File created: 12-Jun-90 17:44:14 **{DSK}<usr>local>lde>lispcore>library>TCPTFTP.;2**

changes to: (VARS TCPTFTPCOMS)

previous date: 1-Jul-87 10:52:03 {DSK}<usr>local>lde>lispcore>library>TCPTFTP.;1

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;


```
(RPAQQ TCPTFTPCOMS
       ((COMS ;; Trivial File Transfer Protocol
              (INITVARS (\TFTP.DEVICE)
                     (TFTP.MAXRETRIES 20))
              (GLOBALVARS \TFTP.DEVICE TFTP.MAXRETRIES)
              (DECLARE%: EVAL@COMPILE DONTCOPY (EXPORT (RECORDS TFTPCON TFTP TFTPSTREAM)
                                                       (CONSTANTS (\TFTPOVLEN 4)
                                                              (\TFTP.SOCKET 69))
                                                       (CONSTANTS * TFTPOPCODES)))
              (INITVARS (TFTP.MAXRETRIES 20))
              (FNS \TFTP.ACKNOWLEDGE \TFTP.CLOSEFILE \TFTP.EOFP \TFTP.ERROR \TFTP.GETNEXTBUFFER \TFTP.INIT
                   \TFTP.INPUT.BUFFER \TFTP.OPENFILE \TFTP.READP \TFTP.SEND.ERROR \TFTP.SETUP)
              (FILES (SYSLOAD)
                     TCPUDP))
        (COMS ;; TFTP Server functions
              (INITVARS (\TFTP.SERVER.CONNECTIONS))
              (GLOBALVARS \TFTP.SERVER.CONNECTIONS)
              (FNS TFTP.SERVER.PROCESS \TFTP.GET.FILE \TFTP.SEND.FILE))
        (COMS ;; User functions
              (FNS TFTP.SERVER TFTP.GET TFTP.PUT))
        (COMS ;; Tracing functions
              (FNS PRINTTFTP \TFTP.PRINT.ACK \TFTP.PRINT.DATA \TFTP.PRINT.ERROR \TFTP.PRINT.REQUEST))
        (P (\TFTP.INIT))))
```

;; Trivial File Transfer Protocol

```
(RPAQ? \TFTP.DEVICE )

(RPAQ? TFTP.MAXRETRIES 20)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \TFTP.DEVICE TFTP.MAXRETRIES)
)

(DECLARE%: EVAL@COMPILE DONTCOPY
```

;; FOLLOWING DEFINITIONS EXPORTED

```
(DECLARE%: EVAL@COMPILE

(RECORD TFTPCON (UDPSOCKET DESTSOCKET STREAM HOST))

[ACCESSFNS TFTP ((TFTPBASE (fetch (UDP UDPCONTENTS) of DATUM)))
       (BLOCKRECORD TFTPBASE ((OPCODE WORD)
                              (BLOCK# WORD)))
       [ACCESSFNS TFTP ((TFTPCONTENTS (\ADDBASE (fetch (UDP UDPCONTENTS) of DATUM)
                                                (FOLDHI \TFTPOVLEN BYTESPERWORD]
       (BLOCKRECORD TFTPBASE ((NIL WORD)
                              (ERRORCODE WORD]

[ACCESSFNS TFTPSTREAM ((TFTPCON (fetch (STREAM F1) of DATUM)
                                (replace (STREAM F1) of DATUM with NEWVALUE))
                       (LASTPACKETIN (fetch (STREAM F2) of DATUM)
                                (replace (STREAM F2) of DATUM with NEWVALUE]
)

(DECLARE%: EVAL@COMPILE

(RPAQQ \TFTPOVLEN 4)

(RPAQQ \TFTP.SOCKET 69)

(CONSTANTS (\TFTPOVLEN 4)
       (\TFTP.SOCKET 69))
)
```

```
(RPAQQ TFTPOPCODES ((\TFTP.RRQ 1)
                    (\TFTP.WRQ 2)
                    (\TFTP.DATA 3)
                    (\TFTP.ACK 4)
                    (\TFTP.ERROR 5)))

(DECLARE%: EVAL@COMPILE

(RPAQQ \TFTP.RRQ 1)

(RPAQQ \TFTP.WRQ 2)

(RPAQQ \TFTP.DATA 3)

(RPAQQ \TFTP.ACK 4)

(RPAQQ \TFTP.ERROR 5)

(CONSTANTS (\TFTP.RRQ 1)
       (\TFTP.WRQ 2)
       (\TFTP.DATA 3)
       (\TFTP.ACK 4)
       (\TFTP.ERROR 5))
)
)

;; END EXPORTED DEFINITIONS

(RPAQ? TFTP.MAXRETRIES 20)

(DEFINEQ
```

(\TFTP.ACKNOWLEDGE

```
  [LAMBDA (STREAM ACK#)                                          (* MPL " 2-Jun-85 17:07")
    (LET ((TFTPCON (fetch (TFTPSTREAM TFTPCON) of STREAM))
          (ACK (\ALLOCATE.ETHERPACKET)))
         (\TFTP.SETUP ACK TFTPCON \TFTP.ACK 'FREE)
         (UDP.APPEND.WORD ACK ACK#)
         (UDP.SEND (fetch (TFTPCON UDPSOCKET) of TFTPCON)
                ACK)
         (BLOCK)
         (COND
            ((AND (EQ (fetch (STREAM ACCESS) of STREAM)
                      'INPUT)
                  (fetch (TFTPSTREAM LASTPACKETIN) of STREAM))
             (UDP.CLOSE.SOCKET (fetch (TFTPCON UDPSOCKET) of TFTPCON])
```

(\TFTP.CLOSEFILE

```
  [LAMBDA (STREAM)                                               (* ejs%: " 9-Feb-85 23:47")
    (LET ((TFTPCON (fetch (TFTPSTREAM TFTPCON) of STREAM)))
         (SELECTQ (fetch (STREAM ACCESS) of STREAM)
             (OUTPUT [COND
                        ((AND (fetch (STREAM CBUFPTR) of STREAM)
                              (NOT (fetch (TFTPSTREAM LASTPACKETIN) of STREAM)))
                         (\TFTP.GETNEXTBUFFER STREAM 'WRITE])
             NIL)
         (UDP.CLOSE.SOCKET (fetch (TFTPCON UDPSOCKET) of TFTPCON)
                T)
         (replace (STREAM ACCESS) of STREAM with NIL)
         STREAM])
```

(\TFTP.EOFP

```
  [LAMBDA (STREAM)                                               (* ejs%: " 9-Feb-85 21:23")
    (OR (NULL (fetch (STREAM CBUFPTR) of STREAM))
        (AND (fetch (TFTPSTREAM LASTPACKETIN) of STREAM)
             (EQ (fetch (STREAM COFFSET) of STREAM)
                 (fetch (STREAM CBUFSIZE) of STREAM])
```

(\TFTP.ERROR

```
  [LAMBDA (TFTP TFTPCON)                                         (* ejs%: " 9-Feb-85 19:04")

          (* * Called upon receipt of error packet in TFTP stream)

    (LET [(ERRORSTRING (ALLOCSTRING (IDIFFERENCE (fetch (UDP UDPLENGTH) of TFTP)
                                                 (CONSTANT (IPLUS \UDPOVLEN (ADD1 \TFTPOVLEN]
         (\MOVEBYTES (fetch (TFTP TFTPCONTENTS) of TFTP)
                0
                (fetch (STRINGP BASE) of ERRORSTRING)
                (fetch (STRINGP OFFST) of ERRORSTRING)
                (fetch (STRINGP LENGTH) of ERRORSTRING))
         (ERROR (CONCAT "TFTP error message: " ERRORSTRING " for code")
                (fetch (TFTP ERRORCODE) of TFTP])
```

(\TFTP.GETNEXTBUFFER

```
[LAMBDA (STREAM WHATFOR NOERRORFLG)                    (* MPL " 2-Jun-85 19:48")
  (DECLARE (GLOBALVARS TFTP.MAXRETRIES))
  (LET* ((TFTPCON (fetch (TFTPSTREAM TFTPCON) of STREAM))
         (IPSOCKET (fetch (TFTPCON UDPSOCKET) of TFTPCON))
         (RETRYCOUNT 0)
         (BUFFER (fetch (STREAM CBUFPTR) of STREAM))
         UDP)
        (SELECTQ WHATFOR
            (READ [COND
                      [(fetch (TFTPSTREAM LASTPACKETIN) of STREAM)
                       (replace (STREAM ACCESS) of STREAM with NIL)
                       (\RELEASE.ETHERPACKET (fetch (STREAM CBUFPTR) of STREAM))
                       (replace (STREAM CBUFPTR) of STREAM with NIL)
                       (COND
                          (NOERRORFLG NIL)
                          (T (\EOF.ACTION STREAM]
                      (T (PROG [(NEXT# (COND
                                           (BUFFER (ADD1 (fetch (TFTP BLOCK#) of BUFFER)))
                                           (T 1]
                               LP  [for I from 1 to TFTP.MAXRETRIES until UDP do (SETQ UDP (UDP.GET IPSOCKET
                                                                                                    \ETHERTIMEOUT))
                                                                                 (COND
                                                                                    ((NOT UDP)
                                                                                     (\TFTP.ACKNOWLEDGE STREAM
                                                                                            (SUB1 NEXT#]
                                   (COND
                                      [UDP (COND
                                              [(EQ (fetch (TFTP OPCODE) of UDP)
                                                   \TFTP.DATA)
                                               (COND
                                                  ((IEQP (fetch (TFTP BLOCK#) of UDP)
                                                         NEXT#)
                                                   (\TFTP.INPUT.BUFFER STREAM UDP)
                                                   (\TFTP.ACKNOWLEDGE STREAM NEXT#)
                                                   (RETURN T))
                                                  [(ILESSP (fetch (TFTP BLOCK#) of UDP)
                                                           NEXT#)
                                                   [COND
                                                      (IPTRACEFLG (COND
                                                                     ((EQ IPTRACEFLG T)
                                                                      (printout IPTRACEFILE "Retransmitting ACK
                                                                             for block " (SUB1 NEXT#)
                                                                             T))
                                                                     (T (PRIN1 "R" IPTRACEFILE]
                                                   (\TFTP.ACKNOWLEDGE STREAM (SUB1 NEXT#))
                                                   (\RELEASE.ETHERPACKET UDP)
                                                   (SETQ UDP NIL)
                                                   (COND
                                                      ((EQ (add RETRYCOUNT 1)
                                                           TFTP.MAXRETRIES)
                                                       (\TFTP.SEND.ERROR TFTPCON 0 "Timeout awaiting next data
                                                              packet; aborting")
                                                       (replace (STREAM STRMBINFN) of STREAM
                                                          with (FUNCTION \STREAM.NOT.OPEN))
                                                       (ERROR "Timeout awaiting next data packet; aborting" STREAM
                                                              ))
                                                      (T (GO LP]
                                                  (T (\TFTP.SEND.ERROR TFTPCON 0 "Protocol error: Block # too
                                                           high.  Aborting...")
                                                     (replace (STREAM STRMBINFN) of STREAM
                                                        with (FUNCTION \STREAM.NOT.OPEN))
                                                     (ERROR "Protocol error: Block # too high.  Aborting..."
                                                            STREAM]
                                              ((EQ (fetch (TFTP OPCODE) of UDP)
                                                   \TFTP.ERROR)
                                               (replace (STREAM STRMBINFN) of STREAM with (FUNCTION STREAM.NOT.OPEN))
                                               (\TFTP.ERROR UDP TFTPCON))
                                              (T [\TFTP.SEND.ERROR TFTPCON 0
                                                         (CONCAT "Protocol error: Illegal TFTP opcode, expected
                                                                DATA but got " (SELECTC (fetch (TFTP OPCODE)
                                                                                               of UDP)
                                                                                        (\TFTP.RRQ "read request.")
                                                                                        (\TFTP.WRQ "write request.")
                                                                                        (\TFTP.ACK "ack.")
                                                                                        (CONCAT "unknown type "
                                                                                                (fetch (TFTP OPCODE)
                                                                                                   of UDP)
                                                                                                "."]
                                                 (replace (STREAM STRMBINFN) of STREAM with (FUNCTION
                                                                                             \STREAM.NOT.OPEN))
                                                 (ERROR "Illegal TFTP opcode rec'd" STREAM]
                                      (T (\TFTP.SEND.ERROR TFTPCON 0 "Timeout awaiting next data packet;
                                                aborting")
                                         (replace (STREAM STRMBINFN) of STREAM with (FUNCTION \STREAM.NOT.OPEN))
                                         (ERROR "Timeout awaiting next data packet; aborting" STREAM])
```

```
(WRITE [COND
          [(fetch (TFTPSTREAM LASTPACKETIN) of STREAM)
           (replace (STREAM ACCESS) of STREAM with NIL)
           (COND
              (NOERRORFLG NIL)
              (T (\EOF.ACTION STREAM]
          (T (PROG (ACK# NBYTES)
                   (SETQ ACK# (fetch (TFTP BLOCK#) of BUFFER))
                   (SETQ NBYTES (IDIFFERENCE (fetch (STREAM COFFSET) of STREAM)
                                             (UNFOLD (IDIFFERENCE (\LOLOC (fetch (TFTP TFTPCONTENTS)
                                                                                 of BUFFER))
                                                                  (\LOLOC BUFFER))
                                                     BYTESPERWORD)))
                   [replace (IP IPTOTALLENGTH) of BUFFER with (IPLUS NBYTES
                                                                     (CONSTANT (IPLUS \UDPOVLEN
                                                                                      \TFTPOVLEN
                                                                                      \IPOVLEN]
                   [replace (UDP UDPLENGTH) of BUFFER with (IPLUS NBYTES (CONSTANT (IPLUS \UDPOVLEN
                                                                                          \TFTPOVLEN
                                                                                          ]
                   (COND
                      ((ILESSP NBYTES 512)
                       (replace (TFTPSTREAM LASTPACKETIN) of STREAM with T)))
                LP (for I from 1 to TFTP.MAXRETRIES until UDP do (SETQ UDP (UDP.EXCHANGE IPSOCKET
                                                                                         BUFFER)))
                   (COND
                      [(AND UDP (EQ (fetch (TFTP OPCODE) of UDP)
                                    \TFTP.ACK))
                       (COND
                          ((EQ (fetch (TFTP BLOCK#) of UDP)
                               ACK#)
                           [COND
                              ((EQ NBYTES 512)
                               (\TFTP.SETUP UDP TFTPCON \TFTP.DATA NIL)
                               (UDP.APPEND.WORD UDP (ADD1 ACK#))
                               (replace (UDP UDPLENGTH) of UDP with (CONSTANT (IPLUS 512 \UDPOVLEN
                                                                                     \TFTPOVLEN)))
                               (\TFTP.INPUT.BUFFER STREAM UDP))
                              (T (replace (STREAM CBUFFPTR) of STREAM with NIL)
                                 (replace (STREAM ACCESS) of STREAM with NIL)
                                 (UDP.CLOSE.SOCKET (fetch (TFTPCON UDPSOCKET) of TFTPCON]
                           (RETURN T))
                          [(ILESSP (fetch (TFTP BLOCK#) of UDP)
                                   ACK#)
                           [COND
                              (IPTRACEFLG (COND
                                             ((EQ IPTRACEFLG T)
                                              (printout IPTRACEFILE "TFTP retransmission on
                                                     block# " ACK# T))
                                             (T (PRIN1 "R" IPTRACEFILE]
                           (\RELEASE.ETHERPACKET UDP)
                           (SETQ UDP NIL)
                           (COND
                              [(EQ (add RETRYCOUNT 1)
                                   TFTP.MAXRETRIES)
                               (\TFTP.SEND.ERROR TFTPCON 0 "Timeout awaiting acknowledgement.
                                      Aborting...")
                               (replace (STREAM STRMBOUTFN) of STREAM with (FUNCTION
                                                                              \STREAM.NOT.OPEN))
                               (COND
                                  (NOERRORFLG NIL)
                                  (T (\EOF.ACTION STREAM]
                              (T (GO LP]
                          (T (\TFTP.SEND.ERROR TFTPCON 0 "Protocol error: Block # too high.
                                    Aborting...")
                             (replace (STREAM STRMBOUTFN) of STREAM with (FUNCTION \STREAM.NOT.OPEN)
                                    )
                             (COND
                                (NOERRORFLG NIL)
                                (T (\EOF.ACTION STREAM]
                      ((AND UDP (EQ (fetch (TFTP OPCODE) of UDP)
                                    \TFTP.ERROR))
                       (replace (STREAM STRMBOUTFN) of STREAM with (FUNCTION \STREAM.NOT.OPEN))
                       (\TFTP.ERROR UDP TFTPCON))
                      [UDP [\TFTP.SEND.ERROR TFTPCON 0 (CONCAT "Protocol error: Illegal TFTP
                                                               opcode, expected ACK but got "
                                                               (SELECTC (fetch (TFTP OPCODE)
                                                                               of UDP)
                                                                    (\TFTP.RRQ "read request.")
                                                                    (\TFTP.WRQ "write request.")
                                                                    (\TFTP.DATA "data.")
                                                                    (CONCAT "unknown type "
                                                                            (fetch (TFTP OPCODE)
                                                                               of UDP)
                                                                            "."]
                           (replace (STREAM STRMBOUTFN) of STREAM with (FUNCTION \STREAM.NOT.OPEN))
                           (COND
```

```
                                        (NOERRORFLG NIL)
                                        (T (\EOF.ACTION STREAM]
                                  (T (\TFTP.SEND.ERROR TFTPCON 0 "Protocol error, aborting...")
                                     (replace (STREAM STRMBOUTFN) of STREAM with (FUNCTION \STREAM.NOT.OPEN))
                                     (COND
                                        (NOERRORFLG NIL)
                                        (T (\EOF.ACTION STREAM])
            (ERROR "Illegal ACCESS" WHATFOR])
```

(\TFTP.INIT

```
  [LAMBDA NIL                                                    (* ejs%: " 2-Feb-86 12:00")
    (DECLARE (GLOBALVARS \TFTP.DEVICE))
    (OR \TFTP.DEVICE
        (\DEFINEDEVICE NIL
               (SETQ \TFTP.DEVICE
                (create FDEV
                       FDBINABLE _ T
                       FDBOUTABLE _ T
                       NODIRECTORIES _ T
                       RESETABLE _ NIL
                       RANDOMACCESSP _ NIL
                       BUFFERED _ T
                       PAGEMAPPED _ NIL
                       DEVICENAME _ 'TFTP
                       HOSTNAMEP _ (FUNCTION NILL)
                       EVENTFN _ (FUNCTION NILL)
                       FORCEOUTPUT _ (FUNCTION NILL)
                       BIN _ (FUNCTION \BUFFERED.BIN)
                       BOUT _ (FUNCTION \BUFFERED.BOUT)
                       GETNEXTBUFFER _ (FUNCTION \TFTP.GETNEXTBUFFER)
                       READP _ (FUNCTION \TFTP.READP)
                       EOFP _ (FUNCTION \TFTP.EOFP)
                       CLOSEFILE _ (FUNCTION \TFTP.CLOSEFILE])
```

(\TFTP.INPUT.BUFFER

```
  [LAMBDA (STREAM UDP)                                           (* ejs%: " 9-Feb-85 20:51")

          (* * Sets up the fields of the stream necessary to support buffered operation, with UDP as the next packet)

    (LET [(OFFSET (UNFOLD (IDIFFERENCE (\LOLOC (fetch (TFTP TFTPCONTENTS) of UDP))
                                       (\LOLOC UDP))
                          BYTESPERWORD))
          (LENGTH (IDIFFERENCE (fetch (UDP UDPLENGTH) of UDP)
                               (CONSTANT (IPLUS \UDPOVLEN \TFTPOVLEN]
         [COND
            ((type? ETHERPACKET (fetch (STREAM CBUFPTR) of STREAM))
             (\RELEASE.ETHERPACKET (fetch (STREAM CBUFPTR) of STREAM]
         (replace (STREAM CBUFPTR) of STREAM with UDP)
         (replace (STREAM COFFSET) of STREAM with OFFSET)
         (replace (STREAM CBUFSIZE) of STREAM with (replace (STREAM CBUFMAXSIZE) of STREAM with (IPLUS OFFSET LENGTH)
                                                            ))
         (COND
            ((ILESSP LENGTH 512)
             (replace (TFTPSTREAM LASTPACKETIN) of STREAM with T])
```

(\TFTP.OPENFILE

```
  [LAMBDA (FILENAME ACCESS RECOG PARAMETERS)                     (* ejs%: "15-Sep-85 17:48")

          (* * Open a file using TFTP)

    (LET* ((HOSTNAME (FILENAMEFIELD FILENAME 'HOST))
           [DEVICE (COND
                      ((DODIP.HOSTP HOSTNAME)
                       (create FDEV using \TFTP.DEVICE DEVICENAME _ HOSTNAME))
                      (T (ERROR "Unknown IP host: " HOSTNAME]
           (STREAM (create STREAM
                           DEVICE _ DEVICE))
           [TFTPCON (replace (FDEV DEVICEINFO) of DEVICE with (create TFTPCON
                                                                      UDPSOCKET _ (UDP.OPEN.SOCKET)
                                                                      STREAM _ STREAM
                                                                      HOST _ (DODIP.HOSTP HOSTNAME]
           (UDP (\ALLOCATE.ETHERPACKET))
           UDPIN)
          (RESETLST
              (RESETSAVE NIL (LIST [FUNCTION (LAMBDA (SOCKET)
                                               (AND RESETSTATE (UDP.CLOSE.SOCKET SOCKET T]
                                   (fetch (TFTPCON UDPSOCKET) of TFTPCON)))
              (replace (TFTPCON DESTSOCKET) of TFTPCON with \TFTP.SOCKET)
              (\TFTP.SETUP UDP TFTPCON (SELECTQ ACCESS
                                           (INPUT \TFTP.RRQ)
                                           (OUTPUT \TFTP.WRQ)
                                           (ERROR "ACCESS must be INPUT or OUTPUT" ACCESS)))
              (UDP.APPEND.STRING UDP (SUBATOM FILENAME (STRPOS '} FILENAME NIL NIL NIL T)))
              (UDP.APPEND.BYTE UDP 0)
```

```
              (UDP.APPEND.STRING UDP (COND
                                        ((EQ (CADR (FASSOC 'TYPE PARAMETERS))
                                             'BINARY)
                                         "OCTET")
                                        (T "NETASCII")))
              (UDP.APPEND.BYTE UDP 0)
              (for I from 1 to \MAXETHERTRIES do (SETQ UDPIN (UDP.EXCHANGE (fetch (TFTPCON UDPSOCKET) of TFTPCON)
                                                                           UDP))
                 until UDPIN finally (\RELEASE.ETHERPACKET UDP))
              (COND
                 [UDPIN (SELECTC (fetch (TFTP OPCODE) of UDPIN)
                                (\TFTP.ACK (COND
                                              ((AND (EQ ACCESS 'OUTPUT)
                                                    (EQ (fetch (TFTP BLOCK#) of UDPIN)
                                                        0))
                                               (replace (TFTPSTREAM TFTPCON) of STREAM with TFTPCON)
                                               (replace (STREAM ACCESS) of STREAM with ACCESS)
                                               (replace (STREAM FULLFILENAME) of STREAM with FILENAME)
                                               (replace (TFTPCON DESTSOCKET) of TFTPCON with (fetch (UDP UDPSOURCEPORT)
                                                                                                    of UDPIN))
                                               (\TFTP.SETUP UDPIN TFTPCON \TFTP.DATA NIL)
                                               (UDP.APPEND.WORD UDPIN 1)
                                               (add (fetch (UDP UDPLENGTH) of UDPIN)
                                                    512)
                                               (\TFTP.INPUT.BUFFER STREAM UDPIN)
                                               STREAM)))
                                (\TFTP.DATA (COND
                                               ((AND (EQ ACCESS 'INPUT)
                                                     (EQ (fetch (TFTP BLOCK#) of UDPIN)
                                                         1))
                                                (replace (TFTPSTREAM TFTPCON) of STREAM with TFTPCON)
                                                (replace (STREAM ACCESS) of STREAM with ACCESS)
                                                (replace (STREAM FULLFILENAME) of STREAM with FILENAME)
                                                (replace (TFTPCON DESTSOCKET) of TFTPCON with (fetch (UDP UDPSOURCEPORT
                                                                                                     )
                                                                                                  of UDPIN))
                                                (\TFTP.INPUT.BUFFER STREAM UDPIN)
                                                (\TFTP.ACKNOWLEDGE STREAM 1)
                                                STREAM)))
                                (\TFTP.ERROR (\TFTP.ERROR UDPIN))
                                (ERROR "Unknown TFTP opcode" (fetch (TFTP OPCODE) of UDPIN]
                    (T (UDP.CLOSE.SOCKET (fetch (TFTPCON UDPSOCKET) of TFTPCON)
                              T)
                       NIL)))])
```

(\TFTP.READP

```
  [LAMBDA (STREAM)                                          (* ejs%: " 9-Feb-85 20:48")
    (ILESSP (fetch (STREAM COFFSET) of STREAM)
            (fetch (STREAM CBUFSIZE) of STREAM])
```

(\TFTP.SEND.ERROR

```
  [LAMBDA (TFTPCON ERRORCODE ERRORSTRING)                   (* ejs%: " 1-Jun-85 15:34")

          (* * Send an error back to the requestor)

    (LET ((TFTP (\ALLOCATE.ETHERPACKET)))
         (\TFTP.SETUP TFTP TFTPCON \TFTP.ERROR NIL)
         (UDP.APPEND.WORD TFTP ERRORCODE)
         (UDP.APPEND.STRING TFTP ERRORSTRING)
         (UDP.APPEND.BYTE TFTP 0)
         (UDP.SEND (fetch (TFTPCON UDPSOCKET) of TFTPCON)
                TFTP])
```

(\TFTP.SETUP

```
  [LAMBDA (UDP TFTPCON OPCODE REQUEUE)                      (* ejs%: " 9-Feb-85 20:32")
    (UDP.SETUP UDP (fetch (TFTPCON HOST) of TFTPCON)
           (fetch (TFTPCON DESTSOCKET) of TFTPCON)
           0
           (fetch (TFTPCON UDPSOCKET) of TFTPCON))
    (replace EPREQUEUE of UDP with REQUEUE)
    (UDP.APPEND.WORD UDP OPCODE])

)

(FILESLOAD (SYSLOAD)
       TCPUDP)
```

;; TFTP Server functions

```
(RPAQ? \TFTP.SERVER.CONNECTIONS )

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \TFTP.SERVER.CONNECTIONS)
```

)

(DEFINEQ

(TFTP.SERVER.PROCESS

```
  [LAMBDA (LOGSTREAM)                                    (* ejs%: " 3-Jun-85 01:52")

          (* * A server for TFTP file transfer)

    (DECLARE (GLOBALVARS \TFTP.SERVER.CONNECTIONS))
    (LET* ((DEVICE (create FDEV using \TFTP.DEVICE DEVICENAME _ 'TFTPSERVER))
           (SERVERSOCKET (UDP.OPEN.SOCKET \TFTP.SOCKET T))
           CONNECTION)
          [COND
             ((NULL LOGSTREAM)
              (COND
                 ((NOT (HASTTYWINDOWP))
                  (\CREATE.TTYDISPLAYSTREAM)))
              (SETQ LOGSTREAM (TTYDISPLAYSTREAM]
          (SETQ \TFTP.SERVER.CONNECTIONS NIL)
          (COND
             (SERVERSOCKET
              (RESETLST
                  (RESETSAVE NIL (LIST [FUNCTION (LAMBDA (SOCKET)
                                                   (UDP.CLOSE.SOCKET SOCKET T]
                                       SERVERSOCKET))
                  [while T
                     do (LET ((UDP (UDP.GET SERVERSOCKET T)))
                             (SETQ CONNECTION (CONS (fetch (IP IPSOURCEADDRESS) of UDP)
                                                    (fetch (UDP UDPSOURCEPORT) of UDP)))
                             (COND
                                [(NOT (MEMBER CONNECTION \TFTP.SERVER.CONNECTIONS))
                                 (push \TFTP.SERVER.CONNECTIONS CONNECTION)
                                 (SELECTC (fetch (TFTP OPCODE) of UDP)
                                      (\TFTP.RRQ (ADD.PROCESS `(\TFTP.SEND.FILE %, UDP
                                                                  (QUOTE %, (create TFTPCON UDPSOCKET _
                                                                                    (UDP.OPEN.SOCKET)))
                                                                  %, DEVICE %, LOGSTREAM)))
                                      (\TFTP.WRQ (ADD.PROCESS `(\TFTP.GET.FILE %, UDP (QUOTE %,
                                                                                      (create TFTPCON
                                                                                              UDPSOCKET _
                                                                                              (
                                                                                              UDP.OPEN.SOCKET
                                                                                               )))
                                                                  %, DEVICE %, LOGSTREAM)))
                                      (PROGN (printout LOGSTREAM "TFTP Server: Unexpected opcode "
                                                       (fetch (TFTP OPCODE) of UDP)
                                                       T)
                                             (SETQ \TFTP.SERVER.CONNECTIONS (DREMOVE CONNECTION
                                                                                    \TFTP.SERVER.CONNECTIONS))
                                             (\RELEASE.ETHERPACKET UDP]
                                (T                                  (* Duplicate request)
                                   (\RELEASE.ETHERPACKET UDP])])
```

(\TFTP.GET.FILE

```
  [LAMBDA (UDP TFTPCON DEVICE LOGSTREAM)                 ; Edited 14-Apr-87 20:19 by FS
    ;; Try to start receiving a file from the requestor as directed by the contents of the received UDP packet
    (DECLARE (GLOBALVARS \TFTP.SERVER.CONNECTIONS))
    (LET* ([FILENAMELENGTH (for I from BYTESPERWORD until (EQ 0 (\GETBASEBYTE (fetch (TFTP TFTPBASE) of UDP)
                                                                              I))
                           finally (RETURN (IDIFFERENCE I BYTESPERWORD]
           (FILENAME (ALLOCSTRING FILENAMELENGTH))
           [MODELENGTH (for I from (IPLUS BYTESPERWORD FILENAMELENGTH 1)
                        until (EQ 0 (\GETBASEBYTE (fetch (TFTP TFTPBASE) of UDP)
                                                  I))
                        finally (RETURN (IDIFFERENCE I (IPLUS BYTESPERWORD FILENAMELENGTH 1]
           (MODE (ALLOCSTRING MODELENGTH))
           (HOST (fetch (IP IPSOURCEADDRESS) of UDP))
           FILE TYPE TFTPSTREAM RESULT)
          (RESETLST
              (RESETSAVE NIL (LIST [FUNCTION (LAMBDA (TFTPCON)
                                               (LET* [(UDPSOCKET (fetch (TFTPCON UDPSOCKET) of TFTPCON))
                                                      (CONNECTION (CONS (fetch (TFTPCON HOST) of TFTPCON)
                                                                        (fetch (TFTPCON DESTSOCKET) of TFTPCON]
                                                     (SETQ \TFTP.SERVER.CONNECTIONS (REMOVE CONNECTION
                                                                                            \TFTP.SERVER.CONNECTIONS
                                                                                            ))
                                                     (AND RESETSTATE (UDP.CLOSE.SOCKET UDPSOCKET T]
                                   TFTPCON))
              (replace (TFTPCON DESTSOCKET) of TFTPCON with (fetch (UDP UDPSOURCEPORT) of UDP))
              (replace (TFTPCON HOST) of TFTPCON with HOST)
              ;; Read the filename out of the packet
              (\MOVEBYTES (fetch (TFTP TFTPBASE) of UDP)
                     BYTESPERWORD
```

```
                    (fetch (STRINGP BASE) of FILENAME)
                    (fetch (STRINGP OFFST) of FILENAME)
                    FILENAMELENGTH)
```

;; Read the mode out of the packet

```
(\MOVEBYTES (fetch (TFTP TFTPBASE) of UDP)
       (IPLUS BYTESPERWORD FILENAMELENGTH 1)
       (fetch (STRINGP BASE) of MODE)
       (fetch (STRINGP OFFST) of MODE)
       MODELENGTH)
(SETQ MODE (U-CASE MODE))
(printout LOGSTREAM "TFTP Server: Will attempt to receive " FILENAME " in " MODE " mode from host
" (\IP.ADDRESS.TO.STRING HOST)
       T)
(SETQ RESULT (COND
                  [[AND (SETQ TYPE (COND
                                   ((STREQUAL MODE "NETASCII")
                                    'TEXT)
                                   ((STREQUAL MODE "OCTET")
                                    'BINARY)
                                   (T (\TFTP.SEND.ERROR TFTPCON 0 (CONCAT "Unknown transfer
                                                                          type--" MODE))
                                      NIL)))
                        (SETQ FILE (LET [(OUTSTREAM (CAR (NLSETQ (OPENSTREAM
                                                                  FILENAME
                                                                  'OUTPUT
                                                                  'NEW
                                                                  (LIST (LIST 'TYPE TYPE]
                                                    (COND
                                                       ((NULL OUTSTREAM)
                                                        (\TFTP.SEND.ERROR TFTPCON 1 (CONCAT "Can't open file--"
                                                                                            FILENAME))
                                                        NIL)
                                                       (T OUTSTREAM]
```

;; Mode is OK, and file is open for input. Open the TFTP stream back to the requestor

```
                        (RESETSAVE NIL (LIST [FUNCTION (LAMBDA (FILE)
                                                       (COND
                                                          (RESETSTATE (CLOSEF? FILE)
                                                                 (DELFILE (FULLNAME FILE]
                                             FILE))
                        (SETQ TFTPSTREAM (create STREAM
                                               DEVICE _ DEVICE))
                        (replace (TFTPCON STREAM) of TFTPCON with TFTPSTREAM)
                        (replace (STREAM ACCESS) of TFTPSTREAM with 'INPUT)
                        (replace (TFTPSTREAM TFTPCON) of TFTPSTREAM with TFTPCON)
```

;; Send the first acknowledgement

```
                        (\TFTP.ACKNOWLEDGE TFTPSTREAM 0)
                        (\RELEASE.ETHERPACKET UDP)
                        (printout LOGSTREAM "TFTP Server: receiving " (FULLNAME FILE)
                              T)
                        (COND
                           ((NLSETQ (COPYBYTES TFTPSTREAM FILE))
                            (printout LOGSTREAM "TFTP Server: Done receiving " (FULLNAME FILE)
                                  T)
                            (CLOSEF? FILE))
                           (T (printout LOGSTREAM "TFTP Server: Failed to receive " (FULLNAME FILE)
                                 T)
                              (DELFILE (FULLNAME (CLOSEF? FILE]
                       (T (printout LOGSTREAM "TFTP Server: Failed to receive " (FULLNAME FILE)
                                T)
                          (\RELEASE.ETHERPACKET UDP)
                          NIL)))
```

;; Remove connection from list.

```
(LET (UDPSOCKET CONNECTION)
     (SETQ UDPSOCKET (fetch (TFTPCON UDPSOCKET) of TFTPCON))
     (SETQ CONNECTION (CONS (fetch (TFTPCON HOST) of TFTPCON)
                            (fetch (UDP UDPDESTPORT) of UDPSOCKET)))
     (SETQ \TFTP.SERVER.CONNECTIONS (REMOVE CONNECTION \TFTP.SERVER.CONNECTIONS)))
RESULT)])
```

(**TFTP.SEND.FILE**

```
[LAMBDA (UDP TFTPCON DEVICE LOGSTREAM)                    ; Edited 30-Jun-87 22:12 by scp
```

;; Try to start sending a file to the requestor as directed by the contents of the received UDP packet

```
(DECLARE (GLOBALVARS \TFTP.SERVER.CONNECTIONS))
(LET* ([FILENAMELENGTH (for I from BYTESPERWORD until (EQ 0 (\GETBASEBYTE (fetch (TFTP TFTPBASE) of UDP)
                                                                           I))
                          finally (RETURN (IDIFFERENCE I BYTESPERWORD]
       (FILENAME (ALLOCSTRING FILENAMELENGTH))
       [MODELENGTH (for I from (IPLUS BYTESPERWORD FILENAMELENGTH 1)
                      until (EQ 0 (\GETBASEBYTE (fetch (TFTP TFTPBASE) of UDP)
                                                I))
                      finally (RETURN (IDIFFERENCE I (IPLUS BYTESPERWORD FILENAMELENGTH 1]
```

```
          (MODE (ALLOCSTRING MODELENGTH))
          (HOST (fetch (IP IPSOURCEADDRESS) of UDP))
          FILE TYPE TFTPSTREAM RESULT)
         (RESETLST
             (RESETSAVE NIL (LIST [FUNCTION (LAMBDA (TFTPCON)
                                              (LET* [(UDPSOCKET (fetch (TFTPCON UDPSOCKET) of TFTPCON))
                                                     (CONNECTION (CONS (fetch (TFTPCON HOST) of TFTPCON)
                                                                       (fetch (TFTPCON DESTSOCKET) of TFTPCON]
                                                    (SETQ \TFTP.SERVER.CONNECTIONS (REMOVE CONNECTION
                                                                                           \TFTP.SERVER.CONNECTIONS
                                                                                           ))
                                                    (AND RESETSTATE (UDP.CLOSE.SOCKET UDPSOCKET T]
                                  TFTPCON))
             (replace (TFTPCON DESTSOCKET) of TFTPCON with (fetch (UDP UDPSOURCEPORT) of UDP))
             (replace (TFTPCON HOST) of TFTPCON with HOST)
             ;; Read the filename out of the packet
             (\MOVEBYTES (fetch (TFTP TFTPBASE) of UDP)
                    BYTESPERWORD
                    (fetch (STRINGP BASE) of FILENAME)
                    (fetch (STRINGP OFFST) of FILENAME)
                    FILENAMELENGTH)
             ;; Read the mode out of the packet
             (\MOVEBYTES (fetch (TFTP TFTPBASE) of UDP)
                    (IPLUS BYTESPERWORD FILENAMELENGTH 1)
                    (fetch (STRINGP BASE) of MODE)
                    (fetch (STRINGP OFFST) of MODE)
                    MODELENGTH)
             (SETQ MODE (U-CASE MODE))
             (printout LOGSTREAM "TFTP Server: Will attempt to send " FILENAME " in " MODE " mode to host "
                    (\IP.ADDRESS.TO.STRING HOST)
                    T)
             (SETQ RESULT (COND
                            ([AND (SETQ TYPE (COND
                                               ((STREQUAL MODE "NETASCII")
                                                'TEXT)
                                               ((STREQUAL MODE "OCTET")
                                                'BINARY)
                                               (T (\TFTP.SEND.ERROR TFTPCON 0 (CONCAT "Unknown transfer
                                                                                      type--" MODE))
                                                  NIL)))
                                  (SETQ FILE
                                   (LET* [(FULLFILENAME (INFILEP FILENAME))
                                          (INSTREAM (AND FULLFILENAME
                                                         (CAR (NLSETQ (OPENSTREAM FULLFILENAME 'INPUT
                                                                                  'OLD
                                                                                  (LIST (LIST 'TYPE TYPE]
                                                 (COND
                                                    ((NULL INSTREAM)
                                                     (\TFTP.SEND.ERROR TFTPCON 1 (CONCAT "Can't open file--" FILENAME)
                                                            )
                                                     NIL)
                                                    (T INSTREAM]
                                  ;; Mode is OK, and file is open for input.  Open the TFTP stream back to the requestor
                                  (SETQ TFTPSTREAM (create STREAM
                                                           DEVICE _ DEVICE))
                                  (replace (TFTPCON STREAM) of TFTPCON with TFTPSTREAM)
                                  (replace (STREAM ACCESS) of TFTPSTREAM with 'OUTPUT)
                                  (replace (TFTPSTREAM TFTPCON) of TFTPSTREAM with TFTPCON)
                                  ;; Use the incoming packet as the first data packet on the way out
                                  (\TFTP.SETUP UDP TFTPCON \TFTP.DATA NIL)
                                  ;; This is block number 1
                                  (UDP.APPEND.WORD UDP 1)
                                  (add (fetch (UDP UDPLENGTH) of UDP)
                                       512)
                                  (\TFTP.INPUT.BUFFER TFTPSTREAM UDP)
                                  (printout LOGSTREAM "TFTP Server: Sending " FILENAME T)
                                  (COND
                                     ((NLSETQ (PROGN (COPYBYTES FILE TFTPSTREAM)
                                                     (\TFTP.GETNEXTBUFFER TFTPSTREAM 'WRITE T)
                                                     (\TFTP.CLOSEFILE TFTPSTREAM)))
                                      (printout LOGSTREAM "TFTP Server: Done sending " FILENAME T))
                                     (T (printout LOGSTREAM "TFTP Server: Failed to send " FILENAME T)))
                                  (CLOSEF? FILE))
                                 (T (printout LOGSTREAM "TFTP Server: Failed to send " FILENAME T)
                                    (\RELEASE.ETHERPACKET UDP)
                                    NIL)))
             ;; Remove connection from list.
             (LET (UDPSOCKET CONNECTION)
                  (SETQ UDPSOCKET (fetch (TFTPCON UDPSOCKET) of TFTPCON))
                  (SETQ CONNECTION (CONS (fetch (TFTPCON HOST) of TFTPCON)
                                         (fetch (UDP UDPDESTPORT) of UDPSOCKET)))
```

```
                (SETQ \TFTP.SERVER.CONNECTIONS (REMOVE CONNECTION \TFTP.SERVER.CONNECTIONS)))
            RESULT)])

)

;; User functions

(DEFINEQ

(TFTP.SERVER
  [LAMBDA (LOGSTREAM)                                                    (* MPL " 2-Jun-85 19:39")

          (* * Create a new TFTP server. LOGSTREAM defaults to a popup window)

    (ADD.PROCESS `(TFTP.SERVER.PROCESS %, LOGSTREAM)
           'RESTARTABLE
           'HARDRESET])

(TFTP.GET
  [LAMBDA (FROM TO PARAMETERS)                                           (* MPL " 2-Jun-85 17:15")
    (LET ((EOLCONVENTION (CADR (FASSOC 'EOLCONVENTION PARAMETERS)))
          (TYPE (FASSOC 'TYPE PARAMETERS))
          (FROMNAME FROM)
          (TONAME TO))
         (RESETLST
             [SETQ TO (OPENSTREAM TO 'OUTPUT 'NEW NIL (COND
                                                         (TYPE (LIST TYPE]
             (RESETSAVE NIL (LIST [FUNCTION (LAMBDA (STREAM)
                                            (COND
                                               ((AND STREAM RESETSTATE)
                                                (CLOSEF? STREAM)
                                                (DELFILE (FULLNAME STREAM]
                                  TO))
             (SETQ FROM (\TFTP.OPENFILE FROM 'INPUT 'OLD PARAMETERS))
             (RESETSAVE NIL (LIST [FUNCTION (LAMBDA (STREAM)
                                            (AND STREAM RESETSTATE (CLOSEF STREAM]
                                  FROM))
             (COND
                (EOLCONVENTION (replace (STREAM EOLCONVENTION) of FROM with EOLCONVENTION)))
             [COND
                ((AND FROM TO)
                 (COPYCHARS FROM TO)
                 (AND (OPENP FROM)
                      (CLOSEF FROM))
                 (FULLNAME (CLOSEF TO)))
                (TO (ERRORX (LIST 9 FROMNAME)))
                (FROM (ERRORX (LIST 9 TONAME]) ])

(TFTP.PUT
  [LAMBDA (FROM TO PARAMETERS)                                           ; Edited 15-Apr-87 20:55 by FS
    (LET ((EOLCONVENTION (CADR (FASSOC 'EOLCONVENTION PARAMETERS)))
          (TYPE (FASSOC 'TYPE PARAMETERS)))
         ;; Why is TYPE not used anywhere?
         (RESETLST
             (SETQ FROM (OPENSTREAM FROM 'INPUT 'OLD))
             (RESETSAVE NIL (LIST [FUNCTION (LAMBDA (STREAM)
                                            (AND RESETSTATE (CLOSEF STREAM]
                                  FROM))
             (SETQ TO (\TFTP.OPENFILE TO 'OUTPUT 'NEW PARAMETERS))
             (RESETSAVE NIL (LIST [FUNCTION (LAMBDA (STREAM)
                                            (AND RESETSTATE (CLOSEF STREAM]
                                  TO))
             (COND
                (EOLCONVENTION (replace (STREAM EOLCONVENTION) of TO with EOLCONVENTION)))
             (COPYCHARS FROM TO)
             (CLOSEF FROM)
             ;; Removed (FULLNAME (CLOSEF TO))
             (CLOSEF TO))]) )

)

;; Tracing functions

(DEFINEQ

(PRINTTFTP
  [LAMBDA (TFTP FILE)                                                    (* ejs%: " 2-Jun-85 14:00")
    (DECLARE (GLOBALVARS TFTPOPCODES))
    (PRINTCONSTANT (fetch (TFTP OPCODE) of TFTP)
           TFTPOPCODES FILE "TFTP Opcode: ")
    (SELECTC (fetch (TFTP OPCODE) of TFTP)
        (\TFTP.RRQ (printout FILE " ")
```

```
                    (\TFTP.PRINT.REQUEST TFTP FILE))
          (\TFTP.WRQ (printout FILE " ")
                    (\TFTP.PRINT.REQUEST TFTP FILE))
          (\TFTP.ACK (printout FILE " ")
                    (\TFTP.PRINT.ACK TFTP FILE))
          (\TFTP.DATA (printout FILE " ")
                     (\TFTP.PRINT.DATA TFTP FILE))
          (\TFTP.ERROR (printout FILE " ")
                      (\TFTP.PRINT.ERROR TFTP FILE))
          NIL)
    (TERPRI FILE)
    (TERPRI FILE])
```

(\TFTP.PRINT.ACK

```
  [LAMBDA (TFTP FILE)                                          (* ejs%: " 2-Jun-85 12:48")
    (printout FILE "Block #: " (fetch (TFTP BLOCK#) of TFTP)
           T])
```

(\TFTP.PRINT.DATA

```
  [LAMBDA (TFTP FILE)                                          (* ejs%: " 2-Jun-85 14:00")
    (printout FILE "Block #: " (fetch (TFTP BLOCK#) of TFTP)
           T)
    (PRINTPACKETDATA (fetch (TFTP TFTPCONTENTS) of TFTP)
           \TFTPOVLEN
           '(CHARS 12 |...|)
           (IDIFFERENCE (fetch (UDP UDPLENGTH) of TFTP)
                  (CONSTANT (IPLUS \TFTPOVLEN \UDPOVLEN])
```

(\TFTP.PRINT.ERROR

```
  [LAMBDA (TFTP FILE)                                          (* ejs%: " 2-Jun-85 13:15")
    (printout FILE "Error code: " (fetch (TFTP ERRORCODE) of TFTP)
           T)
    (PRINTPACKETDATA (fetch (TFTP TFTPCONTENTS) of TFTP)
           0
           '(CHARS |...|)
           (IDIFFERENCE (fetch (UDP UDPLENGTH) of TFTP)
                  (CONSTANT (IPLUS \UDPOVLEN \TFTPOVLEN)))
           FILE])
```

(\TFTP.PRINT.REQUEST

```
  [LAMBDA (TFTP FILE)                                          (* ejs%: " 2-Jun-85 13:16")

          (* * Try to start sending a file to the requestor as directed by the contents of the received TFTP packet)

    (LET* ([FILENAMELENGTH (for I from BYTESPERWORD until (EQ 0 (\GETBASEBYTE (fetch (TFTP TFTPBASE) of TFTP)
                                                                              I))
                                finally (RETURN (IDIFFERENCE I BYTESPERWORD]
           (FILENAME (ALLOCSTRING FILENAMELENGTH))
           [MODELENGTH (for I from (IPLUS BYTESPERWORD FILENAMELENGTH 1)
                          until (EQ 0 (\GETBASEBYTE (fetch (TFTP TFTPBASE) of TFTP)
                                                    I))
                          finally (RETURN (IDIFFERENCE I (IPLUS BYTESPERWORD FILENAMELENGTH 1]
           (MODE (ALLOCSTRING MODELENGTH)))

          (* * Read the filename out of the packet)

          (\MOVEBYTES (fetch (TFTP TFTPBASE) of TFTP)
                 BYTESPERWORD
                 (fetch (STRINGP BASE) of FILENAME)
                 (fetch (STRINGP OFFST) of FILENAME)
                 FILENAMELENGTH)

          (* * Read the mode out of the packet)

          (\MOVEBYTES (fetch (TFTP TFTPBASE) of TFTP)
                 (IPLUS BYTESPERWORD FILENAMELENGTH 1)
                 (fetch (STRINGP BASE) of MODE)
                 (fetch (STRINGP OFFST) of MODE)
                 MODELENGTH)
          (printout FILE (SELECTC (fetch (TFTP OPCODE) of TFTP)
                              (\TFTP.RRQ "Read request for ")
                              (\TFTP.WRQ "Write request for ")
                              (SHOULDNT))
                 FILENAME " in mode " MODE T])

)

(\TFTP.INIT)

(PUTPROPS TCPTFTP COPYRIGHT ("Venue & Xerox Corporation" 1986 1987 1990))
```

FUNCTION INDEX

CONSTANT INDEX

VARIABLE INDEX

RECORD INDEX