File created: 19-Jan-93 11:20:29 **{DSK}<python>lde>lispcore>sources>SPP.;3**

previous date: 5-Jan-93 02:24:51 {DSK}<python>lde>lispcore>sources>SPP.;2

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;


```
(RPAQQ SPPCOMS
       ((COMS                                                    ; Sequenced Packet Protocol.
              (DECLARE%: EVAL@COMPILE DONTCOPY (FILES (SOURCE)
                                                      SPPDECLS)
                     (MACROS RETRANSMITINDEX SEQ.ADD1 SEQ.GREATERP SEQ.GEQ)
                     (GLOBALVARS SPP.USER.TIMEOUT SPP.MIN.TIMEOUT SPP.INACTIVITY.TIMEOUT SPP.MAX.FAILED.PROBES
                            XIPTRACEFLG XIPTRACEFILE))
              (SYSRECORDS SPPCON)
              (INITRECORDS SPPCON)
              (INITVARS (SPP.USER.TIMEOUT 15000)
                     (SPP.INACTIVITY.TIMEOUT 120000)
                     (SPP.MIN.TIMEOUT 50)
                     (SPP.MAX.FAILED.PROBES 5))
              (FNS \SPPCONNECTION \SPP.CREATE.CON \SPP.CREATE.STREAMS \SPP.CREATE.WATCHER \SPP.SENDPKT
                   \FILLINSPP \SPP.SYSPKT \GETSPP \SENDSPP \SPP.SEND.ENDREPLY \TERMINATESPP \SPP.CLEANUP)
              (FNS \SPPWATCHER \SPP.HANDLE.INPUT \SPP.HANDLE.DATA \SPP.HANDLE.ATTN \SPP.RELEASE.ACKED.PACKETS
                   \SPP.NOT.RESPONDING \SPP.PROBE \SPP.RETRANSMIT.NEXT \SPP.DUPLICATE.REQUEST \SPP.ESTABLISH
                   \SPPGETERROR \SPPSENDERROR))
        [COMS                                                    ; Stream interface to Sequenced Packet Protocol.
              (FNS \INITSPP \SPP.EVENTFN \CREATE.SPP.DEVICE SPP.OPEN \SPP.CREATE.STREAM SPP.DESTADDRESS
                   SPPOUTPUTSTREAM SPP.OPENP \STREAM.FROM.PACKET SPP.FORCEOUTPUT SPP.FLUSH.TO.EOF SPP.SENDEOM
                   SPP.CLEAREOM SPP.SENDATTENTION SPP.CLEARATTENTION SPP.CLOSE \SPP.CLOSE.IF.ERROR
                   \SPP.RESETCLOSE SPP.BACKFILEPTR \SPP.GETFILEPTR \SPP.SETFILEPTR \SPP.SKIPBYTES \SPP.BOUTS
                   \SPP.OTHER.BOUT \SPP.GETNEXTBUFFER \SPP.STREAM.LOST \SPP.DEFAULT.ERRORHANDLER
                   \SPP.PREPARE.INPUT \SPP.PREPARE.OUTPUT SPP.DSTYPE SPP.READP SPP.EOFP)
              (FNS SPPSTREAMP)
              (DECLARE%: DONTEVAL@LOAD DOCOPY (P (\INITSPP]
        (COMS                                                    ; Debugging
              (ALISTS (XIPPRINTMACROS 5))
              (FNS PPSPP \SPP.INFO.HOOK PPSPPSTREAM \SPP.CHECK.INPUT.QUEUE PRINTSPP)
              (INITVARS (PRINTSPPDATAFLG))
              (GLOBALVARS PRINTSPPDATAFLG))))
```

;; Sequenced Packet Protocol.

```
(DECLARE%: EVAL@COMPILE DONTCOPY

(FILESLOAD (SOURCE)
       SPPDECLS)

(DECLARE%: EVAL@COMPILE

(PUTPROPS RETRANSMITINDEX MACRO ((SEQNO)
                                 (IMOD SEQNO \SPP.RETRANSMITQ.SIZE)))

(PUTPROPS SEQ.ADD1 MACRO [(FORM INC)
                          (\LOLOC (\ADDBASE FORM (OR INC 1])

(PUTPROPS SEQ.GREATERP MACRO ((X Y)
                              (ILESSP (\LOLOC (IDIFFERENCE (IDIFFERENCE X Y)
                                                           1))
                                      32768)))

(PUTPROPS SEQ.GEQ MACRO ((X Y)
                         (ILESSP (\LOLOC (IDIFFERENCE X Y))
                                 32768)))
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS SPP.USER.TIMEOUT SPP.MIN.TIMEOUT SPP.INACTIVITY.TIMEOUT SPP.MAX.FAILED.PROBES XIPTRACEFLG
       XIPTRACEFILE)
)
)

(ADDTOVAR SYSTEMRECLST
          (DATATYPE SPPCON ((SPPXIPLENGTH WORD)
                            (NIL BYTE)
                            (SPPXIPTYPE BYTE)
                            (SPPDESTNSADDRESS0 5 WORD)
                            (SPPDESTSKT# WORD)
                            (SPPSOURCENSADDRESS0 5 WORD)
                            (SPPSOURCESKT# WORD)
```

```
                              (NIL BYTE)
                              (SPPDSTYPE BYTE)
                              (SPPSOURCEID WORD)
                              (SPPDESTID WORD)
                              (SPPSEQNO WORD)
                              (SPPACKNO WORD)
                              (SPPACCEPTNO WORD)
                              (SPPESTABLISHEDP FLAG)
                              (SPPDESTINATIONKNOWN FLAG)
                              (SPPTERMINATEDP FLAG)
                              (SPPOUTPUTABORTEDP FLAG)
                              (SPPOUTPUTABORTEDFN POINTER)
                              (SPPACKPENDING FLAG)
                              (SPPEOMONFORCEOUT FLAG)
                              (SPPSERVERFLAG FLAG)
                              (SPPINPUTBLOCKED FLAG)
                              (SPPINPUTQ POINTER)
                              (SPPRETRANSMITQ POINTER)
                              (SPPRETRANSMITTING POINTER)
                              (SPPLOCK POINTER)
                              (SPPMYNSOCKET POINTER)
                              (SPPACKEDSEQNO WORD)
                              (SPPOUTPUTALLOCNO WORD)
                              (SPPRETRANSMITTIMER POINTER)
                              (SPPACKREQUESTED POINTER)
                              (SPPACKREQTIME POINTER)
                              (SPPACKREQTIMEOUT POINTER)
                              (SPPROUNDTRIPTIME POINTER)
                              (SPPACTIVITYTIMER POINTER)
                              (SPPATTENTIONFN POINTER)
                              (SPPINPKT POINTER)
                              (SPPOUTPKT POINTER)
                              (SPPSYSPKT POINTER)
                              (SPPINPUTSTREAM POINTER)
                              (SPPSUBSTREAM POINTER)
                              (SPPPROCESS POINTER)
                              (SPPALLOCATIONEVENT POINTER)
                              (SPPINPUTEVENT POINTER)
                              (SPPOUTPUTSTREAM POINTER)
                              (SPPWHENCLOSEDFN POINTER)
                              (SPPSTATE POINTER)
                              (SPPERRORHANDLER POINTER)
                              (SPPSERVERFN POINTER)
                              (SPPOTHERXIPHANDLER POINTER)
                              (SPPINACTIVECOUNT POINTER)
                              (SPPINPUTDSTYPE BYTE)
                              (SPPDSTYPECHANGEFN POINTER))))

(/DECLAREDATATYPE 'SPPCON
       '(WORD BYTE BYTE WORD WORD WORD WORD WORD WORD WORD WORD WORD WORD WORD BYTE BYTE WORD WORD WORD
              WORD WORD FLAG FLAG FLAG FLAG FLAG POINTER FLAG FLAG FLAG FLAG POINTER POINTER POINTER POINTER POINTER
              WORD WORD POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER
              POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER BYTE
              POINTER)
       ;; ---field descriptor list elided by lister---
       '82)

(RPAQ? SPP.USER.TIMEOUT 15000)

(RPAQ? SPP.INACTIVITY.TIMEOUT 120000)

(RPAQ? SPP.MIN.TIMEOUT 50)

(RPAQ? SPP.MAX.FAILED.PROBES 5)

(DEFINEQ

(\SPPCONNECTION
  [LAMBDA (HOST SKT# NAME)                                          ; Edited 26-May-91 12:02 by jds

;;; Create an active connection if HOST is specified.  NAME is optional name of connection watcher process.  If HOST is NIL, sets up a listener on
;;; socket SKT#

    [if HOST
        then (SETQ HOST (OR (\COERCE.TO.NSADDRESS HOST SKT#)
                            (\ILLEGAL.ARG HOST]
    (LET [(CON (\SPP.CREATE.CON (AND (NULL HOST)
                                     SKT#]
         (COND
            (HOST                                                   ; User wants to initiate connection to host.
                  (\BLT (LOCF (fetch (SPPCON SPPDESTNSADDRESS0) of CON))
                        HOST \#WDS.NSADDRESS)
                  (replace (SPPCON SPPDESTINATIONKNOWN) of CON with T)))
         [\SPP.CREATE.WATCHER CON (COND
                                     (NAME)
                                     (HOST (CONCAT "SPP#" (SPP.DESTADDRESS CON)))
```

```
                                    (T 'SPPSERVER]
        CON])
```

(\SPP.CREATE.CON

```
[LAMBDA (SKT#)                                          ; Edited 26-May-91 12:02 by jds
```

;; Creates an SPPCON object, initialized to be a connection from this machine, etc. If SKT# is specified, we open that exact socket, else a random ;; user socket.

```
(LET* ((NSOC (OPENNSOCKET SKT#))
       (CON (create SPPCON
                    SPPXIPLENGTH _ (+ \XIPOVLEN \SPPHEAD.LENGTH)
                    SPPXIPTYPE _ \XIPT.SPP
                    SPPSOURCEID _ (LOGOR 32768 (LOGAND (DAYTIME)
                                                       32767))
                    SPPMYNSOCKET _ NSOC
                    SPPSOURCESKT# _ (NSOCKETNUMBER NSOC)
                    SPPACCEPTNO _ \SPP.INITIAL.ALLOCATION)))
      (\BLT (LOCF (fetch (SPPCON SPPSOURCENSADDRESS0) of CON))
            \MY.NSADDRESS
            (SUB1 \#WDS.NSADDRESS))
      CON])
```

(\SPP.CREATE.STREAMS

```
[LAMBDA (CON)                                           ; Edited 26-May-91 12:18 by jds
```

;;; Creates input and output streams for SPP connection CON and installs them appropriately. Returns the input stream

```
(LET [(INSTREAM (\SPP.CREATE.STREAM 'INPUT))
      (OUTSTREAM (\SPP.CREATE.STREAM 'OUTPUT]
     (replace (SPPCON SPPINPUTSTREAM) of CON with INSTREAM)
     (replace (SPPSTREAM SPP.CONNECTION) of INSTREAM with CON)
     (replace (STREAM STRMBOUTFN) of INSTREAM with (FUNCTION \SPP.OTHER.BOUT))
     (replace (SPPCON SPPOUTPUTSTREAM) of CON with OUTSTREAM)
     (replace (SPPSTREAM SPP.CONNECTION) of OUTSTREAM with CON)
     (push (fetch (FDEV DEVICEINFO) of \SPPDEVICE)
           INSTREAM)
     INSTREAM])
```

(\SPP.CREATE.WATCHER

```
[LAMBDA (CON NAME)                                      ; Edited 26-May-91 12:02 by jds
  (replace (SPPCON SPPINPUTEVENT) of CON with (CREATE.EVENT NAME))
  (replace (SPPCON SPPLOCK) of CON with (CREATE.MONITORLOCK NAME))
  (replace (SPPCON SPPROUNDTRIPTIME) of CON with \SPP.INITIAL.ROUNDTRIP)
  (replace (SPPCON SPPPROCESS) of CON with (ADD.PROCESS (LIST (FUNCTION \SPPWATCHER)
                                                              (KWOTE CON))
                                                        'NAME NAME 'RESTARTABLE 'HARDRESET 'AFTEREXIT 'DELETE])
```

(\SPP.SENDPKT

```
[LAMBDA (CON EPKT RETRANSMITP)                          ; Edited 26-May-91 12:21 by jds
```

;; This function makes sure the variable connection information in the packet is current, and actually sends the packet. If the packet is to be ;; retransmitted, the connection must be locked when this function is called. Note that the sequence number is NOT updated; it was allocated ;; once and for all by \SENDSPP

```
(PROG ((ACK# (fetch (SPPCON SPPACKNO) of CON))
       (ALLOC# (fetch (SPPCON SPPACCEPTNO) of CON))
       (BASE (fetch (XIP XIPCONTENTS) of EPKT))
       SEQNO)
      (AND RETRANSMITP (HELP "RETRANSMITP on"))
      (replace (SPPHEAD ACKNO) of BASE with ACK#)
      (replace (SPPHEAD ALLOCNO) of BASE with ALLOC#)
      (replace (SPPCON SPPINPUTBLOCKED) of CON with (SEQ.GREATERP ACK# ALLOC#))
                                                        ; If ACK# > ALLOC# then partner cannot send more data until
                                                        ; we eat some of what we have
      [COND
         ((fetch (SPPHEAD SENDACK) of BASE)
```

;; We start a timer when we send an Ack request, and turn it off when the next packet arrives (in \SPPINPUTWORK.) If the timer ;; expires, we assume that the connection is wedged. Otherwise, the elapsed time will be used to update our estimate of the round ;; trip delay. The timer will go off after the user-level timeout, or twice the round trip delay, whichever is longer.

```
          (SETQ SEQNO (fetch (SPPHEAD SEQNO) of BASE))
          [COND
             ((OR (NOT (fetch (SPPCON SPPACKREQUESTED) of CON))
                  (SEQ.GREATERP SEQNO (fetch (SPPCON SPPACKREQUESTED) of CON)))
              (replace (SPPCON SPPACKREQUESTED) of CON with SEQNO)
              (replace (SPPCON SPPACKREQTIME) of CON with (SETUPTIMER 0 (fetch (SPPCON SPPACKREQTIME)
                                                                               of CON]
          (replace (SPPCON SPPACKREQTIMEOUT) of CON with (SETUPTIMER (UNFOLD (fetch (SPPCON SPPROUNDTRIPTIME)
                                                                                    of CON)
                                                                             2)
                                                                     (fetch (SPPCON SPPACKREQTIMEOUT) of CON]
      (replace (SPPCON SPPACKPENDING) of CON with NIL)  ; If partner asked for an ack, this will satisfy it
      (SENDXIP (fetch (SPPCON SPPMYNSOCKET) of CON)
               EPKT)
```

```
        (replace (SPPCON SPPRETRANSMITTIMER) of CON with (SETUPTIMER (COND
                                                                        ((fetch (SPPCON SPPRETRANSMITTING)
                                                                            of CON)
                                                                         (fetch (SPPCON SPPROUNDTRIPTIME)
                                                                            of CON))
                                                                        (T (UNFOLD (fetch (SPPCON
                                                                                                  SPPROUNDTRIPTIME
                                                                                                  )
                                                                                          of CON)
                                                                                   2)))
                                                                    (fetch (SPPCON SPPRETRANSMITTIMER) of CON])
```

(\FILLINSPP

```
[LAMBDA (CON CCONTROL DSTYPE)                                   ; Edited 26-May-91 12:21 by jds
  (PROG ((EPKT (\ALLOCATE.ETHERPACKET))
         BASE)
        (replace EPTYPE of EPKT with \EPT.XIP)
        (\BLT (LOCF (fetch (XIP XIPLENGTH) of EPKT))
              (LOCF (fetch (SPPCON SPPXIPLENGTH) of CON))
              \#WDS.SPPINFO)                                    ; Fill in canonical SPP packet for this connection
        (SETQ BASE (fetch (XIP XIPCONTENTS) of EPKT))
        (AND CCONTROL (replace (SPPHEAD CC) of BASE with CCONTROL))
        (AND DSTYPE (replace (SPPHEAD DSTYPE) of BASE with DSTYPE))
        (RETURN EPKT])
```

(\SPP.SYSPKT

```
[LAMBDA (CON CCBITS)                                            ; Edited 26-May-91 12:21 by jds
```

;; Return a System packet for the connection with the specified control bits set. Uses the cached packet if there is one.

```
(PROG ((XIP (fetch (SPPCON SPPSYSPKT) of CON))
       BASE)
      [COND
         ((NULL XIP)
          (SETQ XIP (\FILLINSPP CON))
          (replace (SPPCON SPPSYSPKT) of CON with XIP))
         (T (while (fetch EPTRANSMITTING of XIP) do (BLOCK]
      (SETQ BASE (fetch (XIP XIPCONTENTS) of XIP))
      (replace (SPPHEAD CC) of BASE with (LOGOR \SPPHEAD.CC.SYSTEM (OR CCBITS 0)))
      (replace (SPPHEAD SEQNO) of BASE with (fetch (SPPCON SPPSEQNO) of CON))
      (RETURN XIP])
```

(\GETSPP

```
[LAMBDA (CON TIMEOUT PEEKFLG)                                   ; Edited 26-May-91 12:02 by jds
```

;; Obtains the next packet on this SPP connection. If TIMEOUT is specified and expires before a packet arrives, returns NIL. Also returns NIL if ;; the connection is terminated. If PEEKFLG is true, returns the next packet without removing it from queue.

```
(WITH.MONITOR (fetch (SPPCON SPPLOCK) of CON)
    [bind (EPKT _ NIL)
          (TIMER _ (SETUPTIMER (OR TIMEOUT SPP.USER.TIMEOUT)))
       do (COND
             ((AND (SETQ EPKT (\QUEUEHEAD (fetch (SPPCON SPPINPUTQ) of CON)))
                   (SEQ.GREATERP (fetch (SPPCON SPPACKNO) of CON)
                          (fetch (SPPXIP SEQNO) of EPKT)))
```

;; This is the packet we've been waiting for. The ACKNO field has already been incremented in \SPP.HANDLE.DATA

```
              [COND
                 ((NOT PEEKFLG)
                  (UNINTERRUPTABLY
                      (\DEQUEUE (fetch (SPPCON SPPINPUTQ) of CON))
                      (change (fetch (SPPCON SPPACCEPTNO) of CON)
                              (SEQ.ADD1 DATUM)))
                  (COND
                     ((AND (fetch (SPPCON SPPINPUTBLOCKED) of CON)
                           (SEQ.GREATERP (fetch (SPPCON SPPACCEPTNO) of CON)
                                  (fetch (SPPCON SPPACKNO) of CON)))
```

;; Partner was waiting to be able to transmit again, so allow it now. Don't send this gratuitous ack the moment we ;; open up; wait for the window to at least get a couple of packets wide

```
                      (\SPP.SENDPKT CON (\SPP.SYSPKT CON]
              (RETURN EPKT))
             ((OR (AND TIMEOUT (TIMEREXPIRED? TIMER))
                  (fetch (SPPCON SPPTERMINATEDP) of CON))
              (RETURN NIL))
             (T (MONITOR.AWAIT.EVENT (fetch (SPPCON SPPLOCK) of CON)
                       (fetch (SPPCON SPPINPUTEVENT) of CON)
                       TIMER T])])
```

(\SENDSPP

```
[LAMBDA (CON EPKT IGNOREALLOC)                                  ; Edited 26-May-91 12:02 by jds
```

;; Send the next SPP packet over the connection. Blocks if necessary until the allocation window opens up. Returns T if successful, NIL if ;; connection dropped.

```
(CHECK (type? ETHERPACKET EPKT)
```

```
          (NOT (fetch (SPPXIP SYSTEMPACKET) of EPKT)))
    (WITH.MONITOR (fetch (SPPCON SPPLOCK) of CON)
        [bind SEQNO while (NOT (fetch (SPPCON SPPTERMINATEDP) of CON))
           do (COND
                 ((SEQ.GEQ (COND
                              (IGNOREALLOC
                                     ;; Can send attention packet regardless of allocation, but make sure there is room in the retransmit
                                     ;; pool
                                     (SEQ.ADD1 (fetch (SPPCON SPPACKEDSEQNO) of CON)
                                           (SUB1 \SPP.RETRANSMITQ.SIZE)))
                              (T                                    ; Make sure allocation window open
                                 (fetch (SPPCON SPPOUTPUTALLOCNO) of CON)))
                           (fetch (SPPCON SPPSEQNO) of CON))
                  (\BLT (LOCF (fetch (SPPXIP SOURCECONID) of EPKT))
                        (LOCF (fetch (SPPCON SPPSOURCEID) of CON))
                        5)                                          ; Fill in connection id's and sequence numbers
                  (UNINTERRUPTABLY
                      [replace (SPPCON SPPSEQNO) of CON with (SEQ.ADD1 (SETQ SEQNO (fetch (SPPCON SPPSEQNO)
                                                                                       of CON]
                      ;; Bump the sequence number and stuff the packet into the retransmit bin.  This is the only place, I think, where it would
                      ;; hurt us to be interrupted.  After this, it is okay if we are interrupted even before the packet actually gets sent, since the
                      ;; retransmit logic will take over
                      (SETA (fetch (SPPCON SPPRETRANSMITQ) of CON)
                            (IMOD SEQNO \SPP.RETRANSMITQ.SIZE)
                            EPKT))                                  ; advance the packet sequence number.
                  (COND
                     ((AND (EQ SEQNO (fetch (SPPCON SPPOUTPUTALLOCNO) of CON))
                           (NEQ (fetch (SPPXIP DSTYPE) of EPKT)
                                \SPPDSTYPE.ENDREPLY))               ; Sending this packet exhausts our allocation, so request an ack
                                                                    ; in hopes of getting more
                      (replace (SPPXIP SENDACK) of EPKT with T)))
                  (\SPP.SENDPKT CON EPKT)
                  (RETURN T))
                 (T                                                 ; Otherwise, we have to wait until the other end opens up the
                                                                    ; allocation window.
                    (MONITOR.AWAIT.EVENT (fetch (SPPCON SPPLOCK) of CON)
                           (fetch (SPPCON SPPALLOCATIONEVENT) of CON])])
```

(\SPP.SEND.ENDREPLY

```
  [LAMBDA (CON NOACK)                                               (* bvm%: "22-Jun-84 14:57")
    (\SENDSPP CON (\FILLINSPP CON (LOGOR \SPPHEAD.CC.EOM (COND
                                                           (NOACK 0)
                                                           (T \SPPHEAD.CC.ACKNOWLEDGE)))
                      \SPPDSTYPE.ENDREPLY)
          T])
```

(\TERMINATESPP

```
  [LAMBDA (CON TIMEOUT)                                             ; Edited 26-May-91 12:02 by jds
                                                                    ; Reliable connection termination, as in section 7.5 of the spec.
    (WITH.MONITOR (fetch (SPPCON SPPLOCK) of CON)
        (PROG NIL
              (SELECTC (fetch (SPPCON SPPSTATE) of CON)
                   ((LIST \SPS.CLOSED \SPS.ABORTED)
                        (RETURN NIL))
                   ((LIST \SPS.INIT \SPS.LISTENING)
                        (replace (SPPCON SPPTERMINATEDP) of CON with T)
                        (replace (SPPCON SPPSTATE) of CON with \SPS.ABORTED)
                        (RETURN NIL))
                   (\SPS.OPEN                                       ; We initiate the termination by sending an END packet.
                              (\SENDSPP CON (\FILLINSPP CON (LOGOR \SPPHEAD.CC.ACKNOWLEDGE \SPPHEAD.CC.EOM)
                                                \SPPDSTYPE.END)
                                    T)
                              (replace (SPPCON SPPSTATE) of CON with \SPS.ENDSENT))
                   NIL)
              [COND
                 ((NEQ TIMEOUT 0)
                  (bind (TIMER _ (SETUPTIMER (OR TIMEOUT 5000))) do (MONITOR.AWAIT.EVENT (fetch (SPPCON SPPLOCK)
                                                                                             of CON)
                                                                                         (fetch (SPPCON SPPINPUTEVENT)
                                                                                            of CON)
                                                                                         TIMER T)
                                                                    (SELECTC (fetch (SPPCON SPPSTATE)
                                                                                of CON)
                                                                             (\SPS.CLOSED (RETURN T))
                                                                             (\SPS.ABORTED (RETURN))
                                                                             NIL)
                        repeatuntil (TIMEREXPIRED? TIMER]
              (replace (SPPCON SPPSTATE) of CON with \SPS.ABORTED)
              (DEL.PROCESS (PROG1 (fetch (SPPCON SPPPROCESS) of CON)
                                  (replace (SPPCON SPPPROCESS) of CON with NIL)))
              (RETURN NIL)))])
```

(\SPP.CLEANUP

```
  [LAMBDA (CON)                                                    ; Edited 26-May-91 12:03 by jds
    ;; Called when \SPPWATCHER exits.
    (SELECTQ RESETSTATE
         (HARDRESET                                                ; Don't do this if process is being restarted after HARDRESET
                    NIL)
         (WITH.MONITOR (fetch (SPPCON SPPLOCK) of CON)
             (PROG ((INSTREAM (fetch (SPPCON SPPINPUTSTREAM) of CON))
                    FN)
                   (replace (SPPCON SPPTERMINATEDP) of CON with T)
                   (NOTIFY.EVENT (fetch (SPPCON SPPINPUTEVENT) of CON))
                   (NOTIFY.EVENT (fetch (SPPCON SPPALLOCATIONEVENT) of CON))
                                                                    ; We just notified anyone who might be blocked waiting for
                                                                    ; something to happen on this connection.
                   (replace (FDEV DEVICEINFO) of \SPPDEVICE with (DREMOVE INSTREAM (fetch (FDEV DEVICEINFO)
                                                                                           of \SPPDEVICE)))
                   [COND
                      ((SETQ FN (fetch (SPPCON SPPWHENCLOSEDFN) of CON))
                       (for F in (COND
                                     ((AND (LISTP FN)
                                           (NEQ (CAR FN)
                                                'LAMBDA))
                                      FN)
                                     (T (LIST FN)))
                          do (APPLY* F INSTREAM CON]
                   (replace (SPPCON SPPOUTPUTSTREAM) of CON with (replace (SPPCON SPPINPUTSTREAM) of CON
                                                                          with (replace (SPPCON SPPSUBSTREAM) of CON
                                                                                        with NIL)))
                                                                    ; Snap circular links before we lose control
                   (CLOSENSOCKET (PROG1 (fetch (SPPCON SPPMYNSOCKET) of CON)
                                        (replace (SPPCON SPPMYNSOCKET) of CON with NIL))
                          T)
                   (replace (SPPCON SPPPROCESS) of CON with NIL)))])

)

(DEFINEQ
```

(\SPPWATCHER

```
  [LAMBDA (SPPCON)                                                 ; Edited 26-May-91 12:03 by jds
    (DECLARE (SPECVARS SPPCON))
    (RESETSAVE NIL (LIST (FUNCTION \SPP.CLEANUP)
                         SPPCON))
    (PROCESSPROP (THIS.PROCESS)
           'INFOHOOK
           (FUNCTION \SPP.INFO.HOOK))
    (if (NULL (fetch (SPPCON SPPACTIVITYTIMER) of SPPCON))
        then (replace (SPPCON SPPACTIVITYTIMER) of SPPCON with (SETUPTIMER 0)))
    (WITH.MONITOR (fetch (SPPCON SPPLOCK) of SPPCON)
        [bind (SOCEVENT _ (NSOCKETEVENT (fetch (SPPCON SPPMYNSOCKET) of SPPCON)))
              (ACKINTERVAL _ (IQUOTIENT (TIMES SPP.USER.TIMEOUT 2)
                                        3))
              ACTIVITY TIMER until (fetch (SPPCON SPPTERMINATEDP) of SPPCON)
           do [COND
                 ((SETQ ACTIVITY (\SPP.HANDLE.INPUT SPPCON))       ; Got some input, so partner is alive
                  (replace (SPPCON SPPINACTIVECOUNT) of SPPCON with NIL)
                  (SETUPTIMER 0 (fetch (SPPCON SPPACTIVITYTIMER) of SPPCON]
              (COND
                 ((AND (NULL ACTIVITY)
                       (NOT (fetch (SPPCON SPPESTABLISHEDP) of SPPCON))
                       (NOT (fetch (SPPCON SPPDESTINATIONKNOWN) of SPPCON)))
                                                                    ; Nothing happening, and we're just listening.  Go back to sleep
                  (MONITOR.AWAIT.EVENT (fetch (SPPCON SPPLOCK) of SPPCON)
                         SOCEVENT))
                 (T                                                 ; Do what is appropriate for state of connection
                    (SETQ TIMER (fetch (SPPCON SPPRETRANSMITTIMER) of SPPCON))
                                                                    ; Default time we might want to do something next
                    [COND
                       ((fetch (SPPCON SPPRETRANSMITTING) of SPPCON)
                                                                    ; In the midst of retransmitting one or more packets that appear
                                                                    ; to have been missed
                        (\SPP.RETRANSMIT.NEXT SPPCON))
                       ((fetch (SPPCON SPPACKPENDING) of SPPCON)    ; Partner asked for an ack, and we haven't sent anything yet as
                                                                    ; part of our routine activity, so send simple ack
                        (\SPP.SENDPKT SPPCON (\SPP.SYSPKT SPPCON (if (\CLOCKGREATERP (fetch (SPPCON SPPACKREQTIME)
                                                                                             of SPPCON)
                                                                                     ACKINTERVAL)
                                                                     then
                                                                    ; if we haven't timed an ACK in a while, take the opportunity now.
                                                                    ; This lets us kill two birds with one stone, er, packet--responding
                                                                    ; to partner's ack, and getting round trip info ourselves.
                                                                                 \SPPHEAD.CC.ACKNOWLEDGE)))
                        (replace (SPPCON SPPACKPENDING) of SPPCON with NIL))
                       ((NULL ACTIVITY)                             ; No input activity
                        (COND
                           ((fetch (SPPCON SPPACKREQUESTED) of SPPCON)
```

```
                                                    ; We requested an ack, haven't heard anything back
                    (if (TIMEREXPIRED? (SETQ TIMER (fetch (SPPCON SPPACKREQTIMEOUT) of SPPCON)))
                        then (\SPP.NOT.RESPONDING SPPCON)))
                   ((OR (SEQ.GREATERP (fetch (SPPCON SPPSEQNO) of SPPCON)
                                      (fetch (SPPCON SPPACKEDSEQNO) of SPPCON))
                        (SEQ.GREATERP (fetch (SPPCON SPPSEQNO) of SPPCON)
                                      (fetch (SPPCON SPPOUTPUTALLOCNO) of SPPCON)))
                                                    ; Not all outstanding packets are acked, or we are out of
                                                    ; allocation
                    (if (TIMEREXPIRED? TIMER)
                        then                        ; Time to poke again
                             (\SPP.PROBE SPPCON)))
                   (T                               ; Connection is quiet.  Periodically poke the other end to make
                                                    ; sure it's still alive
                      (if (\CLOCKGREATERP (fetch (SPPCON SPPACTIVITYTIMER) of SPPCON)
                                 SPP.USER.TIMEOUT)
                          then                      ; Haven't heard anything in a while.  Next time thru,
                                                    ; SPPACKREQUESTED will be true, so we'll never do this twice
                                                    ; in a row.
                               (\SPP.PROBE SPPCON)
                          else                      ; Don't need to wake up again until previous clause wants it
                               (SETUPTIMER [IMAX 0 (- SPP.USER.TIMEOUT (CLOCKDIFFERENCE (fetch (SPPCON
                                                                                                SPPACTIVITYTIMER
                                                                                                      )
                                                                                               of SPPCON]
                                                   TIMER]
                (if (fetch (SPPCON SPPTERMINATEDP) of SPPCON)
                    then (RETURN))
                (MONITOR.AWAIT.EVENT (fetch (SPPCON SPPLOCK) of SPPCON)
                       SOCEVENT TIMER T])])
```

(**SPP.HANDLE.INPUT**

```
[LAMBDA (CON)                                       ; Edited 26-May-91 12:21 by jds
```

;; Handle all queued input packets. Returns T if there was activity on the connection.

```
(PROG (XIP SPPBASE PKTSEQNO ACTIVE? ATTN ACKED ACKRECEIVED ALLOCINCREASED ADDRESSEDID NEWALLOCNO MAXALLOCNO)
  LOOP
      (COND
         ((fetch (SPPCON SPPTERMINATEDP) of CON)
          (RETURN T)))
      (SETQ XIP (GETXIP (fetch (SPPCON SPPMYNSOCKET) of CON)))
      (COND
         ((NULL XIP)
          [COND
             ((AND ACKRECEIVED (NOT ALLOCINCREASED)
                   (SEQ.GREATERP (fetch (SPPCON SPPSEQNO) of CON)
                                 (fetch (SPPCON SPPACKEDSEQNO) of CON))
                   (NULL (fetch (SPPCON SPPRETRANSMITTING) of CON)))
              ;; We received an apparently genuine ack, but there are still unacked packets, so assume that they have not been seen--start
              ;; retransmitting them.  The test for ALLOCINCREASED is in the hopes that this ack was so old that future acks will say the data
              ;; arrived okay
              (replace (SPPCON SPPRETRANSMITTING) of CON with (fetch (SPPCON SPPACKEDSEQNO) of CON]
          (RETURN ACTIVE?)))
      (SELECTC (fetch (XIP XIPTYPE) of XIP)
          (\XIPT.SPP)
          (\XIPT.ERROR (COND
                          ((EQ (fetch ERRORXIPCODE of XIP)
                               \XIPE.NOSOCKET)       ; Partner not there, or disappeared
                           (replace (SPPCON SPPTERMINATEDP) of CON with T)
                           (\RELEASE.ETHERPACKET XIP)
                           (RETURN T)))
                       (GO DROPIT))
          (PROGN (APPLY* (OR (fetch (SPPCON SPPOTHERXIPHANDLER) of CON)
                             (FUNCTION RELEASE.XIP))
                         XIP
                         (fetch (SPPCON SPPMYNSOCKET) of CON))
                 (GO LOOP)))
      (SETQ SPPBASE (fetch (XIP XIPCONTENTS) of XIP))
      (COND
         ((OR (AND (fetch (SPPCON SPPESTABLISHEDP) of CON)
                   (NEQ (fetch (SPPHEAD SOURCECONID) of SPPBASE)
                        (fetch (SPPCON SPPDESTID) of CON)))
              (AND (NEQ (SETQ ADDRESSEDID (fetch (SPPHEAD DESTCONID) of SPPBASE))
                        (fetch (SPPCON SPPSOURCEID) of CON))
                   (NEQ ADDRESSEDID 0)))
          ;; If the connection has already been established, then both connection IDs must match.  Otherwise, the destination ID in the packet
          ;; must be ours if it is nonzero.
          (\SPPSENDERROR CON XIP "Wrong connection ID.")
          (GO DROPIT)))
      (SETQ PKTSEQNO (fetch (SPPHEAD SEQNO) of SPPBASE))
      (COND
         ((OR (SEQ.GREATERP (fetch (SPPCON SPPACKNO) of CON)
                            (SEQ.ADD1 PKTSEQNO 3000))
              (SEQ.GREATERP PKTSEQNO (SEQ.ADD1 (fetch (SPPCON SPPACCEPTNO) of CON)
```

```
                                                2)))
            ;; Sequence numbers more than 1 or 2 past the allocation or delayed by more than a few thousand are grounds for generating an
            ;; error response.  See section 7.2 of the spec.
             (\SPPSENDERROR CON XIP "Packet out of allocation sequence.")
             (GO DROPIT)))                                          ; We have a legal packet for this connection.
        (COND
           [(NOT (fetch (SPPCON SPPESTABLISHEDP) of CON))           ; We're just now establishing the connection.
             (\SPP.ESTABLISH CON XIP)
             (COND
                ((fetch (SPPCON SPPSERVERFLAG) of CON)
                 ;; This process is a server.  Remain a server in the listening state
                 (GO LOOP]
           (T (SETQ ACTIVE? T)))
        (COND
           ((fetch (SPPHEAD ATTENTION) of SPPBASE)
            (COND
               ((fetch (SPPHEAD SYSTEMPACKET) of SPPBASE)
                (\SPPSENDERROR CON XIP "Both System and Attention control bits?")
                (GO DROPIT)))
            (COND
               ((IGREATERP (IDIFFERENCE (fetch (XIP XIPLENGTH) of XIP)
                                        (IPLUS \XIPOVLEN \SPPHEAD.LENGTH))
                           1)
                (\SPPSENDERROR CON XIP "More than 1 byte of data in Attention packet?")
                (GO DROPIT)))
            (SETQ ATTN T)))
        (COND
           ((SEQ.GREATERP (SETQ ACKED (fetch (SPPHEAD ACKNO) of SPPBASE))
                  (fetch (SPPCON SPPACKEDSEQNO) of CON))
            (\SPP.RELEASE.ACKED.PACKETS CON ACKED)))
        [COND
           ([AND (SEQ.GREATERP (SETQ NEWALLOCNO (fetch (SPPHEAD ALLOCNO) of SPPBASE))
                         (fetch (SPPCON SPPOUTPUTALLOCNO) of CON))
                 (OR (SEQ.GEQ (SETQ MAXALLOCNO (IPLUS (fetch (SPPCON SPPACKEDSEQNO) of CON)
                                                      (SUB1 \SPP.RETRANSMITQ.SIZE)))
                              NEWALLOCNO)
                     (SEQ.GREATERP (SETQ NEWALLOCNO MAXALLOCNO)
                            (fetch (SPPCON SPPOUTPUTALLOCNO) of CON]
                                                                    ; Limit our actual allocation to the maximum we are willing to
                                                                    ; buffer up
            (replace (SPPCON SPPOUTPUTALLOCNO) of CON with NEWALLOCNO)
            (SETQ ALLOCINCREASED T)
            (NOTIFY.EVENT (fetch (SPPCON SPPALLOCATIONEVENT) of CON]
        (COND
           ((fetch (SPPHEAD SENDACK) of SPPBASE)                    ; The other end wants an acknowledgment.  Wait until we have
                                                                    ; processed all input
            (replace (SPPCON SPPACKPENDING) of CON with T)))
        (COND
           ((fetch (SPPHEAD SYSTEMPACKET) of SPPBASE)               ; Don't keep system packets
            (RELEASE.XIP XIP))
           (T (\SPP.HANDLE.DATA CON XIP)                            ; Note that this call may increment the connection's ACKNO field.

             ))
        (COND
           ([AND (fetch (SPPCON SPPACKREQUESTED) of CON)
                 (OR (NEQ ACKED (fetch (SPPCON SPPACKREQUESTED) of CON))
                     (EQ ACKED (fetch (SPPCON SPPSEQNO) of CON]
            ;; This is the first packet that has arrived since we turned on the Ack request timer in \SPP.SENDPKT.  Turn off the timer and update
            ;; our estimate of round trip delay.  This packet might be delayed, and not really in response to our Ack request.  The NEQ test filters
            ;; out packets that cannot possibly be in response to our ACK: if partner received our request at seqno N, and has seen up thru N-1,
            ;; ACKED should be N+1, unless the ack request was on a system packet.
            (replace (SPPCON SPPROUNDTRIPTIME) of CON
               with (LRSH (IPLUS (ITIMES 3 (fetch (SPPCON SPPROUNDTRIPTIME) of CON))
                                 (IMAX SPP.MIN.TIMEOUT (IMIN (CLOCKDIFFERENCE (fetch (SPPCON SPPACKREQTIME)
                                                                                     of CON))
                                                             SPP.USER.TIMEOUT)))
                          2))
            (replace (SPPCON SPPACKREQUESTED) of CON with NIL)
            (SETQ ACKRECEIVED T)))
        (COND
           (ATTN (\SPP.HANDLE.ATTN CON XIP)))
        (GO LOOP)
     DROPIT
        (RELEASE.XIP XIP)
        (GO LOOP])
```

(\SPP.HANDLE.DATA

```
  [LAMBDA (CON XIP)                                                 ; Edited 26-May-91 12:03 by jds
    ;; This function is called when a non-System packet has arrived for a connection.  It inserts the packet in the proper place in the queue, ordered by
    ;; sequence number.  If the packet is a duplicate, it is dropped.
    (PROG ((ACKNO (fetch (SPPCON SPPACKNO) of CON))
           (INQ (fetch (SPPCON SPPINPUTQ) of CON))
```

```
           (XIPNO (fetch (SPPXIP SEQNO) of XIP))
           CURRENT NEXT PKTNO)
          (CHECK (\SPP.CHECK.INPUT.QUEUE CON))
          [COND
             ((SEQ.GREATERP ACKNO XIPNO)                          ; This packet is a duplicate, so drop it.
              (RELEASE.XIP XIP)
              (RETURN))
             ([OR (NULL (SETQ CURRENT (\QUEUEHEAD INQ)))
                  (SEQ.GREATERP XIPNO (fetch (SPPXIP SEQNO) of (fetch SYSQUEUETAIL of INQ]
                                                                  ; Goes at tail end of queue.
              (\ENQUEUE INQ XIP))
             ((SEQ.GREATERP (SETQ PKTNO (fetch (SPPXIP SEQNO) of CURRENT))
                    XIPNO)                                        ; Goes right at head of queue.
              (replace QLINK of XIP with CURRENT)
              (replace SYSQUEUEHEAD of INQ with XIP))
             (T (do                                               ; Loop until the correct place is found for this packet.
                    (COND
                       ((EQ XIPNO PKTNO)                          ; This packet is a duplicate, so drop it.
                        (RELEASE.XIP XIP)
                        (RETURN)))
                    (SETQ NEXT (fetch QLINK of CURRENT))
                    (SETQ PKTNO (fetch (SPPXIP SEQNO) of NEXT))
                    (COND
                       ((SEQ.GREATERP PKTNO XIPNO)                ; Here's where it goes.
                        (replace QLINK of XIP with NEXT)
                        (replace QLINK of CURRENT with XIP)
                        (RETURN)))
                    (SETQ CURRENT NEXT]
          (SELECTC (fetch (SPPXIP DSTYPE) of XIP)
              (\SPPDSTYPE.END
                   (replace (SPPCON SPPSTATE) of CON with \SPS.ENDRECEIVED)
                   (LET ((OUTSTREAM (fetch (SPPCON SPPOUTPUTSTREAM) of CON)))
                                                                  ; Can't send any more
                        (replace (STREAM ACCESS) of OUTSTREAM with NIL)
                        (replace (STREAM STRMBOUTFN) of OUTSTREAM with (FUNCTION \SPP.STREAM.LOST))
                        ;; Make attempt to output to this stream go thru same error mechanism as other ways of losing stream, rather than
                        ;; getting lisp's FILE NOT OPEN error.

                        )
                   (\SPP.SEND.ENDREPLY CON)
                   (replace (SPPCON SPPSTATE) of CON with \SPS.DALLYING))
              (\SPPDSTYPE.ENDREPLY
                   (SELECTC (fetch (SPPCON SPPSTATE) of CON)
                        (\SPS.DALLYING                            ; This is the closing end reply, so can quit now

                            )
                        (\SPS.ENDSENT                             ; This is the reply to our END
                                      (\SPP.SEND.ENDREPLY CON T))
                        (\SPPSENDERROR CON XIP "unexpected ENDREPLY"))
                   (replace (SPPCON SPPSTATE) of CON with \SPS.CLOSED)
                   (replace (SPPCON SPPTERMINATEDP) of CON with T))
              NIL)
          (COND
             ((EQ XIPNO ACKNO)
              ;; Looks like this packet opens the way for some acknowledgements.  Find the end of the run of consecutive packets starting with the
              ;; one we've just inserted.
              (while (AND (SETQ XIP (fetch QLINK of XIP))
                          (EQ (SETQ PKTNO (fetch (SPPXIP SEQNO) of XIP))
                              (SEQ.ADD1 XIPNO)))
                 do (SETQ XIPNO PKTNO))
              (replace (SPPCON SPPACKNO) of CON with (SEQ.ADD1 XIPNO))
              (NOTIFY.EVENT (fetch (SPPCON SPPINPUTEVENT) of CON])
```

(\SPP.HANDLE.ATTN

```
  [LAMBDA (CON XIP)                                               ; Edited 26-May-91 12:03 by jds
```

;;; Called when a packet is received with Attention bit set

```
    (PROG ((ATTENTIONFN (fetch (SPPCON SPPATTENTIONFN) of CON))
           (BYTE (fetch (SPPXIP FIRSTSPPDATABYTE) of XIP))
           (DSTYPE (fetch (SPPXIP DSTYPE) of XIP))
           STREAM)
          (COND
             ((AND ATTENTIONFN (for FN in (COND
                                             ((OR (NLISTP ATTENTIONFN)
                                                  (MEMB (CAR ATTENTIONFN)
                                                        LAMBDASPLST))
                                              (LIST ATTENTIONFN))
                                             (T ATTENTIONFN))
                                   thereis (APPLY* FN (fetch (SPPCON SPPINPUTSTREAM) of CON)
                                                   BYTE DSTYPE)))    ; Somebody knew how to handle it

              )
             (NSWIZARDFLG                                         ; Some other kind of attention we don't know about
                    (printout PROMPTWINDOW .TAB0 0 "[Attention packet (" BYTE ")]"])
```

(\SPP.RELEASE.ACKED.PACKETS

```
[LAMBDA (CON ACKNO)                                          ; Edited 26-May-91 12:03 by jds

;;; Releases packets that are acked by incoming ACKNO, i.e., any packets with sequence number less than ACKNO.  Packets are held in
;;; SPPRETRANSMITQ array

    (bind (POOL _ (fetch (SPPCON SPPRETRANSMITQ) of CON))
          (OLDACKNO _ (fetch (SPPCON SPPACKEDSEQNO) of CON))
          (MAXACKNO _ (fetch (SPPCON SPPSEQNO) of CON))
          XIP while (SEQ.GREATERP ACKNO OLDACKNO) do [COND
                                                      ((EQ OLDACKNO MAXACKNO)
                                                       (RETURN (AND XIPTRACEFLG (HELP "SPP Partner acked a
                                                                                      packet I haven't sent
                                                                                      yet" ACKNO]
                                                     (SETQ XIP (ELT POOL (RETRANSMITINDEX OLDACKNO)))
                                                     [CHECK (AND XIP (EQ OLDACKNO (fetch (SPPXIP SEQNO)
                                                                                         of XIP]
                                                     (UNINTERRUPTABLY
                                                         (SETA POOL (RETRANSMITINDEX OLDACKNO)
                                                               NIL)
                                                         (RELEASE.XIP XIP)
                                                         (replace (SPPCON SPPACKEDSEQNO) of CON
                                                            with (SETQ OLDACKNO (SEQ.ADD1 OLDACKNO))))
                                                     (replace (SPPCON SPPRETRANSMITTING) of CON with NIL)
                                                                 ; If we get ANY interesting ack, stop retransmission until we
                                                                 ; figure out what's going on
                                                     ])
```

(\SPP.NOT.RESPONDING

```
[LAMBDA (CON)                                                ; Edited 26-May-91 12:03 by jds
    ;; There hasn't been any response to our probes for a while.
    (COND
       ((AND (>= (replace (SPPCON SPPINACTIVECOUNT) of CON with (ADD1 (OR (fetch (SPPCON SPPINACTIVECOUNT)
                                                                             of CON)
                                                                          0)))
                 SPP.MAX.FAILED.PROBES)
             (OR (NOT (fetch (SPPCON SPPESTABLISHEDP) of CON))
                 (\CLOCKGREATERP (fetch (SPPCON SPPACTIVITYTIMER) of CON)
                        SPP.INACTIVITY.TIMEOUT)))
        ;; If the connection hasn't been established yet, or if the roundtrip time is intolerably long, we drop the connection.
        (replace (SPPCON SPPTERMINATEDP) of CON with T))
       (T (replace (SPPCON SPPROUNDTRIPTIME) of CON with (IMIN SPP.USER.TIMEOUT (TIMES (fetch (SPPCON
                                                                                                SPPROUNDTRIPTIME
                                                                                                )
                                                                                           of CON)
                                                                                        2)))
                                                             ; Increase our estimate of the time it takes the other end to
                                                             ; respond.
          (\SPP.PROBE CON)
          (COND
             ((AND (fetch (SPPCON SPPESTABLISHEDP) of CON)
                   (EQ (fetch (SPPCON SPPINACTIVECOUNT) of CON)
                       (- SPP.MAX.FAILED.PROBES 2)))     ; Warn the user after a while that the other end may have
                                                         ; crashed, but hang in there.
              (if (\CLOCKGREATERP (fetch (SPPCON SPPACTIVITYTIMER) of CON)
                        (LRSH SPP.INACTIVITY.TIMEOUT 2))
                  then (printout PROMPTWINDOW T (PROCESSPROP (THIS.PROCESS)
                                                             'NAME)
                               " not responding. ")
                else                                     ; Don't be unduly alarming--it hasn't been that long.  If the round
                                                         ; trip time had once been exceedingly low, doubling it a few times
                                                         ; doesn't get us very far, so back off
                     (add (fetch (SPPCON SPPINACTIVECOUNT) of CON)
                          -1])
```

(\SPP.PROBE

```
[LAMBDA (CON)                                                (* bvm%: " 2-Aug-84 16:32")
    ;; Send out a system packet requesting acknowledgement from other side.
    (\SPP.SENDPKT CON (\SPP.SYSPKT CON \SPPHEAD.CC.ACKNOWLEDGE])
```

(\SPP.RETRANSMIT.NEXT

```
[LAMBDA (CON)                                                ; Edited 26-May-91 12:04 by jds
  (PROG ((SEQNO (fetch (SPPCON SPPRETRANSMITTING) of CON))
         XIP)
        (SETQ XIP (ELT (fetch (SPPCON SPPRETRANSMITQ) of CON)
                       (IMOD SEQNO \SPP.RETRANSMITQ.SIZE)))
        (CHECK (EQ SEQNO (fetch (SPPXIP SEQNO) of XIP)))
        [replace (SPPXIP SENDACK) of XIP with (if T
                                                  then T
```

```
                                                    else                ; Turn off any undesired acknowledge bit
                                                      (EQ SEQNO (fetch (SPPCON SPPOUTPUTALLOCNO) of CON]
          (replace (SPPCON SPPRETRANSMITTING) of CON with (COND
                                                             ((EQ (SETQ SEQNO (SEQ.ADD1 SEQNO))
                                                                  (fetch (SPPCON SPPSEQNO) of CON))
                                                              ; Finished
                                                              NIL)
                                                             (T SEQNO)))
          (\SPP.SENDPKT CON XIP])
```

(\SPP.DUPLICATE.REQUEST

```
  [LAMBDA (XIP)                                                     ; Edited 26-May-91 11:58 by jds
```

;;; Return T if the incoming XIP is a connection request for a connection we've already established

```
    (bind CONNECTION (SOURCEID _ (fetch (SPPXIP SOURCECONID) of XIP)) for INSTREAM in (fetch (FDEV DEVICEINFO)
                                                                                          of \SPPDEVICE)
       thereis (AND (SETQ CONNECTION (fetch (SPPSTREAM SPP.CONNECTION) of INSTREAM))
                    (EQ SOURCEID (fetch (SPPCON SPPDESTID) of CONNECTION])
```

(\SPP.ESTABLISH

```
  [LAMBDA (INITCON XIP)                                             ; Edited 26-May-91 12:21 by jds
```

;;; The arrival of XIP causes this SPP connection to be established. Fix up state as appropriate

```
    (PROG (CON INSTREAM NAME)
          [COND
             ((NOT (fetch (SPPCON SPPSERVERFLAG) of INITCON))
              (SETQ CON INITCON)                                    ; For user connection, need to update socket info, as server may
                                                                    ; have switched from a well-known socket to a private one.
              (\BLT (LOCF (fetch (SPPCON SPPDESTNSADDRESS0) of CON))
                    (LOCF (fetch (XIP XIPSOURCENET) of XIP))
                    \#WDS.NSADDRESS))
             (T                                                     ;; The connection was opened in server mode.  Create a new spp connection, and establish it to the remote side, spawning a new
                                                                    ;; process
                (COND
                   ((\SPP.DUPLICATE.REQUEST XIP)                    ; We've already spawned a server for this source
                    (RETURN)))
                (SETQ CON (\SPP.CREATE.CON))
                (\BLT (LOCF (fetch (SPPCON SPPDESTNSADDRESS0) of CON))
                      (LOCF (fetch (XIP XIPSOURCENET) of XIP))
                      \#WDS.NSADDRESS)                              ; Fill in address of port that contacted us
                [SETQ NAME (CONCAT (PROCESSPROP (fetch (SPPCON SPPPROCESS) of INITCON)
                                                'NAME)
                                   '+
                                   (OCTALSTRING (fetch (SPPCON SPPSOURCESKT#) of CON]
                (replace (SPPCON SPPATTENTIONFN) of CON with (fetch (SPPCON SPPATTENTIONFN) of INITCON))
                                                                    ; Copy some methods from the listener con
                (replace (SPPCON SPPWHENCLOSEDFN) of CON with (fetch (SPPCON SPPWHENCLOSEDFN) of INITCON))
                (replace (SPPCON SPPERRORHANDLER) of CON with (fetch (SPPCON SPPERRORHANDLER) of INITCON]
          (replace (SPPCON SPPDESTID) of CON with (fetch (SPPXIP SOURCECONID) of XIP))
          (replace (SPPCON SPPSYSPKT) of CON with NIL)              ; Flush any cached sys packet, now out of date
          (replace (SPPCON SPPESTABLISHEDP) of CON with T)
          (replace (SPPCON SPPSTATE) of CON with \SPS.OPEN)
          (replace (SPPCON SPPDESTINATIONKNOWN) of CON with T)
          (if NAME
              then                                                  ; Finally, get server going
                   (SETQ INSTREAM (\SPP.CREATE.STREAMS CON))
                   (\SPP.CREATE.WATCHER CON NAME)
                   (WITH.MONITOR (fetch (SPPCON SPPLOCK) of CON)    ; Have to reply to the sender so he knows our id & socket
                       (\SPP.PROBE CON))
                   (ADD.PROCESS (LIST (fetch (SPPCON SPPSERVERFN) of INITCON)
                                      INSTREAM
                                      (SPPOUTPUTSTREAM INSTREAM))
                                'AFTEREXIT
                                'DELETE))
          (NOTIFY.EVENT (fetch (SPPCON SPPINPUTEVENT) of CON])
```

(\SPPGETERROR

```
  [LAMBDA (CON TRIALPKT MOREMSG)                                    (* ecc " 3-OCT-83 17:09")
    (if XIPTRACEFLG
        then (printout XIPTRACEFILE "Error packet received on Sequenced Packet Protocol connection." T)
             (PRINTPACKET TRIALPKT NIL XIPTRACEFILE)
             (if MOREMSG
                 then (printout XIPTRACEFILE .TAB0 0 MOREMSG))
             (TERPRI XIPTRACEFILE])
```

(\SPPSENDERROR

```
  [LAMBDA (CON EPKT MSG)                                            (* bvm%: " 8-Mar-85 16:17")
                                                                    ; Stub for now
    (COND
       ((OR XIPTRACEFLG NSWIZARDFLG)
```

```
        (printout XIPTRACEFILE MSG T)
        (PRINTPACKET EPKT NIL XIPTRACEFILE)
        (TERPRI XIPTRACEFILE])

)

;; Stream interface to Sequenced Packet Protocol.

(DEFINEQ
```

(\INITSPP

```
  [LAMBDA NIL                                                    ; Edited 26-May-91 11:58 by jds
    ;; Set up devices so that SPP streams can be used generically.  The Bulk Data device enables a naive stream user to read or write a Bulk Data
    ;; object.
    [\DEFINEDEVICE NIL (SETQ \SPPDEVICE (\CREATE.SPP.DEVICE 'SPP (FUNCTION SPP.CLOSE]
    (replace (FDEV EVENTFN) of \SPPDEVICE with (FUNCTION \SPP.EVENTFN))
    (\DEFINEDEVICE NIL (SETQ \SPP.BULKDATA.DEVICE (\CREATE.SPP.DEVICE 'COURIER.BULK.DATA (FUNCTION
                                                                                          \BULK.DATA.CLOSE])
```

(\SPP.EVENTFN

```
  [LAMBDA (DEVICE EVENT)                                         ; Edited 26-May-91 11:58 by jds
    ;; Fixed to copy DEVICEINFO, since SPP.CLOSE DREMOVEs from it - TAL
    (SELECTQ EVENT
         (BEFORELOGOUT                                           ; Abort any open streams before we logout
                       (for STREAM in (APPEND (fetch (FDEV DEVICEINFO) of DEVICE)) do (SPP.CLOSE STREAM T)))
         NIL])
```

(\CREATE.SPP.DEVICE

```
  [LAMBDA (NAME CLOSEFN)                                         (* bvm%: " 9-Jun-85 16:39")
    (create FDEV
            DEVICENAME _ NAME
            FDBINABLE _ T
            BUFFERED _ T
            EVENTFN _ (FUNCTION NILL)
            TRUNCATEFILE _ (FUNCTION NILL)
            CLOSEFILE _ CLOSEFN
            BIN _ (FUNCTION \BUFFERED.BIN)
            BOUT _ (FUNCTION \BUFFERED.BOUT)
            EOFP _ (FUNCTION SPP.EOFP)
            READP _ (FUNCTION SPP.READP)
            PEEKBIN _ (FUNCTION \BUFFERED.PEEKBIN)
            BACKFILEPTR _ (FUNCTION SPP.BACKFILEPTR)
            FORCEOUTPUT _ (FUNCTION SPP.FORCEOUTPUT)
            BLOCKIN _ (FUNCTION \BUFFERED.BINS)
            BLOCKOUT _ (FUNCTION \SPP.BOUTS)
            GETNEXTBUFFER _ (FUNCTION \SPP.GETNEXTBUFFER)
            GETFILEPTR _ (FUNCTION \SPP.GETFILEPTR)
            SETFILEPTR _ (FUNCTION \SPP.SETFILEPTR])
```

(SPP.OPEN

```
  [LAMBDA (HOST SOCKET PROBEP NAME PROPS)                        ; Edited 26-May-91 12:04 by jds
    (RESETLST
        [LET ((CON (\SPPCONNECTION HOST SOCKET NAME)))
             (OBTAIN.MONITORLOCK (fetch (SPPCON SPPLOCK) of CON)
                    NIL T)
             [RESETSAVE (fetch (SPPCON SPPMYNSOCKET) of CON)
                    '(AND RESETSTATE (CLOSENSOCKET OLDVALUE T]   ; Close socket if we abort out of SPP.OPEN
             (COND
                ([COND
                    [(NULL HOST)                                 ; Server connection
                     (LET [(SERVERFN (LISTGET PROPS 'SERVER.FUNCTION]
                          (COND
                             (SERVERFN                           ; Handler for each of multiple possible connections to this server
                                                                 ; socket
                                    (replace (SPPCON SPPSERVERFLAG) of CON with T)
                                    (replace (SPPCON SPPSERVERFN) of CON with SERVERFN)
                                    T)
                             (T                                  ; Wait for single user to connect, then return it
                                (until (fetch (SPPCON SPPESTABLISHEDP) of CON)
                                   do (MONITOR.AWAIT.EVENT (fetch (SPPCON SPPLOCK) of CON)
                                                           (fetch (SPPCON SPPINPUTEVENT) of CON)))
                                T]
                    ((OR (fetch (SPPCON SPPESTABLISHEDP) of CON)
                         (NOT PROBEP))                           ; User connection
                     T)
                    (T (\SPP.PROBE CON)
                       (bind (TIMER _ (SETUPTIMER (IMAX (TIMES 3000 (OR (NSNET.DISTANCE (fetch (SPPCON
                                                                                                SPPDESTNSNET
                                                                                                )
                                                                                        of CON))
                                                                        4))
                                                        SPP.USER.TIMEOUT)))
```

```
                    do (COND
                           ((fetch (SPPCON SPPESTABLISHEDP) of CON)
                            (RETURN T))
                           ((TIMEREXPIRED? TIMER)           ; We've waited long enough without response.  Wait period
                                                            ; based on hop count. Kill the watcher and get out of here.
                            (replace (SPPCON SPPTERMINATEDP) of CON with T)
                            (RELEASE.MONITORLOCK (fetch (SPPCON SPPLOCK) of CON))
                                                            ; So that watcher will be able to run
                            (WAKE.PROCESS (fetch (SPPCON SPPPROCESS) of CON))
                            (BLOCK)                         ; Give watcher a chance to clean up.
                            (RETURN NIL))
                           ((fetch (SPPCON SPPTERMINATEDP) of CON)
                                                            ; It died quickly?  Probably no such socket
                            (RETURN NIL))
                           (T (MONITOR.AWAIT.EVENT (fetch (SPPCON SPPLOCK) of CON)
                                     (fetch (SPPCON SPPINPUTEVENT) of CON)
                                    TIMER T]
                ;; CON is okay to use -- either established, or willing to be
                (for TAIL on PROPS by (CDDR TAIL) do (SELECTQ (CAR TAIL)
                                                             (CLOSEFN (replace (SPPCON SPPWHENCLOSEDFN) of CON
                                                                        with (CADR TAIL)))
                                                             (ATTENTIONFN (replace (SPPCON SPPATTENTIONFN)
                                                                            of CON with (CADR TAIL)))
                                                             (ERRORHANDLER (replace (SPPCON SPPERRORHANDLER)
                                                                             of CON with (CADR TAIL)))
                                                             (EOM.ON.FORCEOUT
                                                                 (replace (SPPCON SPPEOMONFORCEOUT) of CON
                                                                    with (CADR TAIL)))
                                                             (OTHERXIPHANDLER
                                                                 [COND
                                                                    ((FNTYP (CADR TAIL))
                                                                     (replace (SPPCON SPPOTHERXIPHANDLER)
                                                                        of CON with (CADR TAIL])
                                                            NIL))
                (\SPP.CREATE.STREAMS CON])])
```

(\SPP.CREATE.STREAM

```
  [LAMBDA (ACCESS)                                          (* bvm%: "12-Oct-84 22:43")
    (create STREAM
            DEVICE _ \SPPDEVICE
            ACCESS _ ACCESS])
```

(SPP.DESTADDRESS

```
  [LAMBDA (STREAM)                                          ; Edited 26-May-91 12:04 by jds
    (PROG ([CON (COND
                   ((type? SPPCON STREAM)
                    STREAM)
                   (T (GETSPPCON STREAM]
           (ADDRESS (create NSADDRESS)))
          (\BLT ADDRESS (LOCF (fetch (SPPCON SPPDESTNSADDRESS0) of CON))
                \#WDS.NSADDRESS)
          (RETURN ADDRESS])
```

(SPPOUTPUTSTREAM

```
  [LAMBDA (SPPINPUTSTREAM)                                  ; Edited 26-May-91 12:04 by jds
    (LET ((CON (GETSPPCON SPPINPUTSTREAM)))
         (OR (AND CON (fetch (SPPCON SPPOUTPUTSTREAM) of CON))
             (\SPP.STREAM.LOST SPPINPUTSTREAM])
```

(SPP.OPENP

```
  [LAMBDA (STREAM)                                          ; Edited 26-May-91 12:04 by jds
    (PROG (CON)
          (RETURN (AND STREAM (SPPSTREAMP STREAM)
                       (SETQ CON (GETSPPCON STREAM))
                       (NOT (fetch (SPPCON SPPTERMINATEDP) of CON])
```

(\STREAM.FROM.PACKET

```
  [LAMBDA (EPKT)                                            ; Edited 26-May-91 12:22 by jds
    ;; Return a stream which will read out of the contents of a single Packet Exchange packet.
    (CHECK (EQP (fetch (XIP XIPTYPE) of EPKT)
                \XIPT.EXCHANGE))
    (\MAKEBASEBYTESTREAM (fetch PACKETEXCHANGEBODY of EPKT)
           0
           (IDIFFERENCE (fetch (XIP XIPLENGTH) of EPKT)
                  (CONSTANT (IPLUS \XIPOVLEN 6)))
           'INPUT])
```

(SPP.FORCEOUTPUT

```
  [LAMBDA (STREAM)                                          ; Edited 26-May-91 12:22 by jds
```

```
    (PROG ((CON (GETSPPCON STREAM))
           EPKT)
          (COND
             ((SETQ EPKT (fetch (SPPCON SPPOUTPKT) of CON))
              [COND
                 ((EQ STREAM (fetch (SPPCON SPPINPUTSTREAM) of CON))
                  (SETQ STREAM (fetch (SPPCON SPPOUTPUTSTREAM) of CON]
              (UNINTERRUPTABLY
                   (add (fetch (XIP XIPLENGTH) of EPKT)
                        (fetch (STREAM COFFSET) of STREAM))
                   (\SPPINCFILEPTR STREAM (fetch (STREAM COFFSET) of STREAM))
                   (replace (SPPCON SPPOUTPKT) of CON with NIL)
                   (replace (STREAM CBUFMAXSIZE) of STREAM with 0)
                   (replace (STREAM COFFSET) of STREAM with 0)
                   (replace (STREAM CBUFPTR) of STREAM with NIL))
              (COND
                 ((fetch (SPPCON SPPEOMONFORCEOUT) of CON)
                  (replace (SPPXIP ENDOFMESSAGE) of EPKT with T)))
              (COND
                 ((fetch (SPPCON SPPOUTPUTABORTEDP) of CON)
                  (replace (SPPCON SPPOUTPUTABORTEDP) of CON with NIL)
                  (APPLY* (fetch (SPPCON SPPOUTPUTABORTEDFN) of CON)
                          STREAM))
                 ((NOT (\SENDSPP CON EPKT))
                  (\SPP.STREAM.LOST STREAM])
```

(SPP.FLUSH.TO.EOF

```
  [LAMBDA (INSTREAM)                                        ; Edited 26-May-91 12:19 by jds
    (while (NOT (\SPP.PREPARE.INPUT INSTREAM)) do (replace (STREAM COFFSET) of INSTREAM with (fetch (STREAM CBUFSIZE
                                                                                                      )
                                                                                                of INSTREAM))
       finally (RETURN (SELECTC (fetch (SPPSTREAM SPPEOFBITS) of INSTREAM)
                                (\SPPFLAG.EOM (replace (SPPSTREAM SPPEOFP) of INSTREAM with NIL)
                                              'EOM)
                                (\SPPFLAG.ATTENTION
                                     (SPP.CLEARATTENTION INSTREAM)
                                     (BIN INSTREAM))
                                (\SPPFLAG.END 'EOF)
                                NIL])
```

(SPP.SENDEOM

```
  [LAMBDA (STREAM)                                          ; Edited 26-May-91 12:04 by jds
    ;; Send the End of Message indication.
    (PROG ((CON (GETSPPCON STREAM))
           EPKT)
          (OR (WRITEABLE STREAM)
              (SETQ STREAM (fetch (SPPCON SPPOUTPUTSTREAM) of CON))
              (\SPP.STREAM.LOST STREAM))
          (replace (SPPXIP ENDOFMESSAGE) of (OR (fetch (SPPCON SPPOUTPKT) of CON)
                                                (\SPP.PREPARE.OUTPUT STREAM CON)
                                                (\SPP.STREAM.LOST STREAM))
             with T)
          (SPP.FORCEOUTPUT STREAM])
```

(SPP.CLEAREOM

```
  [LAMBDA (STREAM NOERRORFLG)                               ; Edited 26-May-91 12:19 by jds
    (PROG ((CON (GETSPPCON STREAM))
           FLG)
          (RETURN (COND
                     ((AND (\SPP.PREPARE.INPUT STREAM)
                           (EQ (fetch (SPPSTREAM SPPEOFBITS) of STREAM)
                               \SPPFLAG.EOM))
                      (replace (SPPSTREAM SPPEOFP) of STREAM with NIL)
                      T)
                     ((NOT NOERRORFLG)
                      (ERROR "SPP.CLEAREOM - not at end of message" STREAM])
```

(SPP.SENDATTENTION

```
  [LAMBDA (STREAM ATTENTIONBYTE CC)                         ; Edited 26-May-91 12:22 by jds
    ;; Send an Attention packet with the specified data byte and control bits.  Can't use normal stream mechanism because stream may be read only.
    (PROG ((CON (GETSPPCON STREAM))
           EPKT)
          [SETQ EPKT (\FILLINSPP CON (LOGOR \SPPHEAD.CC.ATTENTION (OR CC 0]
          (replace (SPPXIP FIRSTSPPDATABYTE) of EPKT with ATTENTIONBYTE)
          (add (fetch (XIP XIPLENGTH) of EPKT)
               1)
          (RETURN (\SENDSPP CON EPKT T])
```

(SPP.CLEARATTENTION

```
  [LAMBDA (STREAM NOERRORFLG)                               ; Edited 26-May-91 12:20 by jds
```

```
    (PROG ((CON (GETSPPCON STREAM))
           FLG)
          (RETURN (COND
                      ((AND (\SPP.PREPARE.INPUT STREAM)
                            (EQ (fetch (SPPSTREAM SPPEOFBITS) of STREAM)
                                \SPPFLAG.ATTENTION))
                       (UNINTERRUPTABLY
                           (replace (SPPSTREAM SPPEOFP) of STREAM with NIL)
                           (replace (STREAM CBUFSIZE) of STREAM with 1))
                       T)
                      ((NOT NOERRORFLG)
                       (ERROR "SPP.CLEARATTENTION - not at attention packet" STREAM])
```

(SPP.CLOSE

```
  [LAMBDA (STREAM ABORT?)                                   ; Edited 26-May-91 12:10 by jds
```

;; Close an SPP stream. Don't close it if there's still an open Bulk Data stream, unless the user is aborting the connection.

```
    (PROG (CON SUBSTREAM)
          (RETURN (COND
                      ((OR (NULL STREAM)
                           (NULL (SETQ CON (GETSPPCON STREAM)))
                           (fetch (SPPCON SPPTERMINATEDP) of CON))
                       NIL)
                      (T [COND
                            ((AND (SETQ SUBSTREAM (fetch (SPPCON SPPSUBSTREAM) of CON))
                                  (OPENED SUBSTREAM))           ; This connection still has an active bulk data stream.  Must want
                                                                ; to abort it
                             (\BULK.DATA.CLOSE SUBSTREAM (SETQ ABORT? T]
                         (COND
                            ((NOT ABORT?)
                             (SPP.FORCEOUTPUT STREAM)))
                         (\TERMINATESPP CON])
```

(\SPP.CLOSE.IF.ERROR

```
  [LAMBDA (STREAM)                                          (* bvm%: "16-NOV-83 14:57")
    (COND
       (RESETSTATE (SPP.CLOSE STREAM T])
```

(\SPP.RESETCLOSE

```
  [LAMBDA (STREAM)                                          (* bvm%: "16-NOV-83 14:59")
```

;;; For use in RESETSAVE -- sets the abort arg to SPP.CLOSE according to RESETSTATE

```
    (SPP.CLOSE STREAM RESETSTATE])
```

(SPP.BACKFILEPTR

```
  [LAMBDA (STREAM)                                          ; Edited 26-May-91 12:01 by jds
    (if (NEQ (fetch (STREAM COFFSET) of STREAM)
             0)
        then (add (fetch (STREAM COFFSET) of STREAM)
                  -1])
```

(\SPP.GETFILEPTR

```
  [LAMBDA (STREAM)                                          ; Edited 26-May-91 12:20 by jds
    (IPLUS (fetch (SPPSTREAM SPPFILEPTR) of STREAM)
           (fetch (STREAM COFFSET) of STREAM])
```

(\SPP.SETFILEPTR

```
  [LAMBDA (STREAM INDX)                                     ; Edited 26-May-91 12:01 by jds
    (PROG ((CON (GETSPPCON STREAM))
           SKIPBYTES)
          (RETURN (COND
                      ((AND (EQ (fetch (STREAM ACCESSBITS) of STREAM)
                                ReadBit)
                            (IGEQ (SETQ SKIPBYTES (IDIFFERENCE INDX (\SPP.GETFILEPTR STREAM)))
                                  0))                           ; Can only move file pointer on input, and then only forward
                       (\SPP.SKIPBYTES STREAM SKIPBYTES))
                      (T (\IS.NOT.RANDACCESSP STREAM])
```

(\SPP.SKIPBYTES

```
  [LAMBDA (STREAM NBYTES)                                   ; Edited 26-May-91 12:01 by jds
    (PROG (BYTESLEFT CONDITION)
      LP  [COND
             ((SETQ CONDITION (\SPP.PREPARE.INPUT STREAM))
              (COND
                 ((NEQ (SETQ CONDITION (SPP.STREAM.ERROR STREAM CONDITION))
                       T)
                  (RETURN CONDITION]
          (COND
             ([IGREATERP NBYTES (SETQ BYTESLEFT (IDIFFERENCE (fetch (STREAM CBUFSIZE) of STREAM)
```

```
                                                    (fetch (STREAM COFFSET) of STREAM]
            (SETQ NBYTES (IDIFFERENCE NBYTES BYTESLEFT))
            (replace (STREAM COFFSET) of STREAM with (fetch (STREAM CBUFSIZE) of STREAM))
            (GO LP))
           (T (add (fetch (STREAM COFFSET) of STREAM)
                   NBYTES])
```

(\SPP.BOUTS

```
[LAMBDA (STREAM BASE OFF NBYTES)                                    ; Edited 26-May-91 12:10 by jds
  (PROG ((CON (GETSPPCON STREAM)))
        (RETURN (\BUFFERED.BOUTS (OR (COND
                                        ((NULL CON)
                                         NIL)
                                        ((EQ STREAM (fetch (SPPCON SPPINPUTSTREAM) of CON))
                                         (fetch (SPPCON SPPOUTPUTSTREAM) of CON))
                                        (T STREAM))
                                     (RETURN (\SPP.STREAM.LOST STREAM)))
                       BASE OFF NBYTES])
```

(\SPP.OTHER.BOUT

```
[LAMBDA (STREAM BYTE)                                               (* bvm%: "31-Jan-86 16:49")
```

;; BOUT function for the input side of an SPP connection, in case someone doesn't want to bother with SPPOUTPUTSTREAM

```
  (\BOUT (OR (SPPOUTPUTSTREAM STREAM)
             (\SPP.STREAM.LOST STREAM))
         BYTE])
```

(\SPP.GETNEXTBUFFER

```
[LAMBDA (STREAM WHATFOR NOERRORFLG)                                 ; Edited 26-May-91 12:10 by jds
```

;;; Generic buffer refiller for SPP streams

```
  (PROG (CON ERRCODE)
        (RETURN (SELECTQ WHATFOR
                    (READ (COND
                              ((NULL (SETQ ERRCODE (\SPP.PREPARE.INPUT STREAM)))
                               T)
                              ((OR (NEQ ERRCODE 'EOM)
                                   (NULL NOERRORFLG))
                               (SPP.STREAM.ERROR STREAM ERRCODE))))
                    (WRITE (SETQ CON (GETSPPCON STREAM))
                           (COND
                              ((\SPP.PREPARE.OUTPUT (COND
                                                       ((EQ STREAM (fetch (SPPCON SPPINPUTSTREAM) of CON))
                                                        (ffetch (SPPCON SPPOUTPUTSTREAM) of CON))
                                                       (T STREAM))
                                                    CON)
                               T)
                              (T                                    ; If that returned, then client must want no error
                                 (RETFROM (OR (STKPOS '\BUFFERED.BOUT)
                                              (STKPOS '\BUFFERED.BOUTS)
                                              (RETURN (\SPP.STREAM.LOST STREAM)))
                                          NIL T))))
                    (SHOULDNT])
```

(\SPP.STREAM.LOST

```
[LAMBDA (STREAM)                                                    (* bvm%: "31-Jan-86 16:47")
  (SPP.STREAM.ERROR STREAM 'STREAM.LOST])
```

(\SPP.DEFAULT.ERRORHANDLER

```
[LAMBDA (STREAM CONDITION)                                          ; Edited 26-May-91 12:01 by jds
  (SELECTQ CONDITION
       (STREAM.LOST (ERROR "Connection lost" (OR (fetch (STREAM FULLFILENAME) of STREAM)
                                                 STREAM)))
       (ATTENTION [LET ((CON (GETSPPCON STREAM)))
                       (COND
                          ((AND CON (EQ (fetch (SPPCON SPPINPUTDSTYPE) of CON)
                                        \SPPDSTYPE.BULKDATA))   ; Bulk data abort
                           (\COURIER.OUTPUT.ABORTED STREAM))
                          (T (\EOF.ACTION STREAM]))
       (\EOF.ACTION STREAM])
```

(\SPP.PREPARE.INPUT

```
[LAMBDA (STREAM TIMEOUT)                                            ; Edited 26-May-91 12:22 by jds
```

;;; Gets the next input packet for the stream interface. If OK, returns NIL, otherwise returns the error condition as one of the canonical error codes, or
;;; one of the SPP-specific error codes

```
  (PROG ((CON (GETSPPCON STREAM))
         EPKT CONDITION OLD.DSTYPE NEW.DSTYPE SPPDSTYPECHANGEFN)
        (SETQ EPKT (fetch (SPPCON SPPINPKT) of CON))
```

```
      CHECK-CURRENT
           (COND
              (EPKT                                          ; Look at previous packet to make sure we're not trying to read
                                                             ; past the end of the stream.
                     (COND
                        ((ILESSP (fetch (STREAM COFFSET) of STREAM)
                                 (fetch (STREAM CBUFSIZE) of STREAM))
                                                             ; Not finished with this packet yet
                         (RETURN NIL)))
                     [COND
                        ((EQ (fetch (SPPSTREAM SPPEOFBITS) of STREAM)
                             \SPPFLAG.ATTENTION)              ; Waiting to read attention packet.  Has to be cleared first, so
                                                             ; indicate eof now
                         (RETURN 'ATTENTION]
                     ;; Throw away the previous packet in preparation for the next one.
                     (UNINTERRUPTABLY
                         (\SPPINCFILEPTR STREAM (fetch (STREAM CBUFSIZE) of STREAM))
                         (replace (STREAM COFFSET) of STREAM with (replace (STREAM CBUFSIZE) of STREAM with 0))
                         (replace (SPPCON SPPINPKT) of CON with NIL)
                         (replace (STREAM CBUFPTR) of STREAM with NIL)
                         [COND
                            ((fetch (SPPXIP EOMP) of EPKT)
                             (replace (SPPSTREAM SPPEOFP) of STREAM with 'EOM])
                     (RELEASE.XIP EPKT)))
           (COND
              ((SETQ CONDITION (fetch (SPPSTREAM SPPEOFP) of STREAM))
               (RETURN CONDITION)))
      AGAIN
           (SETQ EPKT (\GETSPP CON TIMEOUT))
           [COND
              ((NULL EPKT)
               (RETURN (COND
                           (TIMEOUT 'BIN.TIMEOUT)
                           (T 'STREAM.LOST]
           (SELECTC (SETQ NEW.DSTYPE (fetch (SPPXIP DSTYPE) of EPKT))
                ((LIST \SPPDSTYPE.END \SPPDSTYPE.ENDREPLY)
                     (replace (SPPSTREAM SPPEOFP) of STREAM with 'END)
                     (RETURN 'END))
                (\SPPDSTYPE.BULKDATA
                     (COND
                        ((NULL (fetch (SPPSTREAM BULK.DATA.CONTINUATION) of STREAM))
                         ;; We got a Bulk Data packet but not on a Bulk Data stream.  It's probably a straggler after we aborted a transfer, so
                         ;; ignore it.
                         (GO AGAIN))))
                NIL)
           (UNINTERRUPTABLY
               (replace (STREAM CBUFPTR) of STREAM with (fetch (SPPXIP SPPCONTENTS) of EPKT))
               (replace (STREAM COFFSET) of STREAM with 0)
               [replace (STREAM CBUFSIZE) of STREAM with (COND
                                                             ((fetch (SPPXIP ATTENTION) of EPKT)
                                                                    ; Not readable yet
                                                              (replace (SPPSTREAM SPPEOFP) of STREAM
                                                                 with 'ATTENTION)
                                                              0)
                                                             (T (IDIFFERENCE (fetch (XIP XIPLENGTH) of EPKT)
                                                                       (CONSTANT (IPLUS \XIPOVLEN \SPPHEAD.LENGTH]
               (replace (SPPCON SPPINPKT) of CON with EPKT))
           (SETQ OLD.DSTYPE (fetch (SPPCON SPPINPUTDSTYPE) of CON))
           (replace (SPPCON SPPINPUTDSTYPE) of CON with NEW.DSTYPE)
           (COND
              ((AND (NEQ OLD.DSTYPE NEW.DSTYPE)
                    (SETQ SPPDSTYPECHANGEFN (fetch (SPPCON SPPDSTYPECHANGEFN) of CON)))
               (CL:FUNCALL SPPDSTYPECHANGEFN STREAM OLD.DSTYPE NEW.DSTYPE)))
           (GO CHECK-CURRENT)                                ; Finally, loop back to top in case new packet is empty or
                                                             ; otherwise unusual

      ])
```

(\SPP.PREPARE.OUTPUT

```
  [LAMBDA (STREAM CON)                                       ; Edited 26-May-91 12:10 by jds
    ;; Fill in a new packet for the output side of the stream interface.
    (SPP.FORCEOUTPUT STREAM)
    (if (NOT (fetch (SPPCON SPPTERMINATEDP) of CON))
        then (PROG ((EPKT (\FILLINSPP CON)))
                   (replace (SPPCON SPPOUTPKT) of CON with EPKT)
                   (replace (STREAM CBUFPTR) of STREAM with (fetch (SPPXIP SPPCONTENTS) of EPKT))
                   (replace (STREAM COFFSET) of STREAM with 0)
                   (replace (STREAM CBUFMAXSIZE) of STREAM with (IDIFFERENCE \MAX.XIPDATALENGTH \SPPHEAD.LENGTH))
                   (RETURN EPKT])
```

(SPP.DSTYPE

```
  [LAMBDA (STREAM DSTYPE NOBLOCK)                            ; Edited 26-May-91 12:10 by jds
```

```
    ;; Get or set datastream type of current packet.
    (PROG ((CON (GETSPPCON STREAM))
           EPKT CONDITION)
          (RETURN (COND
                      (DSTYPE (COND
                                  ((SETQ EPKT (fetch (SPPCON SPPOUTPKT) of CON))
                                   (replace (SPPXIP DSTYPE) of EPKT with DSTYPE)))
                              (replace (SPPCON SPPDSTYPE) of CON with DSTYPE))
                      (T (COND
                             ((NOT (READABLE STREAM))
                              (fetch (SPPCON SPPDSTYPE) of CON))
                             (NOBLOCK (fetch (SPPCON SPPINPUTDSTYPE) of CON))
                             (T (fetch (SPPXIP DSTYPE) of (OR (fetch (SPPCON SPPINPKT) of CON)
                                                              (COND
                                                                 ((AND (SETQ CONDITION (\SPP.PREPARE.INPUT STREAM))
                                                                       (NEQ (SETQ CONDITION (SPP.STREAM.ERROR
                                                                                                 STREAM CONDITION
                                                                                                 ))
                                                                            T))
                                                                  (RETURN CONDITION))
                                                                 (T (fetch (SPPCON SPPINPKT) of CON])
```

(SPP.READP

```
  [LAMBDA (STREAM)                                          ; Edited 26-May-91 12:01 by jds
    (COND
       ((NOT (READABLE STREAM))
        (LISPERROR "FILE NOT OPEN" (FULLNAME STREAM)))
       ((ILESSP (fetch (STREAM COFFSET) of STREAM)
                (fetch (STREAM CBUFSIZE) of STREAM))
        T)
       (T (NULL (\SPP.PREPARE.INPUT STREAM 0])
```

(SPP.EOFP

```
  [LAMBDA (STREAM)                                          ; Edited 26-May-91 12:01 by jds
    (COND
       ((NOT (READABLE STREAM))
        T)
       ((ILESSP (fetch (STREAM COFFSET) of STREAM)
                (fetch (STREAM CBUFSIZE) of STREAM))
        NIL)
       (T (LET ((CONDITION (\SPP.PREPARE.INPUT STREAM)))
               (SELECTQ CONDITION
                    (NIL                                    ; There is more
                         NIL)
                    (END T)
                    (STREAM.LOST                            ; Harumph, can't say EOFP because there would have been
                                                            ; more
                                NIL)
                    (PROGN                                  ; Special kinds of EOF
                           CONDITION])

)

(DEFINEQ
```

(SPPSTREAMP

```
  [LAMBDA (STREAM)                                          ; Edited 13-Sep-90 16:18 by jds
    ;; Returns non-NIL if STREAM is an SPP stream.
    (type? SPPCON (GETSPPCON STREAM])

)

(DECLARE%: DONTEVAL@LOAD DOCOPY

(\INITSPP)
)

;; Debugging

(ADDTOVAR XIPPRINTMACROS (5 . PRINTSPP))

(DEFINEQ
```

(PPSPP

```
  [LAMBDA (CON FILE DETAILS)                                ; Edited 26-May-91 12:11 by jds
    (PROG (STR N)
          (SETQ FILE (\GETSTREAM FILE 'OUTPUT))
          (printout FILE "Local: " (fetch (SPPCON SPPSOURCENSADDRESS) of CON)
                 ", id = "
                 (fetch (SPPCON SPPSOURCEID) of CON)
                 T "Remote: " (fetch (SPPCON SPPDESTNSADDRESS) of CON)
                 ", id = "
                 (fetch (SPPCON SPPDESTID) of CON)
```

```
            T)
         [COND
            ((NOT (fetch (SPPCON SPPESTABLISHEDP) of CON))
             (printout FILE " [not established]"))
            (T (printout FILE "DS Type = " (SELECTC (fetch (SPPCON SPPDSTYPE) of CON)
                                                   (\SPPDSTYPE.COURIER
                                                       "courier")
                                                   (\SPPDSTYPE.BULKDATA
                                                       "bulkdata")
                                                   (fetch (SPPCON SPPDSTYPE) of CON]
         (COND
            ((fetch (SPPCON SPPTERMINATEDP) of CON)
             (printout FILE " [terminated]")))
         (COND
            ((fetch (SPPCON SPPACKREQUESTED) of CON)
             (printout FILE T "Ack requested: " .F3.1 (FQUOTIENT (CLOCKDIFFERENCE (fetch (SPPCON SPPACKREQTIME)
                                                                                        of CON))
                                                                                1000)
                    " secs ago")))
         (printout FILE T "Round trip: " .F4.1 (FQUOTIENT (fetch (SPPCON SPPROUNDTRIPTIME) of CON)
                                                          1000)
                " secs")
         (printout FILE T "Last input activity: " .F4.1 (FQUOTIENT (CLOCKDIFFERENCE (fetch (SPPCON
                                                                                              SPPACTIVITYTIMER
                                                                                               )
                                                                                     of CON))
                                                                   1000)
                " secs ago" T)
         (printout FILE T "Input:" T "  Seq# " (fetch (SPPCON SPPACKNO) of CON)
                T "  Allocation: " [\LOLOC (IDIFFERENCE (fetch (SPPCON SPPACCEPTNO) of CON)
                                                        (SUB1 (fetch (SPPCON SPPACKNO) of CON]
                T)
         (PPSPPSTREAM (fetch (SPPCON SPPINPUTSTREAM) of CON)
                FILE)
         (COND
            ((NEQ [SETQ N (IPLUS (COND
                                    ((fetch (SPPCON SPPINPKT) of CON)
                                     1)
                                    (T 0))
                                 (\QUEUELENGTH (fetch (SPPCON SPPINPUTQ) of CON]
                  0)
             (printout FILE "  Packets in queue: " N T)))
         (printout FILE T "Output:" T "  Seq# " (fetch (SPPCON SPPSEQNO) of CON))
         [COND
            ((EQ (fetch (SPPCON SPPSEQNO) of CON)
                 (fetch (SPPCON SPPACKEDSEQNO) of CON))
             (printout FILE ", all acked"))
            (T (printout FILE ", acked# " (fetch (SPPCON SPPACKEDSEQNO) of CON]
         (printout FILE T "  Allocation: " [\LOLOC (IDIFFERENCE (fetch (SPPCON SPPOUTPUTALLOCNO) of CON)
                                                                (SUB1 (fetch (SPPCON SPPSEQNO) of CON]
                T)
         (PPSPPSTREAM (fetch (SPPCON SPPOUTPUTSTREAM) of CON)
                FILE)
         (COND
            (DETAILS (printout FILE "  Awaiting ack: " %# [for (I _ (fetch (SPPCON SPPACKEDSEQNO) of CON))
                                                               by (SEQ.ADD1 I)
                                                               bind (NEXT _ (fetch (SPPCON SPPSEQNO) of CON))
                                                               while (SEQ.GREATERP NEXT I)
                                                               do (PRINTSPP (ELT (fetch (SPPCON SPPRETRANSMITQ)
                                                                                    of CON)
                                                                                 (RETRANSMITINDEX I]
                               T)))
         (COND
            ((SETQ STR (fetch (SPPCON SPPSUBSTREAM) of CON))
             (printout FILE T "Bulk data stream (" (fetch (STREAM ACCESS) of STR)
                    "):" T)
             (PPSPPSTREAM STR FILE])
```

(\SPP.INFO.HOOK

```
[LAMBDA (PROC BUTTON)                                          (* bvm%: "25-Sep-84 13:07")
   (DECLARE (USEDFREE SPPCON))
   ;; This is evaluated underneath \SPPWATCHER
   (if (EQ BUTTON 'MIDDLE)
       then                                                    ; all the details
           (INSPECT SPPCON)
     else (PROG [(WINDOW (PROCESSPROP PROC 'WINDOW]
                (COND
                   ((NULL WINDOW)
                    (SETQ WINDOW (CREATEW (GETBOXREGION 256 240)
                                          "SPP Connection Status"))
                    (DSPFONT (FONTCREATE 'GACHA 8)
                             WINDOW)
                    (PROCESSPROP PROC 'WINDOW WINDOW))
                   (T (CLEARW WINDOW)))
                (PPSPP SPPCON WINDOW])
```

(PPSPPSTREAM

```
[LAMBDA (STREAM FILE)                                   ; Edited 26-May-91 12:21 by jds
  (if STREAM
      then (printout FILE "  File pointer: " (\SPP.GETFILEPTR STREAM))
           (if (fetch (SPPSTREAM SPPEOFP) of STREAM)
               then (printout FILE " [eof]"))
           (TERPRI FILE])
```

(\SPP.CHECK.INPUT.QUEUE

```
[LAMBDA (CON)                                           ; Edited 26-May-91 12:11 by jds
  (PROG ((ACKNO (fetch (SPPCON SPPACKNO) of CON))
         (INQ (fetch (SPPCON SPPINPUTQ) of CON))
         N1 N2 CURRENT NEXT)                            ; Check consistency of input queue.
        (SETQ CURRENT (fetch SYSQUEUEHEAD of INQ))
    L   (COND
           ((NULL CURRENT)
            (RETURN T)))
        (SETQ N1 (fetch (SPPXIP SEQNO) of CURRENT))
        (COND
           ((EQ N1 ACKNO)
            (SHOULDNT "The input queue contains a packet that should have been acknowledged already.")
            (RETURN NIL)))
        (COND
           ((NULL (SETQ NEXT (fetch QLINK of CURRENT)))
            (RETURN T)))
        (SETQ N2 (fetch (SPPXIP SEQNO) of NEXT))
        (COND
           ((EQ N1 N2)
            (SHOULDNT "The input queue has duplicates.")
            (RETURN NIL)))
        (COND
           ((SEQ.GREATERP N1 N2)
            (SHOULDNT "The input queue is out of order.")
            (RETURN NIL)))
        (SETQ CURRENT NEXT)
        (GO L])
```

(PRINTSPP

```
[LAMBDA (EPKT FILE)                                     ; Edited 26-May-91 12:23 by jds
  (PROG ((BASE (fetch (XIP XIPCONTENTS) of EPKT))
         SYSTEMP DS LENGTH)
        (printout FILE (fetch (SPPHEAD SOURCECONID) of BASE)
               "/"
               (fetch (SPPHEAD DESTCONID) of BASE))
        [COND
           ((NEQ (fetch (SPPHEAD CC) of BASE)
                 0)
            (PROG ((SEPR " [")
                   (COMMA ", "))
                  (COND
                     ((fetch (SPPHEAD SYSTEMPACKET) of BASE)
                      (printout FILE SEPR "sys")
                      (SETQ SEPR COMMA)
                      (SETQ SYSTEMP T)))
                  (COND
                     ((fetch (SPPHEAD SENDACK) of BASE)
                      (printout FILE SEPR "ack")
                      (SETQ SEPR COMMA)))
                  (COND
                     ((fetch (SPPHEAD ATTENTION) of BASE)
                      (printout FILE SEPR "attn")
                      (SETQ SEPR COMMA)))
                  (COND
                     ((fetch (SPPHEAD ENDOFMESSAGE) of BASE)
                      (printout FILE SEPR "eom")
                      (SETQ SEPR COMMA)))
                  (COND
                     ((NEQ SEPR COMMA)
                      (printout FILE SEPR "??")))
                  (printout FILE "]"]
        (printout FILE %, (SELECTC (SETQ DS (fetch (SPPHEAD DSTYPE) of BASE))
                                   (\SPPDSTYPE.COURIER
                                       "courier")
                                   (\SPPDSTYPE.BULKDATA
                                       "bulkdata")
                                   (\SPPDSTYPE.END
                                       "end")
                                   (\SPPDSTYPE.ENDREPLY
                                       "end-reply")
                                   DS))
        (printout FILE " seq " (fetch (SPPHEAD SEQNO) of BASE)
               "; ack/alloc = "
               (fetch (SPPHEAD ACKNO) of BASE)
```

```
                "/"
                (fetch (SPPHEAD ALLOCNO) of BASE))
          [COND
             ([NEQ 0 (SETQ LENGTH (IDIFFERENCE (fetch (XIP XIPLENGTH) of EPKT)
                                               (CONSTANT (IPLUS \XIPOVLEN \SPPHEAD.LENGTH]
              (printout FILE "; " LENGTH " bytes")
              (COND
                 (PRINTSPPDATAFLG (printout FILE T "Data: ")
                        (PRINTPACKETDATA (fetch (SPPHEAD SPPCONTENTS) of BASE)
                               0
                               '(CHARS)
                               LENGTH FILE]
          (printout FILE T T])

)

(RPAQ? PRINTSPPDATAFLG )

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS PRINTSPPDATAFLG)
)

(PUTPROPS SPP COPYRIGHT ("Xerox Corporation" 1983 1984 1985 1986 1987 1988 1990 1991 1993))
```

FUNCTION INDEX

VARIABLE INDEX

MACRO INDEX