF ARGTYPES)
          (SETQ PGMDEF (OR (\GET.COURIERPROGRAM PROGRAM)
                           (ERROR "No such Courier program" PROGRAM)))
          (SETQ PROCDEF (\GET.COURIER.DEFINITION PROGRAM PROCEDURE 'PROCEDURES PGMDEF))
          [SETQ %#ARGS (LENGTH (SETQ ARGTYPES (fetch (COURIERFN ARGS) of PROCDEF]
          [COND
             ([for TYPE in ARGTYPES thereis (OR (EQ TYPE 'BULK.DATA.SINK)
                                                (EQ TYPE 'BULK.DATA.SOURCE]
              (ERROR "Can't transfer bulk data with expedited call" (CONS PROGRAM PROCEDURE]
          (OR (SELECTQ (IDIFFERENCE ARGS %#ARGS)
                       (4                                         ; Exactly right
                          T)
                       (5                                         ; Extra arg is errorflg
                          (SELECTQ (SETQ NOERRORFLG (ARG ARGS (IPLUS %#ARGS 5)))
                                   ((NOERROR RETURNERRORS T)
                                       T)
                                   NIL))
                       NIL)
              (ERROR "Wrong number of arguments to Courier procedure" (CONS PROGRAM PROCEDURE)))
          (SETQ ARGLIST (for I from 5 to (IPLUS %#ARGS 4) collect (ARG ARGS I)))
          (RETURN (COURIER.EXECUTE.EXPEDITED.CALL ADDRESS SOCKET# PROGRAM PGMDEF PROCEDURE PROCDEF ARGLIST
                         ARGTYPES NOERRORFLG])
```

(COURIER.EXECUTE.EXPEDITED.CALL

```
  [LAMBDA (ADDRESS SOCKET# PROGRAM PGMDEF PROCEDURE PROCDEF ARGLIST ARGTYPES NOERRORFLG)
                                                                  ; Edited 31-Jul-87 14:19 by bvm:
```

;;; Attempts the actual expedited call

```
    (COND
       (COURIERTRACEFLG (\COURIER.TRACE 'CALL PROGRAM PROCEDURE ARGLIST)))
    (RESETLST
        [PROG ((NSOC (OPENNSOCKET))
               XIP STREAM RESULT)
              (RESETSAVE NIL (LIST 'CLOSENSOCKET NSOC))
              (SETQ XIP (CREATE.PACKET.EXCHANGE.XIP NSOC ADDRESS SOCKET# \EXTYPE.EXPEDITED.COURIER))
              (OR (\BUILD.EXPEDITED.XIP XIP PROGRAM PGMDEF PROCDEF ARGLIST ARGTYPES)
                  (GO USECOURIER))
              (COND
                 ((NEQ (SETQ RESULT (\SEND.EXPEDITED.XIP XIP NSOC PROGRAM PGMDEF PROCEDURE PROCDEF NOERRORFLG))
                       'USE.COURIER)
                  (RETURN RESULT)))
          USECOURIER
              (RETURN (COND
                          ((SETQ STREAM (COURIER.OPEN ADDRESS NIL NOERRORFLG 'COURIER))
                                                                  ; Use regular courier
                           (RESETSAVE NIL (LIST (FUNCTION \SPP.RESETCLOSE)
                                                STREAM))
                           (COURIER.EXECUTE.CALL STREAM PROGRAM PGMDEF PROCEDURE PROCDEF ARGLIST ARGTYPES
                                 NOERRORFLG))
                          ((EQ NOERRORFLG 'RETURNERRORS)
                           '(ERROR CONNECTION.PROBLEM NoResponse])])
```

(\BUILD.EXPEDITED.XIP

```
  [LAMBDA (XIP PROGRAM PGMDEF PROCDEF ARGLIST)                    (* bvm%: " 4-Jul-84 15:41")
    (PROG (STREAM)
          (replace (EXPEDITEDXIP LOWVERSION) of XIP with (replace (EXPEDITEDXIP HIGHVERSION) of XIP with (
                                                                                  COURIER.VERSION#
                                                                                      )))
          (replace (EXPEDITEDXIP MSGTYPE) of XIP with \COURIERMSG.CALL)
          (replace (EXPEDITEDXIP TRANSACTIONID) of XIP with 0)      ; Transaction ID, ignored for now.
          (replace (EXPEDITEDXIP PROGRAM#) of XIP with (fetch (COURIERPGM PROGRAM#) of PGMDEF))
          (replace (EXPEDITEDXIP VERSION#) of XIP with (fetch (COURIERPGM VERSION#) of PGMDEF))
          (replace (EXPEDITEDXIP PROCEDURE#) of XIP with (fetch (COURIERFN FN#) of PROCDEF))
          [replace XIPLENGTH of XIP with (COND
                                            (ARGLIST (SETQ STREAM (\MAKE.EXPEDITED.STREAM XIP 'OUTPUT))
                                                   (OR (\COURIER.EXPEDITED.ARGS STREAM PROGRAM ARGLIST
                                                                (fetch (COURIERFN ARGS) of PROCDEF))
                                                       (RETURN))
                                                   (fetch COFFSET of STREAM))
                                            (T (IPLUS \XIPOVLEN (UNFOLD (IPLUS 3 (INDEXF (fetch (EXPEDITEDXIP ARG0)
```

```
                                                                                    of T)))
                                                                          BYTESPERWORD]
        (RETURN XIP])
```

(\SEND.EXPEDITED.XIP

```
[LAMBDA (XIP NSOC PROGRAM PGMDEF PROCEDURE PROCDEF NOERRORFLG %#TRIES)
                                                                    (* bvm%: "21-Feb-86 14:21")
```

;;; Sends XIP, which is in the form of an expedited courier call, and awaits a response on NSOC. The call is to PROCEDURE of PROGRAM. If there is ;;; no response, or the remote element responds with the USE.COURIER error, returns USE.COURIER else the actual result (which could be NIL)

```
    (bind (TIMER _ (SETUPTIMER 0))
          (EVENT _ (NSOCKETEVENT NSOC))
          (ID _ (fetch PACKETEXCHANGEID of XIP))
          IXIP to (OR %#TRIES \MAXETHERTRIES)
       do (SENDXIP NSOC XIP)
          (SETUPTIMER \ETHERTIMEOUT TIMER)
          [SELECTQ (until (TIMEREXPIRED? TIMER) when (PROGN (AWAIT.EVENT EVENT TIMER T)
                                                            (SETQ IXIP (GETXIP NSOC)))
                      do (SELECTC (fetch XIPTYPE of IXIP)
                              (\XIPT.EXCHANGE
                                   (COND
                                      ((AND (IEQP (fetch PACKETEXCHANGEID of IXIP)
                                                  ID)
                                            (ILEQ (fetch (EXPEDITEDXIP LOWVERSION) of IXIP)
                                                  (COURIER.VERSION#))
                                            (IGEQ (fetch (EXPEDITEDXIP HIGHVERSION) of IXIP)
                                                  (COURIER.VERSION#))
                                            (SELECTC (fetch (EXPEDITEDXIP MSGTYPE) of IXIP)
                                                 ((LIST \COURIERMSG.RETURN \COURIERMSG.REJECT \COURIERMSG.ABORT)

                                                      T)
                                                 NIL))
                                       (RETURN T))))
                              (\XIPT.ERROR [COND
                                              ((AND (EQ (fetch ERRORXIPCODE of IXIP)
                                                        \XIPE.NOSOCKET)
                                                    (NOT (EQNSHOSTNUMBER (fetch XIPDESTHOST of XIP)
                                                                         BROADCASTNSHOSTNUMBER)))
                                               ;; Not responding to calls on this socket.  If XIP were a broadcast, nobody should be replying with an
                                               ;; error, but if some loser did, we should ignore it
                                               (RELEASE.XIP IXIP)
                                               (RETURN 'USE.COURIER])
                              NIL)
                           (RELEASE.XIP IXIP))
          (USE.COURIER (RETURN 'USE.COURIER))
          (NIL                                                  ; Keep trying
               )
          (RETURN (PROG1 (\COURIER.RESULTS (\MAKE.EXPEDITED.STREAM IXIP 'INPUT)
                                 PROGRAM PGMDEF PROCEDURE PROCDEF NOERRORFLG T)
                         (RELEASE.XIP IXIP]
       finally (RETURN 'USE.COURIER])
```

(\COURIER.EXPEDITED.ARGS

```
[LAMBDA (STREAM PROGRAM ARGLIST ARGTYPES)                           (* bvm%: "15-Jun-84 12:00")
```

;;; Store the args for an expedited call into packet addressed by STREAM. Returns T on success. Failure is indicated by a RETFROM this fn with value ;;; NIL

```
    (for VALUE in ARGLIST as TYPE in ARGTYPES do (COURIER.WRITE STREAM VALUE PROGRAM TYPE))
    T])
```

(\MAKE.EXPEDITED.STREAM

```
[LAMBDA (XIP ACCESS OSTREAM)                                        ; Edited  2-Nov-93 13:52 by sybalsky:mv:envos
```

;;; Makes a STREAM to access the contents of XIP as an expedited courier message body. We use the BASEBYTES device for simplicity. All the ;;; operations we actually need are BIN, BOUT, BLOCKIN and BLOCKOUT

```
    (PROG ([STREAM (OR OSTREAM (NCREATE 'STREAM]
           END)
          (replace (STREAM DEVICE) of STREAM with \BASEBYTESDEVICE)
          (replace (STREAM ACCESS) of STREAM with ACCESS)
          (replace (STREAM CBUFPTR) of STREAM with (fetch (XIP XIPBASE) of XIP))
          [replace (STREAM COFFSET) of STREAM with (IPLUS \XIPOVLEN (UNFOLD 3 BYTESPERWORD)
                                                          (COND
                                                             ((EQ ACCESS 'INPUT)
                                                                   ; For COURIER.RESULTS
                                                              (SETQ END (fetch XIPLENGTH of XIP))
                                                              (UNFOLD (INDEXF (fetch (EXPEDITEDXIP MSGTYPE)
                                                                                     of T))
                                                                      BYTESPERWORD))
                                                             (T    ; For COURIER.EXPEDITED.ARGS
                                                                (SETQ END (IPLUS \MAX.XIPDATALENGTH \XIPOVLEN))
```

```
                                                        (UNFOLD (INDEXF (fetch (EXPEDITEDXIP ARG0)
                                                                        of T))
                                                                BYTESPERWORD]
            (replace (STREAM EOFFSET) of STREAM with (replace CBUFSIZE of STREAM with END))
            [COND
               ((EQ ACCESS 'INPUT)                              ; Will cause error if COURIER.RESULTS tries to read more than
                                                                ; was sent -- should never happen
                (replace (STREAM ENDOFSTREAMOP) of STREAM with (FUNCTION \COURIER.EOF)))
               (T                                               ; Invoked if COURIER.EXPEDITED.ARGS tries to write more
                                                                ; than will fit in the packet
                  (replace (BASEBYTESTREAM WRITEXTENSIONFN) of STREAM with (FUNCTION \COURIER.EXPEDITED.OVERFLOW]
            (RETURN STREAM])
```

(\COURIER.EOF

```
  [LAMBDA (STREAM)                                              (* bvm%: "15-Jun-84 11:56")
```

;;; Called if we attempt to read beyond the end of a courier response

```
    (ERROR "Unexpected end of stream while reading Courier response"])
```

(\COURIER.EXPEDITED.OVERFLOW

```
  [LAMBDA (STREAM)                                              (* bvm%: " 4-Jul-84 15:41")
```

;;; Called when \COURIER.EXPEDITED.ARGS tries to write beyond the end of the packet

```
    (COND
       (NSWIZARDFLG (printout PROMPTWINDOW T "[Expedited call did not fit in one packet]")))
    (RETFROM (FUNCTION \COURIER.EXPEDITED.ARGS)
           NIL])

)

(DEFINEQ
```

(COURIER.BROADCAST.CALL

```
  [LAMBDA (DESTSOCKET# PROGRAM PROCEDURE ARGS RESULTFN NETHINT MESSAGE)
                                                                (* bvm%: "21-Feb-86 14:24")
```

;; Performs expanding ring broadcast for Courier PROCEDURE applied to ARGS. If RESULTFN is given, it is applied to the results of the courier
;; call, and its result is returned, unless it is NIL, in which case the broadcast continues. NETHINT is a net or list of nets that are expected to have
;; the desired server. If omitted, or if no server on those nets responds, broadcast starts with the connected net and expands outward

```
    (RESETLST
        (PROG ((PGMDEF (OR (\GET.COURIERPROGRAM PROGRAM)
                           (ERROR "No such Courier program" PROGRAM)))
               PROCDEF SKT EPKT ROUTINGTABLE RESULT NEARBYNETS)
              (DECLARE (SPECVARS NEARBYNETS))                   ; For \MAP.ROUTING.TABLE
              (SETQ PROCDEF (\GET.COURIER.DEFINITION PROGRAM PROCEDURE 'PROCEDURES PGMDEF))
              [RESETSAVE NIL (LIST 'CLOSENSOCKET (SETQ SKT (OPENNSOCKET]
              (SETQ EPKT (CREATE.PACKET.EXCHANGE.XIP SKT BROADCASTNSHOSTNUMBER DESTSOCKET#
                                \EXTYPE.EXPEDITED.COURIER))
              (OR (\BUILD.EXPEDITED.XIP EPKT PROGRAM PGMDEF PROCDEF ARGS)
                  (ERROR "Could not build broadcast for servers packet" (CONS PROGRAM PROCEDURE)))
              (COND
                 (MESSAGE (printout PROMPTWINDOW .TAB0 0 "[Looking for " MESSAGE " on net")))
              [COND
                 ([COND
                     ((NOT NETHINT)
                      NIL)
                     ((FIXP NETHINT)                            ; If there's a hint about the net, try harder for that net
                      (SETQ RESULT (\COURIER.BROADCAST.ON.NET NETHINT SKT EPKT PROGRAM PGMDEF PROCEDURE PROCDEF
                                          RESULTFN MESSAGE 4)))
                     ((LISTP NETHINT)
                      (for NET in NETHINT thereis (SETQ RESULT
                                                        (\COURIER.BROADCAST.ON.NET NET SKT EPKT PROGRAM PGMDEF
                                                               PROCEDURE PROCDEF RESULTFN MESSAGE 4]
                                                        ; Found server on hinted net

                  )
                 (T (SETQ NEARBYNETS (CONS))
                    [\MAP.ROUTING.TABLE \NS.ROUTING.TABLE
                           (FUNCTION (LAMBDA (RT)               ; Gather up info about what nets are nearby in order of hop count
                                     (PROG ((HOPS (fetch (ROUTING RTHOPCOUNT) of RT)))
                                           (COND
                                              ((ILEQ HOPS 5)
                                               (for (TAIL _ NEARBYNETS)
                                                  while (AND (CDR TAIL)
                                                             (ILESSP (CAR (CADR TAIL))
                                                                     HOPS))
                                                  do (SETQ TAIL (CDR TAIL))
                                                  finally (push (CDR TAIL)
                                                                (LIST HOPS (fetch (ROUTING RTNET#) of RT]
                    [COND
                       ((OR (NULL (CDR NEARBYNETS))
                            (NEQ (CAR (CADR NEARBYNETS))
                                 0))                            ; Include local net
```

```
                        (push (CDR NEARBYNETS)
                              (LIST 0 0]
                    (COND
                       ([NOT (find PAIR in (CDR NEARBYNETS)
                                 suchthat (SETQ RESULT (\COURIER.BROADCAST.ON.NET (CADR PAIR)
                                                                                  SKT EPKT PROGRAM PGMDEF PROCEDURE PROCDEF RESULTFN
                                                                                  MESSAGE]
                        ;; Try once more, just in case we didn't wait long enough on the last guy.  The previous tries overlapped each other, and
                        ;; we need to wait a bit to give the last one equal time
                        (SETQ RESULT (\COURIER.BROADCAST.ON.NET (CADR (CADR NEARBYNETS))
                                                                SKT EPKT PROGRAM PGMDEF PROCEDURE PROCDEF RESULTFN]
             [COND
                (MESSAGE (printout PROMPTWINDOW %, (COND
                                                      (RESULT "done]")
                                                      (T "failed]"]
             (RETURN RESULT)))])
```

(\COURIER.BROADCAST.ON.NET

```
  [LAMBDA (NET NSOC XIP PROGRAM PGMDEF PROCEDURE PROCDEF RESULTFN MESSAGE %#TRIES)
                                                                                  ; Edited  2-Nov-93 13:51 by sybalsky:mv:envos
    (replace XIPDESTNET of XIP with NET)
    (COND
       (MESSAGE (printout PROMPTWINDOW %, .I0.8 NET ","")))
    (LET [(RESULT (NLSETQ (\SEND.EXPEDITED.XIP XIP NSOC PROGRAM PGMDEF PROCEDURE PROCDEF T (OR %#TRIES 2]
         (COND
            ((NOT RESULT)
             NIL)
            ((EQ (SETQ RESULT (CAR RESULT))
                 'USE.COURIER)
             NIL)
            (RESULTFN (CL:FUNCALL RESULTFN RESULT))
            (T RESULT])

)

(DEFINEQ
```

(COURIER.READ

```
  [LAMBDA (STREAM PROGRAM TYPE)                                   (* bvm%: "29-Oct-86 18:25")
    (LET (X)
         (COND
            [(LITATOM TYPE)
             (SELECTQ TYPE
                  (BOOLEAN (NEQ 0 (GETWORD STREAM)))
                  ((CARDINAL UNSPECIFIED)
                       (GETWORD STREAM))
                  (INTEGER (SIGNED (GETWORD STREAM)
                                   BITSPERWORD))
                  ((LONGCARDINAL LONGINTEGER)
                       (GETLONG STREAM))
                  (STRING (COURIER.READ.STRING STREAM))
                  (TIME (ALTO.TO.LISP.DATE (GETLONG STREAM)))
                  (COND
                     ((SETQ X (GETPROP TYPE 'COURIERDEF))         ; User-defined type
                      (CL:FUNCALL (CAR X)
                             STREAM PROGRAM TYPE))
                     ((SETQ X (\GET.COURIER.TYPE PROGRAM TYPE))
                      (COURIER.READ STREAM PROGRAM X))
                     (T (\COURIER.UNKNOWN.TYPE PROGRAM TYPE]
            [(AND (LISTP TYPE)
                  (LITATOM (CAR TYPE)))
             (SELECTQ (CAR TYPE)
                  (ENUMERATION (bind (ITEM _ (GETWORD STREAM)) for DEF in (CDR TYPE)
                                  do [COND
                                        ((IEQP ITEM (CADR DEF))
                                         (RETURN (CAR DEF]
                                  finally (RETURN ITEM)))
                  (ARRAY (bind (BASETYPE _ (CADDR TYPE)) to (CADR TYPE) collect (COURIER.READ STREAM PROGRAM
                                                                                               BASETYPE)))
                  (SEQUENCE                                       ; We ignore the maximum length of the sequence.
                            (COURIER.READ.SEQUENCE STREAM PROGRAM (OR (CADDR TYPE)
                                                                      (CADR TYPE))))
                  (RECORD (for NAMEANDTYPE in (CDR TYPE) collect (COURIER.READ STREAM PROGRAM (CADR NAMEANDTYPE))))
                  (NAMEDRECORD                                    ; Expanded form for backward compatibility
                               [for NAMEANDTYPE in (CDR TYPE) collect (LIST (CAR NAMEANDTYPE)
                                                                            (COURIER.READ STREAM PROGRAM
                                                                                  (CADR NAMEANDTYPE])
                  (CHOICE [bind (WHICH _ (GETWORD STREAM)) for DEF in (CDR TYPE)
                             do                                   ; DEF = (tag choice# type);  type = NIL is shorthand for type null
                                                                  ; record
                                [COND
                                   ((IEQP WHICH (CADR DEF))
                                    (RETURN (CONS (CAR DEF)
                                                  (AND (CADDR DEF)
                                                       (LIST (COURIER.READ STREAM PROGRAM (CADDR DEF]
```

```
                        finally (RETURN (LIST WHICH '???])
                (COND
                   ((LITATOM (CDR TYPE))                          ; Qualified name
                    (COURIER.READ STREAM (CAR TYPE)
                           (CDR TYPE)))
                   ((SETQ X (GETPROP (CAR TYPE)
                                     'COURIERDEF))
                    (CL:FUNCALL (CAR X)
                           STREAM PROGRAM TYPE))
                   (T (\COURIER.UNKNOWN.TYPE PROGRAM TYPE]
          (T (\COURIER.UNKNOWN.TYPE PROGRAM TYPE])
```

(\COURIER.UNKNOWN.TYPE

```
[LAMBDA (PROGRAM TYPE)                                            (* bvm%: "27-Jun-84 15:36")
  (ERROR "Unknown Courier Type" (COND
                                   (PROGRAM (CONS PROGRAM TYPE))
                                   (T TYPE])
```

(COURIER.READ.SEQUENCE

```
[LAMBDA (STREAM PROGRAM BASETYPE)                                 (* bvm%: "27-Jun-84 15:16")
```

;;; Reads a Courier SEQUENCE, returning it as a list of objects of type BASETYPE

```
    (to (GETWORD STREAM) collect (COURIER.READ STREAM PROGRAM BASETYPE])
```

(COURIER.READ.STRING

```
[LAMBDA (STREAM)                                                  ; Edited 10-Mar-89 14:59 by bvm
```

;; Read a string. First word is the length, then come that many bytes of an NS encoded string.

```
    (LET* ((LENGTH (GETWORD STREAM))
           (STRING (ALLOCSTRING LENGTH))
           (BASE (fetch (STRINGP BASE) of STRING))
           (OFFSET (fetch (STRINGP OFFST) of STRING)))
          (\BINS STREAM BASE OFFSET LENGTH)
          (COND
             ((ODDP LENGTH)
              (BIN STREAM)))
          (if (for I from OFFSET to (+ OFFSET LENGTH -1) thereis (EQ (\GETBASEBYTE BASE I)
                                                                     255))
              then                                                ; String had NS encoding, so have to read it more carefully
                   (DECODE-NS-STRING STRING)
            else STRING])
```

(COURIER.WRITE

```
[LAMBDA (STREAM ITEM PROGRAM TYPE)                                (* bvm%: "29-Oct-86 18:25")
  (PROG (X)
        (COND
           [(LITATOM TYPE)
            (SELECTQ TYPE
                 (BOOLEAN (PUTWORD STREAM (COND
                                             (ITEM 1)
                                             (T 0))))
                 ((CARDINAL UNSPECIFIED)
                      (PUTWORD STREAM ITEM))
                 (INTEGER (PUTWORD STREAM (UNSIGNED ITEM BITSPERWORD)))
                 ((LONGCARDINAL LONGINTEGER)
                      (PUTLONG STREAM ITEM))
                 (STRING (COURIER.WRITE.STRING STREAM ITEM))
                 (TIME (PUTLONG STREAM (LISP.TO.ALTO.DATE ITEM)))
                 (COND
                    ((SETQ X (GETPROP TYPE 'COURIERDEF))          ; User-defined type
                     (CL:FUNCALL (CADR X)
                                 STREAM ITEM PROGRAM TYPE))
                    ((SETQ X (\GET.COURIER.TYPE PROGRAM TYPE))
                     (COURIER.WRITE STREAM ITEM PROGRAM X))
                    (T (\COURIER.UNKNOWN.TYPE PROGRAM TYPE]
           [(AND (LISTP TYPE)
                 (LITATOM (CAR TYPE)))
            (SELECTQ (CAR TYPE)
                 (ENUMERATION                                     ; Keys can be either atoms, for fast lookup, or strings, to save
                                                                  ; atom space
                             [PUTWORD STREAM (OR [CADR (OR (ASSOC ITEM (CDR TYPE))
                                                           (find X in (CDR TYPE)
                                                              bind (KEY _ (MKSTRING ITEM))
                                                              suchthat (STREQUAL KEY (CAR X]
                                                 (\COURIER.TYPE.ERROR ITEM 'ENUMERATION])
                 (ARRAY (PROG ((SIZE (CADR TYPE))
                               (BASETYPE (CADDR TYPE)))
                              (COND
                                 ((NOT (IEQP SIZE (LENGTH ITEM)))
                                  (\COURIER.TYPE.ERROR ITEM TYPE)))
                              (for X in ITEM do (COURIER.WRITE STREAM X PROGRAM BASETYPE))))
                 (SEQUENCE                                        ; We ignore the maximum length of the sequence.
```

```
                        (COURIER.WRITE.SEQUENCE STREAM ITEM PROGRAM (OR (CADDR TYPE)
                                                                        (CADR TYPE))))
                (RECORD (for NAMEANDTYPE in (CDR TYPE) as VALUE in ITEM do (COURIER.WRITE STREAM VALUE PROGRAM
                                                                                (CADR NAMEANDTYPE))))
                (NAMEDRECORD                                            ; Old style
                             (for NAMEANDTYPE in (CDR TYPE) as NAMEANDVALUE in ITEM
                                do [COND
                                       ((NEQ (CAR NAMEANDTYPE)
                                             (CAR NAMEANDVALUE))
                                        (\COURIER.TYPE.ERROR ITEM (CAR TYPE]
                                      (COURIER.WRITE STREAM (CADR NAMEANDVALUE)
                                             PROGRAM
                                             (CADR NAMEANDTYPE))))
                (CHOICE [PROG [(WHICH (OR (ASSOC (CAR ITEM)
                                                 (CDR TYPE))
                                          (\COURIER.TYPE.ERROR ITEM 'CHOICE]
                              (PUTWORD STREAM (CADR WHICH))
                              (COND
                                 ((CADDR WHICH)
                                  (COURIER.WRITE STREAM (CADR ITEM)
                                         PROGRAM
                                         (CADDR WHICH])
                (COND
                   ((LITATOM (CDR TYPE))                                ; Qualified name
                    (COURIER.WRITE STREAM ITEM (CAR TYPE)
                           (CDR TYPE)))
                   ((SETQ X (GETPROP (CAR TYPE)
                                     'COURIERDEF))                      ; User-defined type
                    (CL:FUNCALL (CADR X)
                           STREAM ITEM PROGRAM TYPE))
                   (T (\COURIER.UNKNOWN.TYPE PROGRAM TYPE]
          (T (\COURIER.UNKNOWN.TYPE PROGRAM TYPE])
```

(COURIER.WRITE.SEQUENCE

```
[LAMBDA (STREAM ITEMLIST PROGRAM TYPE)                                  (* bvm%: " 4-Jul-84 15:13")
  (PROG ((BASETYPE TYPE))
        (COND
           [(EQ (CAR (LISTP ITEMLIST))
                'INTERPRETATION)
```

;; This is how to write a (SEQUENCE UNSPECIFIED) without running it through COURIER.WRITE.REP first. ITEMLIST = ;; (INTERPRETATION type value)

```
            (COND
               ((NEQ BASETYPE 'UNSPECIFIED)
                (\COURIER.TYPE.ERROR ITEMLIST TYPE))
               (T (SETQ BASETYPE (CADR ITEMLIST))
                  (COURIER.WRITE.SEQUENCE.UNSPECIFIED STREAM (CADDR ITEMLIST)
                         (COND
                            [(LISTP BASETYPE)
                             (PROG1 (CAR BASETYPE)
                                 (SETQ BASETYPE (CDR BASETYPE)))]
                            (T PROGRAM))
                         BASETYPE]
           ((NULL ITEMLIST)
            (PUTWORD STREAM 0))
           ((LISTP ITEMLIST)
            (PUTWORD STREAM (LENGTH ITEMLIST))
            (for X in ITEMLIST do (COURIER.WRITE STREAM X PROGRAM BASETYPE)))
           (T (\COURIER.TYPE.ERROR ITEMLIST TYPE])
```

(COURIER.WRITE.STRING

```
[LAMBDA (STREAM STRING)                                                 ; Edited 21-Jul-87 14:36 by bvm:
  (if [fetch (STRINGP FATSTRINGP) of (OR (STRINGP STRING)
                                         (SETQ STRING (MKSTRING STRING]
      then                                                              ; Have to produce NS encoding
           (COURIER.WRITE.FAT.STRING STREAM STRING)
    else (LET ((LENGTH (NCHARS STRING)))
              (PUTWORD STREAM LENGTH)
              (\BOUTS STREAM (fetch (STRINGP BASE) of STRING)
                     (fetch (STRINGP OFFST) of STRING)
                     LENGTH)
              (COND
                 ((ODDP LENGTH)
                  (BOUT STREAM 0])
```

(COURIER.WRITE.FAT.STRING

```
[LAMBDA (STREAM STRING UNSPECIFIED)                                     ; Edited 21-Jul-87 15:24 by bvm:
```

;; Write the fat string STRING to courier STREAM. If UNSPECIFIED is true, encode it as a sequence unspecified, else as a string.

```
  (LET ((CORE (OPENSTREAM '{NODIRCORE} 'BOTH))
        LENGTH)
       (PRIN3 STRING CORE)                                              ; Write out string to get encoding and length, then copy the bytes
       (SETQ LENGTH (GETFILEPTR CORE))
       [if UNSPECIFIED
```

```
            then                                                    ; writing sequence unspecified, so include length of sequence
                 (PUTWORD STREAM (ADD1 (FOLDHI LENGTH BYTESPERWORD]
          (PUTWORD STREAM LENGTH)
          (COPYBYTES CORE STREAM 0 LENGTH)
          (COND
             ((ODDP LENGTH)
              (BOUT STREAM 0])
```

(COURIER.SKIP

```
  [LAMBDA (STREAM PROGRAM TYPE)                                     ; Edited 30-Jun-87 17:40 by bvm:
    (LET (X)
         (COND
            [(LITATOM TYPE)
             (SELECTQ TYPE
                  ((BOOLEAN CARDINAL UNSPECIFIED INTEGER)           ; 2 bytes
                       (\BIN STREAM)
                       (\BIN STREAM))
                  ((LONGCARDINAL LONGINTEGER TIME)                  ; 4 bytes
                       (\BIN STREAM)
                       (\BIN STREAM)
                       (\BIN STREAM)
                       (\BIN STREAM))
                  (STRING                                           ; Count followed by number of bytes, padded to even byte
                          (RPTQ (CEIL (GETWORD STREAM)
                                      BYTESPERWORD)
                                (\BIN STREAM)))
                  (COND
                     ((SETQ X (GETPROP TYPE 'COURIERDEF))           ; User-defined type
                      (CL:FUNCALL (CAR X)
                             STREAM PROGRAM TYPE))
                     ((SETQ X (\GET.COURIER.TYPE PROGRAM TYPE))
                      (COURIER.SKIP STREAM PROGRAM X))
                     (T (\COURIER.UNKNOWN.TYPE PROGRAM TYPE]
            [(AND (LISTP TYPE)
                  (LITATOM (CAR TYPE)))
             (SELECTQ (CAR TYPE)
                  (ENUMERATION                                      ; 2 bytes
                              (\BIN STREAM)
                              (\BIN STREAM))
                  (ARRAY (bind (BASETYPE _ (CADDR TYPE)) to (CADR TYPE) DO (COURIER.SKIP STREAM PROGRAM BASETYPE)
                              ))
                  (SEQUENCE                                         ; We ignore the maximum length of the sequence.
                            (COURIER.SKIP.SEQUENCE STREAM PROGRAM (OR (CADDR TYPE)
                                                                      (CADR TYPE))))
                  ((RECORD NAMEDRECORD)
                       (for NAMEANDTYPE in (CDR TYPE) DO (COURIER.SKIP STREAM PROGRAM (CADR NAMEANDTYPE))))
                  (CHOICE [bind (WHICH _ (GETWORD STREAM)) for DEF in (CDR TYPE)
                             do                                     ; DEF = (tag choice# type);  type = NIL is shorthand for type null
                                                                    ; record
                                (COND
                                   ((IEQP WHICH (CADR DEF))
                                    (RETURN (AND (CADDR DEF)
                                                 (COURIER.SKIP STREAM PROGRAM (CADDR DEF])
                  (COND
                     ((LITATOM (CDR TYPE))                          ; Qualified name
                      (COURIER.SKIP STREAM (CAR TYPE)
                              (CDR TYPE)))
                     ((SETQ X (GETPROP (CAR TYPE)
                                       'COURIERDEF))
                      (CL:FUNCALL (CAR X)
                             STREAM PROGRAM TYPE))
                     (T (\COURIER.UNKNOWN.TYPE PROGRAM TYPE]
            (T (\COURIER.UNKNOWN.TYPE PROGRAM TYPE])
```

(COURIER.SKIP.SEQUENCE

```
  [LAMBDA (STREAM PROGRAM BASETYPE)                                 ; Edited 30-Jun-87 17:40 by bvm:
```

;;; Reads a Courier SEQUENCE, returning it as a list of objects of type BASETYPE

```
    (to (GETWORD STREAM) do (COURIER.SKIP STREAM PROGRAM BASETYPE])
```

(\COURIER.TYPE.ERROR

```
  [LAMBDA (ITEM TYPE)                                               (* bvm%: " 3-Jul-84 17:53")
    (ERROR (CONCAT "Arg not of Courier type " TYPE)
           ITEM])
```

(DECODE-NS-STRING

```
  [LAMBDA (STR)                                                     ; Edited 10-Mar-89 14:50 by bvm
```

;; STR is a string read from an 8-bit stream but that might have NS run coding in it. We return the string that results from interpreting the runcoding

```
    (LET*
     ((LENGTH (NCHARS STR))
      (BASE (fetch (STRINGP BASE) of STR))
```

```
      (OFFSET (fetch (STRINGP OFFST) of STR))
      (LASTOFFSET (+ OFFSET LENGTH))
      (FATLENGTH 0))
     (bind (I _ OFFSET)
           (BYTEINC _ 1) while (< I LASTOFFSET)
        do                                                    ; Count how many chars will be in output
           (if (EQ (\GETBASEBYTE BASE I)
                   255)
               then (SETQ BYTEINC (if (AND (< (add I 1)
                                              LASTOFFSET)
                                           (EQ (\GETBASEBYTE BASE I)
                                               255))
                                      then                    ; 255-255-0 means 2 bytes per char
                                           2
                                    else                      ; 255-x means shift to charset x
                                         1))
             else                                             ; Ordinary character
                  (add FATLENGTH 1))
           ;; Bump I past the number of bytes consumed on this iteration.  Note that in the case of 255 we bumped i once already, and we now
           ;; bump it 1 more ordinarily, or 2 if the sequence was 255-255-0
           (add I BYTEINC))
     (if (< FATLENGTH LENGTH)
         then                                                 ; If length is the same, then there must not have been any fat
                                                              ; chars
          (LET* ((FATSTR (ALLOCSTRING FATLENGTH NIL NIL T))
                 (FATBASE (\ADDBASE (fetch (STRINGP BASE) of FATSTR)
                                    (fetch (STRINGP OFFST) of FATSTR)))
                 (I OFFSET)
                 (CSET 0)
                 CH)
                (while (< I LASTOFFSET)
                   do (if (EQ (SETQ CH (\GETBASEBYTE BASE I))
                              255)
                          then                                ; Switch char sets or runcoding
                               [if (< (add I 1)
                                      LASTOFFSET)
                                   then                       ; Check is for naked 255 at end--bug, but we'll ignore it
                                        (SETQ CSET (if (EQ (SETQ CSET (\GETBASEBYTE BASE I))
                                                           255)
                                                       then   ; Stop runcoding.  Ignore next byte (should be zero; if not, we
                                                              ; haven't the foggiest)
                                                            (add I 1)
                                                            T
                                                     else (LLSH CSET 8]
                                 else (\PUTBASE FATBASE 0 (if (EQ CSET T)
                                                              then (+ (LLSH CH 8)
                                                                      (if (< (add I 1)
                                                                             LASTOFFSET)
                                                                          then (\GETBASEBYTE BASE I)
                                                                       else
                                                                       ; ack, eof.  Don't attempt a possibly illegal fetch
                                                                            0))
                                                            else (+ CSET CH)))
                                      (SETQ FATBASE (\ADDBASE FATBASE 1)))
                          (add I 1))
                   FATSTR)
       else STR])

)

(DEFINEQ
```

(COURIER.READ.BULKDATA

```
  [LAMBDA (STREAM PROGRAM TYPE DONTCLOSE)                     (* bvm%: "13-Feb-85 23:42")
```

;;; Read a Bulk Data object which is a stream of the specified type. This can be done by declaring the stream type in Courier, as is done in the protocol
;;; specs, but that causes COURIER.READ to produce a deeply nested structure. Instead, this function returns a list of objects making up the stream.
;;; See the Bulk Data Transfer spec.

```
    ;; Closes STREAM on exit unless DONTCLOSE is true.  If STREAM is not a stream, returns it directly, presumably an error from COURIER.CALL
    (COND
       [(type? STREAM STREAM)
        (PROG1 (bind LASTSEGMENT? join (PROGN (SETQ LASTSEGMENT? (NEQ (GETWORD STREAM)
                                                                      0))
                                              (COURIER.READ.SEQUENCE STREAM PROGRAM TYPE))
                     repeatuntil LASTSEGMENT?)
               (OR DONTCLOSE (CLOSEF STREAM)))]
        (T                                                    ; An error return from COURIER.CALL -- pass it thru
           STREAM])
```

(BULKDATA.GENERATOR

```
  [LAMBDA (STREAM PROGRAM TYPE)                               (* bvm%: "19-Jul-84 11:40")
```

;; Produces a generator for reading from STREAM a Courier 'Stream of PROGRAM.TYPE' . The value returned from this function is an object to
;; pass to BULKDATA.GENERATE.NEXT to retrieve the next item from the stream.

```
    (create BULKDATAGENERATOR
           BGSTREAM _ STREAM
           BGPROGRAM _ PROGRAM
           BGTYPE _ TYPE
           BGLASTSEGMENT? _ NIL
           BGITEMSLEFT _ 0])
```

(BULKDATA.GENERATE.NEXT

```
  [LAMBDA (GENSTATE)                                          (* bvm%: "19-Jul-84 11:34")
```

;; Returns the next item from bulkdata generator GENSTATE, updating the state. Returns NIL when generator exhausted

```
    (PROG ((STREAM (fetch BGSTREAM of GENSTATE))
           (CNT (fetch BGITEMSLEFT of GENSTATE)))
      LP   (COND
              ((NEQ CNT 0)                                    ; Middle of a segment
               (replace BGITEMSLEFT of GENSTATE with (SUB1 CNT)))
              ((fetch BGLASTSEGMENT? of GENSTATE)             ; Finished last segment
               (RETURN NIL))
              (T                                              ; Finished a segment, get the next
                 (COND
                    ((NEQ (GETWORD STREAM)
                          0)
                     (replace BGLASTSEGMENT? of GENSTATE with T)))
                 (SETQ CNT (GETWORD STREAM))
                 (GO LP)))
           (RETURN (COURIER.READ STREAM (fetch BGPROGRAM of GENSTATE)
                                (fetch BGTYPE of GENSTATE])
```

(COURIER.WRITE.BULKDATA

```
  [LAMBDA (STREAM ITEMLIST PROGRAM TYPE)                      (* bvm%: " 4-Jul-84 15:24")
```

;;; Writes ITEMLIST as a Bulk Data object which is a stream of the specified type, i.e., ITEMLIST is interpreted as a list of (PROGRAM . TYPE) objects.
;;; Returns NIL

;; Format a little strange: a succession of SEQUENCE's, the last of which is flagged as the final sequence. In theory, one could send the entire list,
;; up to 65535 items, as a single sequence, but maybe that overloads some processors, so break it up into smaller chunks

```
    (PROG ((LEN (LENGTH ITEMLIST))
           (TAIL ITEMLIST)
           SEGMENTLENGTH)
          (do (PUTWORD STREAM (COND
                                 ((IGREATERP LEN 100)          ; Don't try to write too long segments
                                  (SETQ SEGMENTLENGTH 100)     ; Not last segment
                                  0)
                                 (T (SETQ SEGMENTLENGTH LEN)
                                    1)))
              (PUTWORD STREAM SEGMENTLENGTH)
              (to SEGMENTLENGTH do (COURIER.WRITE STREAM (pop TAIL)
                                                  PROGRAM TYPE))
              (SETQ LEN (IDIFFERENCE LEN SEGMENTLENGTH)) repeatwhile TAIL])
```

(COURIER.ABORT.BULKDATA

```
  [LAMBDA (ERROR)                                             ; Edited 27-Aug-87 11:18 by bvm:
```

;; Called from within a bulkdata handler to abort the bulk data operation. The corresponding CATCH is in \COURIER.HANDLE.BULKDATA.
;; Optional ERROR should be returned from the courier call, instead of what the procedure returns (typically (error transfer.error Aborted)).

```
    (COND
       (ERROR (CL:THROW :BULKDATA (CL:VALUES :ABORT ERROR)))
       (T (CL:THROW :BULKDATA :ABORT])

)

;; Reading/writing sequence unspecified in an interesting way

(DEFINEQ
```

(COURIER.READ.REP

```
  [LAMBDA (LIST.OF.WORDS PROGRAM TEMPLATE)                    (* bvm%: "14-Jun-84 15:08")
```

;; Like COURIER.READ but 'reads' from a list of integers corresponding to the words in the Courier representation.

```
    (COURIER.READ (\MAKE.COURIER.REP.STREAM LIST.OF.WORDS)
           PROGRAM TEMPLATE])
```

(COURIER.WRITE.REP

```
  [LAMBDA (VALUE PROGRAM TYPE)                                (* bvm%: "14-Jun-84 16:15")
    (PROG ((STREAM (\MAKE.COURIER.REP.STREAM)))
          (COURIER.WRITE STREAM VALUE PROGRAM TYPE)
          (COND
             ((fetch CRNEXTBYTE of STREAM)
              (\BOUT STREAM 0)))
          (RETURN (fetch CRWORDLIST of STREAM])
```

(COURIER.WRITE.SEQUENCE.UNSPECIFIED

```
  [LAMBDA (STREAM ITEM PROGRAM TYPE)                                  ; Edited 21-Jul-87 14:27 by bvm:

;;; Write ITEM on STREAM as a (SEQUENCE UNSPECIFIED) interpreted as a (PROGRAM . TYPE); this means figuring out how long ITEM is so we
;;; can write the appropriate word count before sending ITEM

    (PROG (X FN)
          (COND
             [(LITATOM TYPE)
              (SELECTQ TYPE
                   (BOOLEAN (PUTWORD STREAM 1)
                            (PUTWORD STREAM (COND
                                               (ITEM 1)
                                               (T 0))))
                   ((CARDINAL UNSPECIFIED)
                        (PUTWORD STREAM 1)
                        (PUTWORD STREAM ITEM))
                   (INTEGER (PUTWORD STREAM 1)
                            (PUTWORD STREAM (UNSIGNED ITEM BITSPERWORD)))
                   ((LONGCARDINAL LONGINTEGER)
                        (PUTWORD STREAM 2)
                        (PUTLONG STREAM ITEM))
                   (STRING (if [fetch (STRINGP FATSTRINGP) of (OR (STRINGP ITEM)
                                                                  (SETQ ITEM (MKSTRING ITEM]
                               then                                    ; Have to produce NS encoding
                                    (COURIER.WRITE.FAT.STRING STREAM ITEM T)
                               else (PUTWORD STREAM (ADD1 (FOLDHI (NCHARS ITEM)
                                                                  BYTESPERWORD)))
                                    (COURIER.WRITE.STRING STREAM ITEM)))
                   (TIME (PUTWORD STREAM 2)
                         (PUTLONG STREAM (LISP.TO.ALTO.DATE ITEM)))
                   (COND
                      ((SETQ X (GETPROP TYPE 'COURIERDEF))             ; User-defined type
                       (GO USERTYPE))
                      ((SETQ X (\GET.COURIER.TYPE PROGRAM TYPE))
                       (COURIER.WRITE.SEQUENCE.UNSPECIFIED STREAM ITEM PROGRAM X))
                      (T (\CWSU.DEFAULT STREAM ITEM PROGRAM TYPE]
             [(AND (LISTP TYPE)
                   (LITATOM (CAR TYPE)))
              (SELECTQ (CAR TYPE)
                   (ENUMERATION (PUTWORD STREAM 1)
                                (COURIER.WRITE STREAM ITEM PROGRAM TYPE))
                   ((ARRAY SEQUENCE RECORD NAMEDRECORD CHOICE)
                       [PROG ((LENGTH (COURIER.REP.LENGTH ITEM PROGRAM TYPE)))
                             (COND
                                (LENGTH (PUTWORD STREAM LENGTH)
                                        (COURIER.WRITE STREAM ITEM PROGRAM TYPE))
                                (T (\CWSU.DEFAULT STREAM ITEM PROGRAM TYPE])
                   (COND
                      ((LITATOM (CDR TYPE))                            ; Qualified name
                       (COURIER.WRITE.SEQUENCE.UNSPECIFIED STREAM ITEM (CAR TYPE)
                              (CDR TYPE)))
                      ((SETQ X (GETPROP (CAR TYPE)
                                        'COURIERDEF))                  ; User-defined type
                       (GO USERTYPE))
                      (T (\CWSU.DEFAULT STREAM ITEM PROGRAM TYPE]
             (T (\COURIER.UNKNOWN.TYPE PROGRAM TYPE)))
          (RETURN)
      USERTYPE
                                                                       ; X = (readFn writeFn lengthFn writeSequenceFn)
          (COND
             ((SETQ FN (CADDDR X))
              (CL:FUNCALL FN STREAM ITEM PROGRAM TYPE))
             ([AND (SETQ FN (CADDR X))
                   (OR (FIXP FN)
                       (SETQ FN (CL:FUNCALL FN ITEM PROGRAM TYPE]
                                                                       ; Says how long it is
              (PUTWORD STREAM FN)
              (CL:FUNCALL (CADR X)
                     STREAM ITEM PROGRAM TYPE))
             (T (\CWSU.DEFAULT STREAM ITEM PROGRAM TYPE])
```

(\CWSU.DEFAULT

```
  [LAMBDA (STREAM ITEM PROGRAM TYPE)                                  (* bvm%: " 1-Jul-84 18:05")
    (COURIER.WRITE STREAM (COURIER.WRITE.REP ITEM PROGRAM TYPE)
           NIL
           '(SEQUENCE UNSPECIFIED])
```

(COURIER.REP.LENGTH

```
  [LAMBDA (ITEM PROGRAM TYPE)                                         ; Edited 21-Jul-87 14:35 by bvm:

;;; Returns the number of words that the Courier rep of ITEM as a (PROGRAM . TYPE) would occupy or NIL if we can't easily figure it out

    (LET (X)
         (COND
            [(LITATOM TYPE)
```

```
            (SELECTQ TYPE
                 ((BOOLEAN CARDINAL INTEGER UNSPECIFIED)
                     1)
                 ((LONGCARDINAL LONGINTEGER TIME)
                     2)
                 (STRING (if [NOT (fetch (STRINGP FATSTRINGP) of (OR (STRINGP ITEM)
                                                                     (SETQ ITEM (MKSTRING ITEM]
                             then                                    ; Too hard to figure out fat length
                                  (ADD1 (FOLDHI (NCHARS ITEM)
                                                BYTESPERWORD))))
                 (COND
                    [(SETQ X (GETPROP TYPE 'COURIERDEF))           ; User-defined type
                     (AND (SETQ X (CADDR X))
                          (OR (FIXP X)
                              (CL:FUNCALL X ITEM PROGRAM TYPE]
                    ((SETQ X (\GET.COURIER.TYPE PROGRAM TYPE))
                     (COURIER.REP.LENGTH ITEM PROGRAM X]
           [(AND (LISTP TYPE)
                 (LITATOM (CAR TYPE)))
            (SELECTQ (CAR TYPE)
                 (ENUMERATION 1)
                 (ARRAY (for X in ITEM bind (BASETYPE _ (CADDR TYPE)) sum (OR (COURIER.REP.LENGTH X PROGRAM
                                                                                  BASETYPE)
                                                                              (RETURN))))
                 (SEQUENCE (for X in ITEM bind (BASETYPE _ (OR (CADDR TYPE)
                                                               (CADR TYPE)))
                              sum (OR (COURIER.REP.LENGTH X PROGRAM BASETYPE)
                                      (RETURN))
                              finally                              ; Count the word which is the sequence length
                                    (RETURN (ADD1 $$VAL))))
                 (RECORD (for NAMEANDTYPE in (CDR TYPE) as VALUE in ITEM sum (OR (COURIER.REP.LENGTH
                                                                                    VALUE PROGRAM (CADR NAMEANDTYPE
                                                                                                  ))
                                                                                 (RETURN))))
                 (NAMEDRECORD (for NAMEANDTYPE in (CDR TYPE) as NAMEANDVALUE in ITEM
                                 sum (OR (COURIER.REP.LENGTH (CADR NAMEANDVALUE)
                                                             PROGRAM
                                                             (CADR NAMEANDTYPE))
                                         (RETURN))))
                 (CHOICE (LET* [[WHICH (OR (ASSOC (CAR ITEM)
                                                  (CDR TYPE))
                                           (\COURIER.TYPE.ERROR ITEM 'CHOICE]
                               (N (COND
                                      ((CADDR WHICH)
                                       (COURIER.REP.LENGTH (CADR ITEM)
                                                           PROGRAM
                                                           (CADDR WHICH)))
                                      (T 0]
                                  (AND N (ADD1 N))))
                 (COND
                    ((LITATOM (CDR TYPE))                          ; Qualified name
                     (COURIER.REP.LENGTH ITEM (CAR TYPE)
                             (CDR TYPE)))
                    ((SETQ X (GETPROP (CAR TYPE)
                                      'COURIERDEF))                ; User-defined type
                     (AND (SETQ X (CADDR X))
                          (OR (FIXP X)
                              (CL:FUNCALL X ITEM PROGRAM TYPE]
           (T (\COURIER.UNKNOWN.TYPE PROGRAM TYPE])
```

(\MAKE.COURIER.REP.STREAM

```
  [LAMBDA (LIST.OF.WORDS)                                          (* bvm%: "15-Jun-84 11:54")
```

;;; Makes a STREAM whose BIN operation produces bytes from LIST.OF.WORDS or whose BOUT operation produces a list of words in the stream's
;;; CRWORDLIST field (can only use stream for one or the other, of course)

```
    (PROG [(STREAM (NCREATE 'STREAM]
          (replace DEVICE of STREAM with (OR \COURIER.REP.DEVICE (PROGN (SETQ \COURIER.REP.DEVICE
                                                                              (NCREATE 'FDEV))
                                                                        (replace BLOCKIN of \COURIER.REP.DEVICE
                                                                           with (FUNCTION \NONPAGEDBINS))
                                                                        (replace BLOCKOUT of \COURIER.REP.DEVICE
                                                                           with (FUNCTION \NONPAGEDBOUTS))
                                                                        \COURIER.REP.DEVICE)))
          (replace ACCESSBITS of STREAM with BothBits)
          (replace STRMBINFN of STREAM with (FUNCTION \COURIER.REP.BIN))
          (replace STRMBOUTFN of STREAM with (FUNCTION \COURIER.REP.BOUT))
          (replace ENDOFSTREAMOP of STREAM with (FUNCTION \COURIER.EOF))
          (replace CRWORDLIST of STREAM with LIST.OF.WORDS)
          (RETURN STREAM])
```

(\COURIER.REP.BIN

```
  [LAMBDA (STREAM)                                                 (* bvm%: "14-Jun-84 16:06")
    (PROG ((X (fetch CRNEXTBYTE of STREAM)))
          (RETURN (COND
```

```
                    (X (replace CRNEXTBYTE of STREAM with NIL)
                       X)
                    (T (SETQ X (OR (pop (fetch CRWORDLIST of STREAM))
                                   (ERROR "Courier stream prematurely terminated")))
                       (replace CRNEXTBYTE of STREAM with (fetch LOBYTE of X))
                       (fetch HIBYTE of X])
```

(\COURIER.REP.BOUT

```
  [LAMBDA (STREAM BYTE)                                    (* bvm%: "14-Jun-84 16:13")
    (PROG ((X (fetch CRNEXTBYTE of STREAM))
           TAIL)
          (COND
             (X (SETQ X (create WORD
                                HIBYTE _ X
                                LOBYTE _ BYTE))
                [replace CRLASTWORD of STREAM with (COND
                                                     [(SETQ TAIL (fetch CRLASTWORD of STREAM))
                                                      (CDR (RPLACD TAIL (CONS X]
                                                     (T (replace CRWORDLIST of STREAM with (LIST X]
                (replace CRNEXTBYTE of STREAM with NIL))
             (T (replace CRNEXTBYTE of STREAM with BYTE])

)

(RPAQ? \COURIER.REP.DEVICE NIL)

(DEFINEQ
```

(COURIER.READ.NSADDRESS

```
  [LAMBDA (STREAM)                                         (* bvm%: "12-Jun-84 11:41")
    ;; Read a standard NSADDRESS from the next 6 words of STREAM
    (LET ((ADDR (create NSADDRESS)))
         (\BINS STREAM ADDR 0 (UNFOLD \#WDS.NSADDRESS BYTESPERWORD))
         ADDR])
```

(COURIER.WRITE.NSADDRESS

```
  [LAMBDA (STREAM ADDR)                                    (* bvm%: "12-Jun-84 11:45")
    (\BOUTS STREAM (\DTEST ADDR 'NSADDRESS)
           0
           (UNFOLD \#WDS.NSADDRESS BYTESPERWORD])

)

(PUTPROPS NSADDRESS COURIERDEF (COURIER.READ.NSADDRESS COURIER.WRITE.NSADDRESS 6))
```

;; Debugging

```
(RPAQ? COURIERTRACEFILE )

(RPAQ? COURIERTRACEFLG )

(RPAQ? COURIERPRINTLEVEL '(2 . 4))

(RPAQ? NSWIZARDFLG )

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS COURIERTRACEFLG COURIERTRACEFILE COURIERPRINTLEVEL NSWIZARDFLG)
)

(DEFINEQ
```

(COURIERTRACE

```
  [LAMBDA (FLG REGION)                                     ; Edited  1-May-87 11:22 by bvm:
    (if (NULL FLG)
        then (if (ACTIVEWP COURIERTRACEFILE)
                 then (CLOSEW COURIERTRACEFILE))
             (SETQ COURIERTRACEFILE T)
             (SETQ COURIERTRACEFLG NIL)
      else (if (NOT (ACTIVEWP COURIERTRACEFILE))
               then (SETQ COURIERTRACEFILE (CREATEW REGION "Courier Trace Window")))
           [WINDOWPROP COURIERTRACEFILE 'BUTTONEVENTFN (FUNCTION (LAMBDA (WINDOW)
                                                                   (if (LASTMOUSESTATE (NOT UP))
                                                                       then (\CHANGE.ETHER.TRACING WINDOW
                                                                                  'COURIERTRACEFLG]
           [WINDOWPROP COURIERTRACEFILE 'CLOSEFN (FUNCTION (LAMBDA (WINDOW)
                                                             (if (EQ WINDOW COURIERTRACEFILE)
                                                                 then (SETQ COURIERTRACEFLG NIL)
                                                                      (SETQ COURIERTRACEFILE T]
           [WINDOWPROP COURIERTRACEFILE 'SHRINKFN (FUNCTION (LAMBDA (WINDOW)
                                                             (if (EQ WINDOW COURIERTRACEFILE)
                                                                 then
                                                                 ; Turn off tracing while window shrunk
                                                                      (WINDOWPROP WINDOW 'COURIERTRACEFLG
```

```
                                                                    COURIERTRACEFLG)
                                                                (SETQ COURIERTRACEFLG NIL]
          [WINDOWPROP COURIERTRACEFILE 'EXPANDFN (FUNCTION (LAMBDA (WINDOW)
                                                             (if (EQ WINDOW COURIERTRACEFILE)
                                                                 then
                                                                 ; Restore tracing to previous state
                                                                       (SETQ COURIERTRACEFLG (WINDOWPROP
                                                                                              WINDOW
                                                                                              'COURIERTRACEFLG
                                                                                              NIL]
          (DSPFONT (FONTCREATE 'GACHA 8)
                 COURIERTRACEFILE)
          (SETQ COURIERTRACEFLG FLG)
          (DSPSCROLL T COURIERTRACEFILE)
          (TOTOPW COURIERTRACEFILE)
          T])
```

(\COURIER.TRACE

```
  [LAMBDA (EVENT PROGRAM PROCEDURE ARGUMENTS)                    (* bvm%: "22-Jun-84 17:16")
    (SELECTQ EVENT
         (CALL (printout COURIERTRACEFILE .TAB0 0 PROGRAM "." PROCEDURE "[")
               [COND
                  (ARGUMENTS (COND
                                 ((EQ COURIERTRACEFLG 'PEEK)
                                  (printout COURIERTRACEFILE '--))
                                 (T (for X in ARGUMENTS bind (FIRSTTIME _ T) do (COND
                                                                                   (FIRSTTIME (SETQ FIRSTTIME NIL))
                                                                                   (T (SPACES 1 COURIERTRACEFILE)))
                                                                                LVLPRIN2 X COURIERTRACEFILE
                                                                                       (CAR COURIERPRINTLEVEL)
                                                                                       (CDR COURIERPRINTLEVEL]
               (printout COURIERTRACEFILE '%]))
         (RETURN (printout COURIERTRACEFILE " => ")
                 [COND
                    [(EQ COURIERTRACEFLG 'PEEK)
                     (printout COURIERTRACEFILE (COND
                                                    ((CDR (LISTP ARGUMENTS))
                                                     '--)
                                                    (T "&"]
                    (T (LVLPRINT ARGUMENTS COURIERTRACEFILE (CAR COURIERPRINTLEVEL)
                                 (CDR COURIERPRINTLEVEL])
         (BEGIN.BULK.DATA
              (printout COURIERTRACEFILE (COND
                                             ((EQ COURIERTRACEFLG 'PEEK)
                                              '{)
                                             (T "{bulk data"))))
         (END.BULK.DATA
              (printout COURIERTRACEFILE '}))
         (SHOULDNT])

)

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

(ADDTOVAR NLAMA \DUMP.COURIERPROGRAMS COURIERPROGRAM)

(ADDTOVAR NLAML )

(ADDTOVAR LAMA COURIER.EXPEDITED.CALL COURIER.CALL)
)

(PUTPROPS COURIER COPYRIGHT ("Venue & Xerox Corporation" 1983 1984 1985 1986 1987 1988 1989 1990 1991 1992
                                    1993))
```

FUNCTION INDEX

CONSTANT INDEX

VARIABLE INDEX

RECORD INDEX

MACRO INDEX

PROPERTY INDEX