File created: 30-Aug-90 13:46:39 **{DSK}<LISPFILES>TCP>TCPLLIP.;3**

changes to: (VARS TCPLLIPCOMS)

previous date: 29-Aug-90 16:28:12 {DSK}<LISPFILES>TCP>TCPLLIP.;2

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;


```
(RPAQQ TCPLLIPCOMS
       ((PROP MAKEFILE-ENVIRONMENT TCPLLIP)
        (COMS ;; IP definitions and addressing
              (DECLARE%: DONTCOPY (EXPORT (RECORDS IP IPSOCKET IPADDRESS)
                                          (CONSTANTS \IPOVLEN \MAX.IPDATALENGTH \IP.PROTOCOLVERSION
                                                 \IP.MAX.EPKTS.ON.QUEUE \IP.DEFAULT.TIME.TO.LIVE
                                                 \IP.WAKEUP.INTERVAL)
                                          (CONSTANTS * IPPACKETTYPES)
                                          (CONSTANTS * ICMPUNREACHABLES)
                                          (MACROS \IPDATABASE \IPDATALENGTH)))
              (ADDVARS (*IP-PROTOCOL-NAME-FROM-NUMBER* (17 . "UDP")
                               (6 . "TCP")
                               (1 . "ICMP")))
              (GLOBALVARS *IP-PROTOCOL-NAME-FROM-NUMBER*)
              ;; value in sysout is too small.  This is 512-(indexf (fetch epencapsulation))-2.  489 is more correct, but let's leave a word of slop for
              ;; off-by-ones
              (VARS (\10MBPACKETLENGTH 488))
              ;; Make it easier to see queuelength without opening up q.
              (FNS \SYSQUEUE.DEFPRINT \IPSOCKET.DEFPRINT)
              [DECLARE%: DONTEVAL@LOAD DOCOPY (P (DEFPRINT 'IPSOCKET '\IPSOCKET.DEFPRINT))
                     (P (DEFPRINT 'SYSQUEUE '\SYSQUEUE.DEFPRINT]
              (INITVARS (IPTRACETIME)
                     (IPONLYTYPES)
                     (IPIGNORETYPES)
                     (IPPRINTMACROS)
                     (IPTRACEFLG)
                     (IPTRACEFILE)
                     (\IP.INIT.FILE)
                     (\IP.DEFAULT.CONFIGURATION)
                     (\IP.HOSTNAMES (HASHARRAY 40 1.1))
                     (\IP.HOSTNUMBERS)
                     (INTERNET.LOCAL.DOMAIN))
              (INITRECORDS IP IPSOCKET IPADDRESS)
              (GLOBALVARS IPTRACEFILE IPTRACEFLG IPIGNORETYPES IPONLYTYPES IPPRINTMACROS \IP.HOSTNAMES
                     \IP.INIT.FILE INTERNET.LOCAL.DOMAIN \IP.DEFAULT.CONFIGURATION \IP.HOSTNUMBERS)
              (FILES (SYSLOAD)
                     TCPHTE TCPLLICMP TCPLLAR)
              (ADDVARS (\PACKET.PRINTERS (2048 . PRINTIP)))
              (FNS \CANONICALIZE.IP.HOSTNAME DODIP.HOSTP IPHOSTADDRESS IPHOSTNAME IPTRACE IPTRACEWINDOW.BUTTONFN
                   PRINTIP PRINTIPDATA \IPADDRESSCLASS \IPEVENTFN \IPHOSTADDRESS \IPNETADDRESS
                   \IP.ADDRESS.TO.STRING \IP.BROADCAST.ADDRESS \IP.LEGAL.ADDRESS \IP.MAKE.BROADCAST.ADDRESS
                   \IP.PRINT.ADDRESS \IP.READ.STRING.ADDRESS \DOMAIN.NAME.QUALIFY.FULLY))
        (COMS ;; Startup and shutdown
              (INITVARS (*IP-DEFAULT-HOSTS-FILE*)
                     (TCP.ALWAYS.READ.HOSTS.FILE T)
                     (\TCP.LAST.HOSTS.FILE.DATE)
                     (\TCP.LAST.HOSTS.FILE.READ)
                     (\IPFLG)
                     (\IP.READY)
                     (\IP.READY.EVENT (CREATE.EVENT "IP Ready"))
                     (\IP.WAKEUP.TIMER)
                     (IPTRACEFLG)
                     (\IP.WAKEUP.EVENT (CREATE.EVENT "IP Wakeup")))
              (GLOBALVARS \IPFLG \IP.READY \IP.READY.EVENT \IP.WAKEUP.TIMER \IP.WAKEUP.EVENT
                     TCP.ALWAYS.READ.HOSTS.FILE \TCP.LAST.HOSTS.FILE.DATE \TCP.LAST.HOSTS.FILE.READ
                     *IP-DEFAULT-HOSTS-FILE*)
              (FNS STOPIP \IPINIT \IPLISTENER \IP.REINITIALIZE.FROM.SCRATCH \IP.RESTART.FROM.CONFIGURATION
                   \IP.MAYBE.READ.HOSTS.TXT \IP.READ.INIT.FILE \IP.PROMPT.FOR.FILE.NAME)
              (ADDVARS (RESTARTETHERFNS \IPEVENTFN)))
        (COMS ;; Early IP reception functions
              (DECLARE%: DONTCOPY (EXPORT (CONSTANTS * IPADDRESSTYPES)))
              (INITVARS (\IP.LOCAL.ADDRESSES)
                     (\IP.SUBNET.MASKS)
                     (\IP.GATEWAY.FLG))
              (VARS (\IP.ADDRESS.BOX (\CREATECELL \FIXP)))
              (GLOBALVARS \IP.LOCAL.ADDRESSES \IP.SUBNET.MASKS \IP.GATEWAY.FLG \IP.ADDRESS.BOX)
```

```
            (MACROS \IP.FIX.DEST.HOST \IP.FIX.DEST.NET \IP.FIX.SOURCE.HOST \IP.FIX.SOURCE.NET)
            (FNS \HANDLE.RAW.IP \FORWARD.IP \IP.LOCAL.DESTINATION \IPCHECKSUM \IP.CHECKSUM.OK \IP.SET.CHECKSUM
                 ))
      (COMS ;; Protocol Distribution
            (DECLARE%: DONTCOPY (EXPORT (CONSTANTS * IPPROTOCOLTYPES)))
            (INITVARS (\IP.PROTOCOLS))
            (GLOBALVARS \IP.PROTOCOLS)
            (FNS \IP.HAND.TO.PROTOCOL \IP.DEFAULT.INPUTFN \IP.DEFAULT.NOSOCKETFN \IP.ADD.PROTOCOL
                 \IP.DELETE.PROTOCOL \IP.FIND.PROTOCOL \IP.FIND.PROTOCOL.SOCKET \IP.FIND.SOCKET
                 \IP.OPEN.SOCKET \IP.CLOSE.SOCKET))
      (COMS ;; Fragmentation Handling
            (DECLARE%: DONTCOPY (EXPORT (RECORDS AssemblyRecord FragmentRecord FragmentID)))
            (INITVARS (\IP.FRAGMENT.LIST)
                   (\IP.FRAGMENT.LOCK (CREATE.MONITORLOCK "IP Fragment Processing Lock")))
            (GLOBALVARS \IP.FRAGMENT.LIST \IP.FRAGMENT.LOCK)
            (CONSTANTS (\IP.FRAGMENTATION.UNIT 8))
            (FNS \HANDLE.RAW.IP.FRAGMENT \IP.NEW.FRAGMENT.LST \IP.COPY.FRAGMENT.HEADER.TO.PACKET.HEADER
                 \IP.ADD.FRAGMENT \IP.FIND.MATCHING.FRAGMENTS \IP.FRAGMENTED.PACKET
                 \IP.CHECK.REASSEMBLY.TIMEOUTS \IP.DELETE.FRAGMENT \IP.PRINT.FRAGMENT))
      (COMS ;; Option Processing
            [DECLARE%: DONTCOPY (EXPORT (CONSTANTS * IPOPTIONTYPES)
                                        (CONSTANTS (IP.OPTION.NUMBER.BYTESPEC (BYTE 5 0]
            (FNS \IP.PROCESS.OPTIONS \IP.OPTION.RECORD.ROUTE \IP.OPTION.STRICT.SOURCE.ROUTE
                 \IP.OPTION.TIMESTAMP))
      (COMS ;; Packet Transmission and routing
            (INITVARS (\IP.ROUTING.TABLE (CONS))
                   (\IP.DEFAULT.GATEWAY)
                   (\IP.LOCAL.NETWORKS)
                   (\IP.GATEWAY.FORWARDING.FUNCTIONS))
            (GLOBALVARS \IP.ROUTING.TABLE \IP.DEFAULT.GATEWAY \IP.LOCAL.NETWORKS
                   \IP.GATEWAY.FORWARDING.FUNCTIONS)
            (FNS \IP.SETUPIP \IP.TRANSMIT \IP.ROUTE.PACKET)
            (FNS IP.GET IP.SEND IP.PACKET.WATCHER)
            (MACROS IP.SEND))
      (COMS ;; Client functions for building packets
            (FNS \IP.APPEND.BYTE \IP.APPEND.CELL \IP.APPEND.STRING \IP.APPEND.WORD \IP.GET.BYTE \IP.GET.CELL
                 \IP.GET.STRING \IP.GET.WORD \IP.PUT.BYTE \IP.PUT.CELL \IP.PUT.STRING \IP.PUT.WORD)
            (MACROS \IP.GET.BYTE \IP.GET.CELL \IP.GET.STRING \IP.GET.WORD \IP.PUT.BYTE \IP.PUT.CELL
                   \IP.PUT.STRING \IP.PUT.WORD))
      (P (MOVD? 'NILL 'IP.DEFAULT.CONFIGURATION))
      (DECLARE%: EVAL@COMPILE DONTCOPY (GLOBALVARS \IP.LOCAL.NETWORKS \IP.DEFAULT.GATEWAY \IP.INIT.FILE
                                                   \IP.SUBNET.MASKS \PROCESS.AFTEREXIT.EVENT \PROC.READY
                                                   \AR.IP.TO.10MB.ALIST))))

(PUTPROPS TCPLLIP MAKEFILE-ENVIRONMENT (:PACKAGE "IL" :READTABLE "INTERLISP"))

;; IP definitions and addressing

(DECLARE%: DONTCOPY

;; FOLLOWING DEFINITIONS EXPORTED

(DECLARE%: EVAL@COMPILE

(ACCESSFNS IP [(IPBASE (LOCF (fetch (ETHERPACKET EPBODY) of DATUM]
       [BLOCKRECORD IPBASE ((IPVERSION BITS 4)                    ; Protocol version
                            (IPHEADERLENGTH BITS 4)               ; Head length, in cells
                            (IPSERVICE BYTE)                      ; Service type
                            (IPTOTALLENGTH WORD)                  ; Packet length, in bytes
                            (IPID WORD)                           ; Packet id
                            (NIL BITS 1)
                            (IPDONTFRAGMENT FLAG)                 ; Don't fragment me
                            (IPMOREFRAGMENTS FLAG)                ; Last fragment
                            (IPFRAGMENTOFFSET BITS 13)            ; Fragment position
                            (IPTIMETOLIVE BYTE)                   ; Hop limiter
                            (IPPROTOCOL BYTE)                     ; Client protocol
                            (IPHEADERCHECKSUM WORD)               ; Header-only checksum
                            (IPSOURCEADDRESS FIXP)
                            (IPDESTINATIONADDRESS FIXP)
                            (IPOPTIONSSTART BYTE)                 ; Options or data start here

                            )
            [ACCESSFNS IPSERVICE ((IPSERVICEBASE (LOCF DATUM)))
                   (BLOCKRECORD IPSERVICEBASE ((IPPRECEDENCE BITS 3)
                                               (IPDELAY FLAG)
                                               (IPTHROUGHPUT FLAG)
                                               (IPRELIABILITY FLAG)
                                               (NIL BITS 2]
            [ACCESSFNS IPDESTINATIONADDRESS ((IPDESTBASE (LOCF DATUM)))
                   (ACCESSFNS IPDESTBASE ([IPDESTINATIONNET (COND
                                                             ((EQ \IP.CLASS.A (fetch (IPADDRESS CLASSA)
```

```
                                                                    of DATUM))
                                                              (fetch (IPADDRESS CLASSANET) of DATUM))
                                                             ((EQ \IP.CLASS.B (fetch (IPADDRESS CLASSB)
                                                                    of DATUM))
                                                              (fetch (IPADDRESS CLASSBNET) of DATUM))
                                                             ((EQ \IP.CLASS.C (fetch (IPADDRESS CLASSC)
                                                                    of DATUM))
                                                              (fetch (IPADDRESS CLASSCNET) of DATUM))
                                                             (T (ERROR "Illegal address class" DATUM)))
                                                    (COND
                                                       ((EQ \IP.CLASS.A (fetch (IPADDRESS CLASSA) of DATUM))
                                                        (replace (IPADDRESS CLASSANET) of DATUM with NEWVALUE))
                                                       ((EQ \IP.CLASS.B (fetch (IPADDRESS CLASSB) of DATUM))
                                                        (replace (IPADDRESS CLASSBNET) of DATUM with NEWVALUE))
                                                       ((EQ \IP.CLASS.C (fetch (IPADDRESS CLASSC) of DATUM))
                                                        (replace (IPADDRESS CLASSCNET) of DATUM with NEWVALUE))
                                                       (T (ERROR "Illegal address class" DATUM]
                                                (IPDESTINATIONHOST (COND
                                                                      ((EQ \IP.CLASS.A (fetch (IPADDRESS CLASSA)
                                                                             of DATUM))
                                                                       (fetch (IPADDRESS CLASSAHOST) of DATUM))
                                                                      ((EQ \IP.CLASS.B (fetch (IPADDRESS CLASSB)
                                                                             of DATUM))
                                                                       (fetch (IPADDRESS CLASSBHOST) of DATUM))
                                                                      ((EQ \IP.CLASS.C (fetch (IPADDRESS CLASSC)
                                                                             of DATUM))
                                                                       (fetch (IPADDRESS CLASSCHOST) of DATUM))
                                                                      (T (ERROR "Illegal address class" DATUM)))
                                                    (COND
                                                       ((EQ \IP.CLASS.A (fetch (IPADDRESS CLASSA) of DATUM))
                                                        (replace (IPADDRESS CLASSAHOST) of DATUM with NEWVALUE))
                                                       ((EQ \IP.CLASS.B (fetch (IPADDRESS CLASSB) of DATUM))
                                                        (replace (IPADDRESS CLASSBHOST) of DATUM with NEWVALUE))
                                                       ((EQ \IP.CLASS.C (fetch (IPADDRESS CLASSC) of DATUM))
                                                        (replace (IPADDRESS CLASSCHOST) of DATUM with NEWVALUE))
                                                       (T (ERROR "Illegal address class" DATUM]
                                    (ACCESSFNS IPSOURCEADDRESS ((IPSOURCEBASE (LOCF DATUM)))
                                           (ACCESSFNS IPSOURCEBASE ([IPSOURCENET (COND
                                                                                    ((EQ \IP.CLASS.A (fetch (IPADDRESS CLASSA)
                                                                                           of DATUM))
                                                                                     (fetch (IPADDRESS CLASSANET) of DATUM))
                                                                                    ((EQ \IP.CLASS.B (fetch (IPADDRESS CLASSB)
                                                                                           of DATUM))
                                                                                     (fetch (IPADDRESS CLASSBNET) of DATUM))
                                                                                    ((EQ \IP.CLASS.C (fetch (IPADDRESS CLASSC)
                                                                                           of DATUM))
                                                                                     (fetch (IPADDRESS CLASSCNET) of DATUM))
                                                                                    (T (ERROR "Illegal address class" DATUM)))
                                                                              (COND
                                                                                 ((EQ \IP.CLASS.A (fetch (IPADDRESS CLASSA) of DATUM))
                                                                                  (replace (IPADDRESS CLASSANET) of DATUM with NEWVALUE))
                                                                                 ((EQ \IP.CLASS.B (fetch (IPADDRESS CLASSB) of DATUM))
                                                                                  (replace (IPADDRESS CLASSBNET) of DATUM with NEWVALUE))
                                                                                 ((EQ \IP.CLASS.C (fetch (IPADDRESS CLASSC) of DATUM))
                                                                                  (replace (IPADDRESS CLASSCNET) of DATUM with NEWVALUE))
                                                                                 (T (ERROR "Illegal address class" DATUM]
                                                                        (IPSOURCEHOST (COND
                                                                                         ((EQ \IP.CLASS.A (fetch (IPADDRESS CLASSA)
                                                                                                of DATUM))
                                                                                          (fetch (IPADDRESS CLASSAHOST) of DATUM))
                                                                                         ((EQ \IP.CLASS.B (fetch (IPADDRESS CLASSB)
                                                                                                of DATUM))
                                                                                          (fetch (IPADDRESS CLASSBHOST) of DATUM))
                                                                                         ((EQ \IP.CLASS.C (fetch (IPADDRESS CLASSC)
                                                                                                of DATUM))
                                                                                          (fetch (IPADDRESS CLASSCHOST) of DATUM))
                                                                                         (T (ERROR "Illegal address class" DATUM)))
                                                                                   (COND
                                                                                      ((EQ \IP.CLASS.A (fetch (IPADDRESS CLASSA) of DATUM))
                                                                                       (replace (IPADDRESS CLASSAHOST) of DATUM with NEWVALUE))
                                                                                      ((EQ \IP.CLASS.B (fetch (IPADDRESS CLASSB) of DATUM))
                                                                                       (replace (IPADDRESS CLASSBHOST) of DATUM with NEWVALUE))
                                                                                      ((EQ \IP.CLASS.C (fetch (IPADDRESS CLASSC) of DATUM))
                                                                                       (replace (IPADDRESS CLASSCHOST) of DATUM with NEWVALUE))
                                                                                      (T (ERROR "Illegal address class" DATUM]
                                           (TYPE? (type? ETHERPACKET DATUM)))

(DATATYPE IPSOCKET ((PROTOCOL BYTE)
                    (IPSLINK POINTER)                          ; Other sockets of this protocol type
                    (NIL BYTE)
                    (IPSQUEUE POINTER)                         ; Queue of packets for this protocol
                    (IPSQUEUELENGTH WORD)                      ; Count of packets of input queue
                    (IPSQUEUEALLOC WORD)                       ; Max count allowed
                    (IPSDESTSOCKETCOMPAREFN POINTER)           ; Call this to compare dest protocol socket to this socket
                    (IPSOCKET POINTER)                         ; This socket
                    (IPSINPUTFN POINTER)                       ; Call to hand packet to protocol
                    (IPSEVENT POINTER)                         ; Notify me when a packet arrives
```

```
                    (IPSNOSOCKETFN POINTER)                    ; Call this when no socket found
                    (IPSICMPFN POINTER)                        ; Call this when an ICMP packet is received on this protocol

                    )
       IPSQUEUE _ (create SYSQUEUE)
       IPSQUEUEALLOC _ \IP.MAX.EPKTS.ON.QUEUE IPSEVENT _ (CREATE.EVENT)
       IPSINPUTFN _ (FUNCTION \IP.DEFAULT.INPUTFN)
       IPSICMPFN _ (FUNCTION \RELEASE.ETHERPACKET))

[BLOCKRECORD IPADDRESS ((ADDRESS FIXP))
       ;; Class A nets: high bit is 0
       (BLOCKRECORD IPADDRESS ((CLASSA BITS 1)
                               (CLASSANET BITS 7)
                               (CLASSAHOST BITS 24)))
       ;; Class B nets: high 2 bits are 10
       (BLOCKRECORD IPADDRESS ((CLASSB BITS 2)))
       (BLOCKRECORD IPADDRESS ((CLASSBNET BITS 16)
                               (CLASSBHOST BITS 16)))
       ;; Class C nets: high 3 bits are 110
       (BLOCKRECORD IPADDRESS ((CLASSC BITS 3)))
       (BLOCKRECORD IPADDRESS ((CLASSCNETB1 BITS 8)
                               (CLASSCNETB2 BITS 8)
                               (CLASSCNETB3 BITS 8)
                               (CLASSCHOST BITS 8)))           ; I wish I could say just net bits 24, host bits 8, but
                                                               ; BLOCKRECORD barfs
       (BLOCKRECORD IPADDRESS ((CLASSCNETHI BITS 16)))
       (ACCESSFNS IPADDRESS ((CLASSCNET (\MAKENUMBER (FETCH CLASSCNETB1 OF DATUM)
                                                     (LOGOR (LLSH (FETCH CLASSCNETB2 OF DATUM)
                                                                  8)
                                                            (FETCH CLASSCNETB3 OF DATUM)))
                                        (PROGN (REPLACE CLASSCNETHI OF DATUM WITH (LRSH NEWVALUE 8))
                                               (REPLACE CLASSCNETB3 OF DATUM WITH (LOGAND NEWVALUE 255))
                                               DATUM]
)

(/DECLAREDATATYPE 'IPSOCKET '(BYTE POINTER BYTE POINTER WORD WORD POINTER POINTER POINTER POINTER POINTER
                                   POINTER)
       ;; ---field descriptor list elided by lister---
       '18)

(DECLARE%: EVAL@COMPILE

(RPAQQ \IPOVLEN 20)

(RPAQQ \MAX.IPDATALENGTH 556)

(RPAQQ \IP.PROTOCOLVERSION 4)

(RPAQQ \IP.MAX.EPKTS.ON.QUEUE 16)

(RPAQQ \IP.DEFAULT.TIME.TO.LIVE 120)

(RPAQQ \IP.WAKEUP.INTERVAL 15000)

(CONSTANTS \IPOVLEN \MAX.IPDATALENGTH \IP.PROTOCOLVERSION \IP.MAX.EPKTS.ON.QUEUE \IP.DEFAULT.TIME.TO.LIVE
       \IP.WAKEUP.INTERVAL)
)

(RPAQQ IPPACKETTYPES ((\EPT.IP 2048)
                      (\EPT.AR 2054)
                      (\EET.IP 513)
                      (\EPT.CHAOS 2052)))

(DECLARE%: EVAL@COMPILE

(RPAQQ \EPT.IP 2048)

(RPAQQ \EPT.AR 2054)

(RPAQQ \EET.IP 513)

(RPAQQ \EPT.CHAOS 2052)

(CONSTANTS (\EPT.IP 2048)
       (\EPT.AR 2054)
       (\EET.IP 513)
       (\EPT.CHAOS 2052))
)

(RPAQQ ICMPUNREACHABLES ((\ICMP.NET.UNREACHABLE 0)
                         (\ICMP.HOST.UNREACHABLE 1)
                         (\ICMP.PROTOCOL.UNREACHABLE 2)
                         (\ICMP.PORT.UNREACHABLE 3)
```

```
                          (\ICMP.CANT.FRAGMENT 4)
                          (\ICMP.SOURCE.ROUTE 5)))

(DECLARE%: EVAL@COMPILE

(RPAQQ \ICMP.NET.UNREACHABLE 0)

(RPAQQ \ICMP.HOST.UNREACHABLE 1)

(RPAQQ \ICMP.PROTOCOL.UNREACHABLE 2)

(RPAQQ \ICMP.PORT.UNREACHABLE 3)

(RPAQQ \ICMP.CANT.FRAGMENT 4)

(RPAQQ \ICMP.SOURCE.ROUTE 5)

(CONSTANTS (\ICMP.NET.UNREACHABLE 0)
       (\ICMP.HOST.UNREACHABLE 1)
       (\ICMP.PROTOCOL.UNREACHABLE 2)
       (\ICMP.PORT.UNREACHABLE 3)
       (\ICMP.CANT.FRAGMENT 4)
       (\ICMP.SOURCE.ROUTE 5))
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS \IPDATABASE MACRO [LAMBDA (IP)                        (* ejs%: "26-Dec-84 17:50")
                                                                (* Returns the LOCF of the start of the data in the packet)
                                (\ADDBASE (fetch (IP IPBASE) of IP)
                                       (UNFOLD (fetch (IP IPHEADERLENGTH) of IP)
                                              2])

(PUTPROPS \IPDATALENGTH MACRO [LAMBDA (IP)
                                  (IDIFFERENCE (fetch (IP IPTOTALLENGTH) of IP)
                                         (LLSH (fetch (IP IPHEADERLENGTH) of IP)
                                               2])
)
)

;; END EXPORTED DEFINITIONS

(ADDTOVAR *IP-PROTOCOL-NAME-FROM-NUMBER* (17 . "UDP")
                                         (6 . "TCP")
                                         (1 . "ICMP"))

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS *IP-PROTOCOL-NAME-FROM-NUMBER*)
)

;; value in sysout is too small.  This is 512-(indexf (fetch epencapsulation))-2.  489 is more correct, but let's leave a word of slop for off-by-ones

(RPAQQ \10MBPACKETLENGTH 488)

;; Make it easier to see queuelength without opening up q.

(DEFINEQ
```

(\SYSQUEUE.DEFPRINT

```
  [LAMBDA (Q STREAM)                                    ; Edited  8-Sep-89 11:06 by bvm
    (\DEFPRINT.BY.NAME Q STREAM (if (fetch (SYSQUEUE SYSQUEUEHEAD) of Q)
                                   then (\QUEUELENGTH Q)
                                 else "Empty")
          "SysQueue"])
```

(\IPSOCKET.DEFPRINT

```
  [LAMBDA (SOCKET STREAM)                               ; Edited 25-Aug-88 17:51 by bvm
    ;; Print an object using its name, for example, #<FDev ERIS/76,5432>.  NAME is the object's name (or NIL if this one happens to be nameless),
    ;; TYPENAME is a string giving the generic name you want to appear in front, e.g., "FDev"
    (\OUTCHAR STREAM (fetch (READTABLEP HASHMACROCHAR) of *READTABLE*))
    (\OUTCHAR STREAM (CHARCODE <))
    (LET ((TYPE (CDR (ASSOC (fetch (IPSOCKET PROTOCOL) of SOCKET)
                            *IP-PROTOCOL-NAME-FROM-NUMBER*)))
          (NUM (fetch (IPSOCKET IPSOCKET) of SOCKET))
          (*PRINT-BASE* 10))
         (\SOUT (if TYPE
                    then (MKSTRING TYPE)
                  else "IP")
                STREAM)
         (\SOUT " Socket" STREAM)
         (if (if (FIXP NUM)
               elseif (NULL NUM)
                 then                                   ; I assume this is the master
```

```
                    (SETQ NUM "Head"))
              then (\OUTCHAR STREAM (CHARCODE SPACE))
                   (PRIN3 NUM STREAM)))
    (\OUTCHAR STREAM (CHARCODE /))
    (\PRINTADDR SOCKET STREAM)
    (\OUTCHAR STREAM (CHARCODE >))
    T])

)

(DECLARE%: DONTEVAL@LOAD DOCOPY

(DEFPRINT 'IPSOCKET '\IPSOCKET.DEFPRINT)

(DEFPRINT 'SYSQUEUE '\SYSQUEUE.DEFPRINT)
)

(RPAQ? IPTRACETIME )

(RPAQ? IPONLYTYPES )

(RPAQ? IPIGNORETYPES )

(RPAQ? IPPRINTMACROS )

(RPAQ? IPTRACEFLG )

(RPAQ? IPTRACEFILE )

(RPAQ? \IP.INIT.FILE )

(RPAQ? \IP.DEFAULT.CONFIGURATION )

(RPAQ? \IP.HOSTNAMES (HASHARRAY 40 1.1))

(RPAQ? \IP.HOSTNUMBERS )

(RPAQ? INTERNET.LOCAL.DOMAIN )

(/DECLAREDATATYPE 'IPSOCKET '(BYTE POINTER BYTE POINTER WORD WORD POINTER POINTER POINTER POINTER POINTER
                                POINTER)
       ;; ---field descriptor list elided by lister---
       '18)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS IPTRACEFILE IPTRACEFLG IPIGNORETYPES IPONLYTYPES IPPRINTMACROS \IP.HOSTNAMES \IP.INIT.FILE
       INTERNET.LOCAL.DOMAIN \IP.DEFAULT.CONFIGURATION \IP.HOSTNUMBERS)
)

(FILESLOAD (SYSLOAD)
       TCPHTE TCPLLICMP TCPLLAR)

(ADDTOVAR \PACKET.PRINTERS (2048 . PRINTIP))

(DEFINEQ
```

(\CANONICALIZE.IP.HOSTNAME

```
  [LAMBDA (NAME)                                          ; Edited 12-Apr-88 17:18 by bvm
    (AND \IP.READY (IPHOSTADDRESS NAME)
         NAME])
```

(DODIP.HOSTP

```
  [LAMBDA (NAME)                                          ; Edited 27-Feb-89 21:49 by welch
    (COND
       ((NULL NAME)
        NIL)
       ((NUMBERP NAME))
       (T (LET [(NAME (\DOMAIN.NAME.QUALIFY.FULLY (U-CASE NAME]
                (COND
                   ((IPHOSTADDRESS NAME))
                   (T (if (CL:FBOUNDP 'DOMAIN.LOOKUP.ADDRESS)
                          then (CAR (DOMAIN.LOOKUP.ADDRESS NAME])
```

(IPHOSTADDRESS

```
  [LAMBDA (NAME)                                          ; Edited 19-Jan-88 14:41 by FS
    (LET (ENTRY)
         ;; Hack to handle strings, by canonicalizing NAME
         (SETQ NAME (MKATOM (U-CASE NAME)))
         (SETQ ENTRY (GETHASH NAME \IP.HOSTNAMES))
         (COND
            (ENTRY (LET [(ADDRESS (CAR (fetch (HOSTS.TXT.ENTRY HTE.ADDRESSES) of ENTRY]
                        [COND
```

```
                        ((NOT (SASSOC ADDRESS \IP.HOSTNUMBERS))
                         (push \IP.HOSTNUMBERS (CONS ADDRESS NAME]
                      ADDRESS))
          ((\IP.READ.STRING.ADDRESS NAME])
```

(IPHOSTNAME

```
[LAMBDA (IPADDRESS)                                          (* ejs%: "22-Apr-85 13:54")
  (OR (CDR (SASSOC IPADDRESS \IP.HOSTNUMBERS))
      (MKATOM (\IP.ADDRESS.TO.STRING IPADDRESS])
```

(IPTRACE

```
[LAMBDA (FLG REGION)                                         ; Edited 13-Sep-88 14:53 by bvm
  (MAKE-NETWORK-TRACE-WINDOW 'IPTRACEFLG 'IPTRACEFILE "IP traffic" REGION FLG])
```

(IPTRACEWINDOW.BUTTONFN

```
[LAMBDA (WINDOW)                                             (* ejs%: " 2-Jun-85 13:05")
  (COND
     ((MOUSESTATE (NOT UP))
      (SETQ IPTRACEFLG (SELECTQ IPTRACEFLG
                           (NIL T)
                           (T 'PEEK)
                           (PEEK NIL)
                           NIL))
      (printout WINDOW T "[Tracing " (SELECTQ IPTRACEFLG
                                         (T "on")
                                         (PEEK "peek")
                                         "off")
            "]" T])
```

(PRINTIP

```
[LAMBDA (IP CALLER FILE PRE.NOTE DOFILTER)                   ; Edited 28-Apr-88 14:05 by bvm
  (PROG ((*STANDARD-OUTPUT* (GETSTREAM (OR FILE IPTRACEFILE)
                                       'OUTPUT))
         (PROTOCOL (fetch (IP IPPROTOCOL) of IP))
         MACRO LENGTH)
        [COND
           (DOFILTER (COND
                        ((COND
                            (IPONLYTYPES (NOT (FMEMB PROTOCOL IPONLYTYPES)))
                            (IPIGNORETYPES (FMEMB PROTOCOL IPIGNORETYPES)))
                         (RETURN (PRIN1 (SELECTQ CALLER
                                                 ((PUT RAWPUT)
                                                      '!)
                                                 ((GET RAWGET)
                                                      '+)
                                                 '?]
        (AND PRE.NOTE (printout NIL T PRE.NOTE))
        (if CALLER
            then                                             ; Print GET or PUT
                 (FRESHLINE)
                 (PRINTOUT NIL CALLER " "))
        (printout NIL "From " (\IP.ADDRESS.TO.STRING (fetch (IP IPSOURCEADDRESS) of IP))
               " to "
               (\IP.ADDRESS.TO.STRING (fetch (IP IPDESTINATIONADDRESS) of IP)))
        (if IPTRACETIME
            then (LET ((CSECS (\CENTICLOCK IP)))
                      (PRINTOUT NIL " [" .I4 (IQUOTIENT CSECS 100)
                             "." .I2..T (IREMAINDER CSECS 100)
                             "]")))
        (TERPRI)
        [COND
           ((AND (SETQ MACRO (CDR (FASSOC PROTOCOL IPPRINTMACROS)))
                 (NLISTP MACRO))                             ; Macro is a function to which to dispatch for the printing.
            (CL:FUNCALL MACRO IP *STANDARD-OUTPUT*)
            (RETURN (TERPRI]
        (printout NIL "Length = " .P2 (SETQ LENGTH (fetch (IP IPTOTALLENGTH) of IP))
               " bytes" " (header + " .P2 (IDIFFERENCE LENGTH \IPOVLEN)
               ")" T "Protocol = ")
        (PRINTCONSTANT PROTOCOL IPPROTOCOLTYPES NIL)
        (TERPRI)
        [COND
           ((IGREATERP LENGTH \IPOVLEN)                      ; MACRO tells how to print data.
            (PRIN1 "Contents: ")
            (PRINTIPDATA IP (OR MACRO '(BYTES 12 |...|]
        (TERPRI)
        (RETURN IP])
```

(PRINTIPDATA

```
[LAMBDA (IP MACRO OFFSET FILE)                               (* ejs%: "27-Dec-84 18:43")

        (* * Prints DATA part of IP starting at OFFSET (Default zero) according to MACRO.
        MACRO contains elements describing what format the data is in -
        WORDS, BYTES, CHARS%: print as words, bytes (numeric) or ascii characters -
```

<number>%: subsequent commands apply starting at this byte offset -
...%: print "..." and quit if you still have data at this point)

```
(PROG ((DATA (\IPDATABASE IP))
       (LENGTH (\IPDATALENGTH IP)))
      (PRINTPACKETDATA DATA OFFSET MACRO LENGTH FILE])
```

(\IPADDRESSCLASS

```
[LAMBDA (IPADDRESS)                                          ; Edited 26-Oct-88 12:49 by bvm
  (if (SMALLP IPADDRESS)
      then                                                   ; bogus unless it's broadcastp
           '\IP.CLASS.A
    elseif (EQ \IP.CLASS.C (SETQ IPADDRESS (fetch (IPADDRESS CLASSC) of IPADDRESS)))
      then '\IP.CLASS.C
    elseif (EQ \IP.CLASS.B (SETQ IPADDRESS (LRSH IPADDRESS 1)))
      then '\IP.CLASS.B
    elseif (EQ \IP.CLASS.A (LRSH IPADDRESS 1))
      then '\IP.CLASS.A])
```

(\IPEVENTFN

```
[LAMBDA (EVENT)                                              ; Edited 13-Sep-88 18:53 by Hiroshi Hayata
   ;; If maiko, do nothing.
   ;; Call of \IPINIT with AFTERSYSOUT on maiko cause RAID.
   (COND
      ((EQ \MACHINETYPE \MAIKO)
       NIL)
      (T (COND
            (\IPFLG (\IPINIT EVENT])
```

(\IPHOSTADDRESS

```
[LAMBDA (IPADDRESS)                                          ; Edited 26-Oct-88 12:43 by bvm
  (if (SMALLP IPADDRESS)
      then                                                   ; can only be class a or bogus
           (LOGAND IPADDRESS MAX.SMALLP)
    elseif (EQ (fetch (IPADDRESS CLASSA) of IPADDRESS)
               \IP.CLASS.A)
      then (fetch (IPADDRESS CLASSAHOST) of IPADDRESS)
    elseif (EQ (fetch (IPADDRESS CLASSB) of IPADDRESS)
               \IP.CLASS.B)
      then (fetch (IPADDRESS CLASSBHOST) of IPADDRESS)
    elseif (EQ (fetch (IPADDRESS CLASSC) of IPADDRESS)
               \IP.CLASS.C)
      then (fetch (IPADDRESS CLASSCHOST) of IPADDRESS])
```

(\IPNETADDRESS

```
[LAMBDA (IPADDRESS)                                          ; Edited 26-Oct-88 12:45 by bvm
  (if (SMALLP IPADDRESS)
      then                                                   ; bogus unless it's broadcastp
           (if (< IPADDRESS 0)
               then -1
             else 0)
    elseif (EQ (fetch (IPADDRESS CLASSA) of IPADDRESS)
               \IP.CLASS.A)
      then (fetch (IPADDRESS CLASSANET) of IPADDRESS)
    elseif (EQ (fetch (IPADDRESS CLASSB) of IPADDRESS)
               \IP.CLASS.B)
      then (fetch (IPADDRESS CLASSBNET) of IPADDRESS)
    elseif (EQ (fetch (IPADDRESS CLASSC) of IPADDRESS)
               \IP.CLASS.C)
      then (fetch (IPADDRESS CLASSCNET) of IPADDRESS])
```

(\IP.ADDRESS.TO.STRING

```
[LAMBDA (IPADDRESS)                                          (* ejs%: "28-Dec-84 08:43")
  (RESETFORM (RADIX 10)
         (CONCAT (LDB (BYTE 8 24)
                      IPADDRESS)
                 "."
                 (LDB (BYTE 8 16)
                      IPADDRESS)
                 "."
                 (LDB (BYTE 8 8)
                      IPADDRESS)
                 "."
                 (LDB (BYTE 8 0)
                      IPADDRESS])
```

(\IP.BROADCAST.ADDRESS

```
[LAMBDA (IPADDRESS)                                          ; Edited 26-Oct-88 14:59 by bvm
   ;; 0's in the host field are now considered broadcasts, so this code works with Berkeley Unix
```

```
    (LET (HOST MASK)
         (if (SMALLP IPADDRESS)
             then (OR (EQ IPADDRESS 0)
                      (EQ IPADDRESS -1))
           elseif (EQ (fetch (IPADDRESS CLASSA) of IPADDRESS)
                      \IP.CLASS.A)
             then [if (AND \IP.SUBNET.MASKS (ASSOC (fetch (IPADDRESS CLASSANET) of IPADDRESS)
                                                   \IP.LOCAL.NETWORKS))
                      then                                          ; If it's our subnet, check only the subnetted host part.  The
                                                                    ; LOGOR patches bogus subnet masks
                           [SETQ HOST (LOGAND IPADDRESS (SETQ MASK (LOGXOR (LOGOR (CDAR \IP.SUBNET.MASKS)
                                                                                  -16777216)
                                                                           -1]
                           (OR (EQ HOST 0)
                               (EQL HOST MASK))
                    else (SETQ HOST (fetch (IPADDRESS CLASSAHOST) of IPADDRESS))
                         (OR (EQ HOST 0)
                             (EQL HOST (MASK.1'S 0 24]
           elseif (EQ (fetch (IPADDRESS CLASSB) of IPADDRESS)
                      \IP.CLASS.B)
             then [if (AND \IP.SUBNET.MASKS (ASSOC (fetch (IPADDRESS CLASSBNET) of IPADDRESS)
                                                   \IP.LOCAL.NETWORKS))
                      then [SETQ HOST (LOGAND IPADDRESS (SETQ MASK (LOGXOR (LOGOR (CDAR \IP.SUBNET.MASKS)
                                                                                  -65536)
                                                                           -1]
                           (OR (EQ HOST 0)
                               (EQ HOST MASK))
                    else (SETQ HOST (fetch (IPADDRESS CLASSBHOST) of IPADDRESS))
                         (OR (EQ HOST 0)
                             (EQ HOST (MASK.1'S 0 16]
           elseif (EQ (fetch (IPADDRESS CLASSC) of IPADDRESS)
                      \IP.CLASS.C)
             then (SETQ HOST (fetch (IPADDRESS CLASSCHOST) of IPADDRESS))
                                                                    ; No subnetting here
                  (OR (EQ HOST 0)
                      (EQ HOST (MASK.1'S 0 8)))
           elseif (EQ (fetch (IPADDRESS CLASSBNET) of IPADDRESS)
                      MAX.SMALLP)
             then                                                   ; Sort of illegal, but recognize all ones as broadcast
                  (EQ (fetch (IPADDRESS CLASSBHOST) of IPADDRESS)
                      MAX.SMALLP])
```

(**\IP.LEGAL.ADDRESS**

```
  [LAMBDA (ADDRESS)                                                 (* ejs%: "25-Mar-86 16:00")
    (AND (NOT (EQ ADDRESS 0))
         (NOT (EQ ADDRESS -1))
         (OR (EQ \IP.CLASS.C (SETQ ADDRESS (LRSH ADDRESS 29)))
             (EQ \IP.CLASS.B (SETQ ADDRESS (LRSH ADDRESS 1)))
             (EQ \IP.CLASS.A (LRSH ADDRESS 1])
```

(**\IP.MAKE.BROADCAST.ADDRESS**

```
  [LAMBDA (IPADDRESS)                                               (* ejs%: " 3-Jun-85 01:02")
    (SELECTQ (\IPADDRESSCLASS IPADDRESS)
         (\IP.CLASS.A (LOGOR (MASK.1'S 0 24)
                             IPADDRESS))
         (\IP.CLASS.B (LOGOR (MASK.1'S 0 16)
                             IPADDRESS))
         (\IP.CLASS.C (LOGOR (MASK.1'S 0 8)
                             IPADDRESS))
         (SHOULDNT])
```

(**\IP.PRINT.ADDRESS**

```
  [LAMBDA (IPADDRESS FILE)                                          (* ejs%: "28-Dec-84 08:42")
    (RESETFORM (RADIX 10)
           (PRIN1 (LDB (BYTE 8 24)
                       IPADDRESS)
                  FILE)
           (PRIN1 "." FILE)
           (PRIN1 (LDB (BYTE 8 16)
                       IPADDRESS)
                  FILE)
           (PRIN1 "." FILE)
           (PRIN1 (LDB (BYTE 8 8)
                       IPADDRESS)
                  FILE)
           (PRIN1 "." FILE)
           (PRIN1 (LDB (BYTE 8 0)
                       IPADDRESS)
                  FILE)
           IPADDRESS])
```

(**\IP.READ.STRING.ADDRESS**

```
  [LAMBDA (STRING.OR.ATOM)                                          ; Edited 21-Apr-88 14:41 by bvm
```

```
    (for CHAR instring (MKSTRING STRING.OR.ATOM) bind (RESULT _ (NCREATE 'FIXP))
                                                      (INDEX _ 0)
                                                      BYTE
       do (if (> INDEX 3)
              then                                                    ; Got 3 parts and there's still more to go, must be bad
                   (RETURN NIL)
            elseif (EQ CHAR (CHARCODE %.))
              then (if BYTE
                       then (\PUTBASEBYTE RESULT INDEX BYTE))
                   (SETQ BYTE NIL)
                   (add INDEX 1)
            elseif (AND (SETQ CHAR (CL:DIGIT-CHAR-P (CL:INT-CHAR CHAR)))
                        (< (SETQ BYTE (+ (if BYTE
                                             then (TIMES BYTE 10)
                                           else 0)
                                         CHAR))
                           256))
              then                                                    ; Accumulated decimal digit, and we haven't overflowed a byte
                                                                      ; yet
            else                                                      ; Malformed
                 (RETURN NIL))
       finally (if BYTE
                   then (\PUTBASEBYTE RESULT INDEX BYTE)
                        (add INDEX 1))
               (RETURN (AND (EQ INDEX 4)
                            RESULT])
```

(\DOMAIN.NAME.QUALIFY.FULLY

```
  [LAMBDA (NAME)                                                      ; Edited 29-Aug-90 16:27 by gadener
                                                                      (* Make a fully qualified domain name from a partial one)
    (if (OR (NULL INTERNET.LOCAL.DOMAIN)
            (STRPOS "." NAME))
        then NAME
      else (MKATOM (CONCAT NAME "." INTERNET.LOCAL.DOMAIN])

)

;; Startup and shutdown

(RPAQ? *IP-DEFAULT-HOSTS-FILE* )

(RPAQ? TCP.ALWAYS.READ.HOSTS.FILE T)

(RPAQ? \TCP.LAST.HOSTS.FILE.DATE )

(RPAQ? \TCP.LAST.HOSTS.FILE.READ )

(RPAQ? \IPFLG )

(RPAQ? \IP.READY )

(RPAQ? \IP.READY.EVENT (CREATE.EVENT "IP Ready"))

(RPAQ? \IP.WAKEUP.TIMER )

(RPAQ? IPTRACEFLG )

(RPAQ? \IP.WAKEUP.EVENT (CREATE.EVENT "IP Wakeup"))

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \IPFLG \IP.READY \IP.READY.EVENT \IP.WAKEUP.TIMER \IP.WAKEUP.EVENT TCP.ALWAYS.READ.HOSTS.FILE
       \TCP.LAST.HOSTS.FILE.DATE \TCP.LAST.HOSTS.FILE.READ *IP-DEFAULT-HOSTS-FILE*)
)

(DEFINEQ
```

(STOPIP

```
  [LAMBDA NIL                                                         (* ejs%: "28-Dec-84 08:10")
    (\DEL.PACKET.FILTER (FUNCTION \HANDLE.RAW.IP))
    (\DEL.PACKET.FILTER (FUNCTION \HANDLE.RAW.AR))
    (DEL.PROCESS '\IPLISTENER)
    (SETQ \IPFLG (SETQ \IP.READY NIL])
```

(\IPINIT

```
  [LAMBDA (EVENT)                                                     ; Edited 18-Mar-88 17:22 by bvm
    ;; Initialize IP protocol.  Called with EVENT NIL for explicit restart, RESTART from RESTART.ETHER, otherwise from usual around exit events via
    ;; \ETHEREVENTFN and RESTARTETHERFNS after Pup and/an \icmp.echo.reply

    ;; or NS turned on.

    (SELECTQ EVENT
         ((NIL RESTART AFTERSYSOUT AFTERMAKESYS AFTERLOGOUT AFTERSAVEVM)
              (if (AND (NULL \IPFLG)
                       (NOT (NULL EVENT)))
```

```
            then                                                    ; Nothing to do.  Only turn IP on for explicit call to \IPINIT
                 NIL
            elseif [OR (NULL EVENT)
                       (NULL \IP.DEFAULT.CONFIGURATION)
                       (NOT (EQUAL \MY.NSHOSTNUMBER (fetch (IPINIT LOCAL.NSHOSTNUMBER) of
                                                                        \IP.DEFAULT.CONFIGURATION
                                                    ]
            then                                                    ; Machine changed, or caller explicitly wants us to reread the init
                                                                    ; file
                 (SETQ \IP.DEFAULT.CONFIGURATION NIL)
                 (SETQ \IP.LOCAL.ADDRESSES NIL)
                 (SETQ \IP.LOCAL.NETWORKS NIL)
                 (SETQ \IP.SUBNET.MASKS NIL)
                 (DEL.PROCESS '\IPLISTENER)
                 [SELECTQ EVENT
                      ((NIL RESTART)                                ; Can do it here--explicit manual restart.  Otherwise spawn
                                                                    ; process, so that we can do arbitrary things like rely on other
                                                                    ; devices initialized later than ether
                           (\IP.REINITIALIZE.FROM.SCRATCH))
                      (ADD.PROCESS `(\IP.REINITIALIZE.FROM.SCRATCH ',EVENT]
            else (\IP.RESTART.FROM.CONFIGURATION EVENT)))
       NIL])
```

(\IPLISTENER

```
[LAMBDA NIL                                                         (* ejs%: "25-Jun-85 18:52")

        (* * IP background process)

   (SETQ \IP.WAKEUP.TIMER (SETUPTIMER \IP.WAKEUP.INTERVAL))
   (bind [\AR.WAKEUP.TIMER _ (SETUPTIMER (CONSTANT (ITIMES 4 \IP.WAKEUP.INTERVAL] while T
      do (AWAIT.EVENT \IP.WAKEUP.EVENT \IP.WAKEUP.INTERVAL)
         (\IP.CHECK.REASSEMBLY.TIMEOUTS)
         (COND
            ((TIMEREXPIRED? \AR.WAKEUP.TIMER)
             (\AR.DAEMON)
             (SETQ \AR.WAKEUP.TIMER (SETUPTIMER (CONSTANT (ITIMES 4 \IP.WAKEUP.INTERVAL))
                                    \AR.WAKEUP.TIMER])
```

(\IP.REINITIALIZE.FROM.SCRATCH

```
[LAMBDA (EVENT)                                                     ; Edited 20-Jan-89 18:35 by bvm
   (DECLARE (GLOBALVARS \IP.DEFAULT.CONFIGURATION \IP.LOCAL.ADDRESSES))

   ;; Called when we have never enabled IP, or the machine's address has changed.

   (RESETBUFS (PROG (FILE ADDRESS.STRING HOSTS.FILE HOSTNAME ADDRESSES)

                    ;; This is a kludge until we know more about IP routing and reverse address resolution (??)

                    [SETQ \IP.DEFAULT.CONFIGURATION (COND
                                                       ((AND (SETQ FILE (INFILEP '{DSK}IP.INIT))
                                                             (\IP.READ.INIT.FILE FILE)))
                                                       ((IP.DEFAULT.CONFIGURATION))
                                                       ((AND (SETQ FILE (\IP.PROMPT.FOR.FILE.NAME "Please enter
                                                                                the name of the IP
                                                                                initialization file for this
                                                                                host: "))
                                                             (\IP.READ.INIT.FILE FILE)))
                                                       (T       ; User declined to specify, or init file failed, so give up
                                                          (PRINTOUT T "IP not initialized" T)
                                                          (RETURN NIL]
                    (COND
                       ((SETQ FILE (OR (fetch (IPINIT HTE.FILE) of \IP.DEFAULT.CONFIGURATION)
                                       *IP-DEFAULT-HOSTS-FILE*))
                        ;; there is a hosts file in the configuration.  Now see if we really want to read it.
                        (\IP.MAYBE.READ.HOSTS.TXT T FILE)))
                    (COND
                       ([AND (NOT (SETQ HOSTNAME (fetch (IPINIT HOSTNAME) of \IP.DEFAULT.CONFIGURATION)))
                             (SETQ HOSTNAME (AND (EQ \PUP.READY T)
                                                 (U-CASE (ETHERHOSTNAME]
                        (replace (IPINIT HOSTNAME) of \IP.DEFAULT.CONFIGURATION with HOSTNAME)))
                    [COND
                       [(SETQ ADDRESSES (fetch (IPINIT LOCAL.ADDRESSES) of \IP.DEFAULT.CONFIGURATION))
                        (SETQ \IP.LOCAL.ADDRESSES (for ADDR in ADDRESSES collect (\IP.READ.STRING.ADDRESS ADDR]
                       ((AND HOSTNAME (SETQ ADDRESSES (DODIP.HOSTP HOSTNAME)))
                        (SETQ \IP.LOCAL.ADDRESSES (LIST ADDRESSES)))
                       (T (until (SETQ ADDRESS.STRING (PROMPTFORWORD "Please enter this machine's IP host
                                                                     address (e.g. 39.9.0.9)")))
                          (SETQ \IP.LOCAL.ADDRESSES (LIST (\IP.READ.STRING.ADDRESS ADDRESS.STRING)))
                          (COND
                             (HOSTNAME                              ; Associate name with local address(es)
                                    (PUTHASH HOSTNAME [create HOSTS.TXT.ENTRY
                                                              HTE.TYPE _ 'HOST
                                                              HTE.ADDRESSES _ \IP.LOCAL.ADDRESSES
                                                              HTE.NAMES _ (LIST HOSTNAME)
                                                              HTE.MACHINE.TYPE _ (SELECTQ (MACHINETYPE)
                                                                                          (DOVE 'XEROX-1185)
```

```
                                                                    (DANDELION 'XEROX-1108)
                                                                    (DOLPHIN 'XEROX-1100)
                                                                    (DORADO 'XEROX-1132)
                                                                    'XEROX-11XX)
                                                    HTE.OS.TYPE _ 'INTERLISP
                                                    HTE.PROTOCOLS _ '((TCP)
                                                                      (IP]
                                        \IP.HOSTNAMES]
                        (\IP.RESTART.FROM.CONFIGURATION EVENT T])
```

(\IP.RESTART.FROM.CONFIGURATION

```
[LAMBDA (EVENT NEW.INIT)                                    ; Edited 26-Feb-89 21:28 by welch
```

;; Reinitialize IP after logout, etc, from the info in the default configuration. This is the only place that sets \IP.READY true.

```
(GLOBALVARS INTERNET.LOCAL.DOMAIN)
(PROG ((GATE (fetch (IPINIT DEFAULT.GATEWAY) of \IP.DEFAULT.CONFIGURATION))
       (NETS (fetch (IPINIT LOCAL.NETWORKS) of \IP.DEFAULT.CONFIGURATION))
       PROC NDB)
      (SETQ \IP.DEFAULT.GATEWAY (AND GATE (\IP.READ.STRING.ADDRESS GATE)))
      (SETQ \IP.ROUTING.TABLE (CONS))
      (SETQ \AR.IP.TO.10MB.ALIST NIL)
      (SETQ INTERNET.LOCAL.DOMAIN (fetch (IPINIT LOCAL.DOMAIN) of \IP.DEFAULT.CONFIGURATION))
      [COND
         [(EQLENGTH NETS (LENGTH \IP.LOCAL.ADDRESSES))
```

;; List tells net numbers of each directly connected net. Each element = ("net.number" . type).

```
          (SETQ \IP.LOCAL.NETWORKS (bind NDB for NET.AND.TYPE in NETS as ADDRESS in \IP.LOCAL.ADDRESSES
                                      collect (LET* [(TYPE (CDR NET.AND.TYPE))
                                                     [NET (\IPNETADDRESS (\IP.READ.STRING.ADDRESS (CAR
                                                                                                    NET.AND.TYPE
                                                                                                          ]
                                                      (NDB (SELECTQ TYPE
                                                                (3 \3MBLOCALNDB)
                                                                (10 \10MBLOCALNDB)
                                                                (SHOULDNT]
                                                     (replace (NDB NDBIPNET#) of NDB with NET)
                                                     (replace (NDB NDBIPHOST#) of NDB with ADDRESS)
                                                     (CONS NET NDB]
          ((NULL \IP.LOCAL.ADDRESSES)
           (RETURN (CL:WARN "Error in IP init file.  No local host address specified")))
          ((AND (NULL (CDR \IP.LOCAL.ADDRESSES))
                (NULL (fetch (NDB NDBNEXT) of \LOCALNDBS)))     ; Only one address, so it goes with our one net
           [SETQ \IP.LOCAL.NETWORKS (LIST (CONS (\IPNETADDRESS (CAR \IP.LOCAL.ADDRESSES))
                                                (SETQ NDB (OR \10MBLOCALNDB \3MBLOCALNDB]
           (replace (NDB NDBIPNET#) of NDB with (CAAR \IP.LOCAL.NETWORKS))
           (replace (NDB NDBIPHOST#) of NDB with (CAR \IP.LOCAL.ADDRESSES)))
          (T (RETURN (CL:WARN "Error in IP init file.  Network list and local address list do not
                              correlate."]
      [SETQ \IP.SUBNET.MASKS (for LOCALADDR in \IP.LOCAL.ADDRESSES as MASK in (fetch (IPINIT SUBNETMASK)
                                                                                  of \IP.DEFAULT.CONFIGURATION)
                              as NETADDRESS in NETS collect (CONS LOCALADDR (\IP.READ.STRING.ADDRESS
                                                                              (OR MASK (CAR NETADDRESS]
      (COND
         ((BOUNDP '\DOMAIN.NAMESERVERS)
          (\DOMAIN.INIT EVENT)))
      (\ADD.PACKET.FILTER (FUNCTION \HANDLE.RAW.IP))
      (\ADD.PACKET.FILTER (FUNCTION \HANDLE.RAW.AR))
      (SETQ \IPFLG T)
      (\IP.ADD.PROTOCOL \ICMP.PROTOCOL (FUNCTION TRUE)
            (FUNCTION NILL)
            (FUNCTION \ICMP.INPUT))
      (COND
         ((SETQ PROC (FIND.PROCESS '\IPLISTENER))
          (RESTART.PROCESS PROC))
         (T (ADD.PROCESS '(\IPLISTENER)
                   'RESTARTABLE
                   'SYSTEM
                   'AFTEREXIT \IP.READY.EVENT)))
      (if (NOT NEW.INIT)
          then                                              ; Finally, check for new hosts.txt file, but we can do this in
                                                            ; background.  If NEW.INIT, the configuration code has already
                                                            ; read it.
              (ADD.PROCESS '(\IP.MAYBE.READ.HOSTS.TXT T)
                     'AFTEREXIT
                     'DELETE))
      (SETQ \IP.READY T)
      (NOTIFY.EVENT \IP.READY.EVENT)
      (\ICMP.REQUEST.ADDRESS.MASK)
      (RETURN T])
```

(\IP.MAYBE.READ.HOSTS.TXT

```
[LAMBDA (AFTEREXIT FILE)                                    ; Edited 20-Jan-89 11:56 by bvm
```

;; Read the hosts.txt file if it has changed

```
(if AFTEREXIT
```

```
        then                                                    ; Have to wait until all devices are happy
             (until \PROC.READY do (AWAIT.EVENT \PROCESS.AFTEREXIT.EVENT 10000)))
    (LET (FULLNAME)
         (COND
            ((NULL FILE))
            (TCP.ALWAYS.READ.HOSTS.FILE                         ; the user wants us to always read it fresh.
                   (\HTE.READ.FILE FILE))
            ((NULL (SETQ FULLNAME (INFILEP FILE)))
             (CL:FORMAT PROMPTWINDOW "~%%Couldn't find hosts file ~A" FILE))
            ([AND \TCP.LAST.HOSTS.FILE.DATE (STRING-EQUAL FULLNAME \TCP.LAST.HOSTS.FILE.READ)
                  (EQUAL \TCP.LAST.HOSTS.FILE.DATE (GETFILEINFO FILE 'ICREATIONDATE]
                                                                ; the file names and the file write dates are the same, don't
                                                                ; re-read the hosts file.
             NIL)
            (T                                                  ; Haven't read this particular file before, so snarf it
               (\HTE.READ.FILE FILE])
```

(\IP.READ.INIT.FILE

```
  [LAMBDA (FILE)                                                ; Edited 18-Mar-88 18:34 by bvm
    (CL:MULTIPLE-VALUE-BIND (CONFIGURATION CONDITION)
        [IGNORE-ERRORS (LET ((*UPPER-CASE-FILE-NAMES* NIL)
                             (*READTABLE* (FIND-READTABLE "INTERLISP")))
                            (CL:WITH-OPEN-FILE (S FILE)
                                   (READ S]
      (if CONDITION
          then (PRINTOUT T "Failed to read init file because: " CONDITION)
               NIL
        else (LET ((HOST (fetch (IPINIT LOCAL.NSHOSTNUMBER) of CONFIGURATION)))
                  (if (NULL HOST)
                      then                                      ; Old file that doesn't have its processor identification in it
                           (create IPINIT using CONFIGURATION LOCAL.NSHOSTNUMBER _ \MY.NSHOSTNUMBER)
                    elseif (EQUAL HOST \MY.NSHOSTNUMBER)
                      then                                      ; Good, init file for same host
                           CONFIGURATION
                    else (PRINTOUT T FILE " gives configuration for host " (\COERCE.TO.NSADDRESS HOST)
                                 " but this is machine "
                                 (\COERCE.TO.NSADDRESS \MY.NSHOSTNUMBER)
                                 T)
                         NIL))))])
```

(\IP.PROMPT.FOR.FILE.NAME

```
  [LAMBDA (PROMPT DEFAULT)                                      ; Edited 18-Mar-88 18:14 by bvm
    ;; Prompts for a file name from user and returns its full name if it is infilep
    (bind NAME do (if [NULL (SETQ NAME (PROG1 (PROMPTFORWORD PROMPT DEFAULT NIL NIL NIL NIL (CHARCODE (CR)))
                                              (TERPRI]
                      then (RETURN NIL)
                    elseif (SETQ NAME (INFILEP NAME))
                      then (RETURN NAME)
                    else (PRINTOUT T "File not found" T])

)

(ADDTOVAR RESTARTETHERFNS \IPEVENTFN)

;; Early IP reception functions

(DECLARE%: DONTCOPY

;; FOLLOWING DEFINITIONS EXPORTED

(RPAQQ IPADDRESSTYPES
       ((\IP.CLASS.A 0)
        (\IP.CLASS.A.BYTESPEC (BYTE 1 31))
        (\IP.CLASS.A.NET.BYTESPEC (BYTE 8 24))
        (\IP.CLASS.A.HOST.BYTESPEC (BYTE 24 0))
        (\IP.CLASS.B 2)
        (\IP.CLASS.B.BYTESPEC (BYTE 2 30))
        (\IP.CLASS.B.NET.BYTESPEC (BYTE 16 16))
        (\IP.CLASS.B.HOST.BYTESPEC (BYTE 16 0))
        (\IP.CLASS.C 6)
        (\IP.CLASS.C.BYTESPEC (BYTE 3 29))
        (\IP.CLASS.C.NET.BYTESPEC (BYTE 24 8))
        (\IP.CLASS.C.HOST.BYTESPEC (BYTE 8 0))))

(DECLARE%: EVAL@COMPILE

(RPAQQ \IP.CLASS.A 0)

(RPAQ \IP.CLASS.A.BYTESPEC (BYTE 1 31))

(RPAQ \IP.CLASS.A.NET.BYTESPEC (BYTE 8 24))

(RPAQ \IP.CLASS.A.HOST.BYTESPEC (BYTE 24 0))
```

```
(RPAQQ \IP.CLASS.B 2)

(RPAQ \IP.CLASS.B.BYTESPEC (BYTE 2 30))

(RPAQ \IP.CLASS.B.NET.BYTESPEC (BYTE 16 16))

(RPAQ \IP.CLASS.B.HOST.BYTESPEC (BYTE 16 0))

(RPAQQ \IP.CLASS.C 6)

(RPAQ \IP.CLASS.C.BYTESPEC (BYTE 3 29))

(RPAQ \IP.CLASS.C.NET.BYTESPEC (BYTE 24 8))

(RPAQ \IP.CLASS.C.HOST.BYTESPEC (BYTE 8 0))

(CONSTANTS (\IP.CLASS.A 0)
       (\IP.CLASS.A.BYTESPEC (BYTE 1 31))
       (\IP.CLASS.A.NET.BYTESPEC (BYTE 8 24))
       (\IP.CLASS.A.HOST.BYTESPEC (BYTE 24 0))
       (\IP.CLASS.B 2)
       (\IP.CLASS.B.BYTESPEC (BYTE 2 30))
       (\IP.CLASS.B.NET.BYTESPEC (BYTE 16 16))
       (\IP.CLASS.B.HOST.BYTESPEC (BYTE 16 0))
       (\IP.CLASS.C 6)
       (\IP.CLASS.C.BYTESPEC (BYTE 3 29))
       (\IP.CLASS.C.NET.BYTESPEC (BYTE 24 8))
       (\IP.CLASS.C.HOST.BYTESPEC (BYTE 8 0)))
)
)

;; END EXPORTED DEFINITIONS

(RPAQ? \IP.LOCAL.ADDRESSES )

(RPAQ? \IP.SUBNET.MASKS )

(RPAQ? \IP.GATEWAY.FLG )

(RPAQ \IP.ADDRESS.BOX (\CREATECELL \FIXP))

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \IP.LOCAL.ADDRESSES \IP.SUBNET.MASKS \IP.GATEWAY.FLG \IP.ADDRESS.BOX)
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS \IP.FIX.DEST.HOST MACRO [LAMBDA (IP NDB)                (* ejs%: "26-Dec-84 15:07")
                                    (replace (IP IPDESTINATIONHOST) of IP with (ffetch (NDB NDBIPHOST#) of NDB])

(PUTPROPS \IP.FIX.DEST.NET MACRO [LAMBDA (IP NDB)                 (* ejs%: "26-Dec-84 15:08")

        (* * Put the IP net# corresponding to the given NDB into the destination net field of the dest address of the IP packet)

                                  (replace (IP IPDESTINATIONADDRESS) of IP
                                     with (LOGOR (fetch (IP IPDESTINATIONADDRESS) of IP)
                                                 (LLSH (fetch (NDB NDBIPNET#) of NDB)
                                                       (SELECTQ (\IPADDRESSCLASS (fetch (NDB NDBIPHOST#)
                                                                                    of NDB))
                                                            (\IP.CLASS.A 24)
                                                            (\IP.CLASS.B 16)
                                                            (\IP.CLASS.C 8)
                                                            (SHOULDNT])

(PUTPROPS \IP.FIX.SOURCE.HOST MACRO [LAMBDA (IP NDB)              (* ejs%: "26-Dec-84 15:07")
                                      (replace (IP IPSOURCEHOST) of IP with (ffetch (NDB NDBIPHOST#) of NDB])

(PUTPROPS \IP.FIX.SOURCE.NET MACRO [LAMBDA (IP NDB)               (* ejs%: "26-Dec-84 15:08")

        (* * Put the IP net# corresponding to the given NDB into the destination net field of the dest address of the IP packet)

                                    (replace (IP IPSOURCENET) of IP with (ffetch (NDB NDBIPNET#) of NDB])
)

(DEFINEQ

(\HANDLE.RAW.IP
  [LAMBDA (IP TYPE)                                               (* ejs%: " 3-Feb-86 11:01")
    (PROG ((NDB (ffetch (ETHERPACKET EPNETWORK) of IP)))
          (COND
             ((NOT (type? NDB NDB))
              (ERROR "No NDB in ETHERPACKET!" IP)))
          (SELECTQ (ffetch (NDB NETTYPE) of NDB)
               (10 (COND
                      ((NEQ TYPE \EPT.IP)
```

```
                    (RETURN))))
              (3 (COND
                    ((NEQ TYPE \EET.IP)
                     (RETURN))))
              (ERROR "Unknown net type" (ffetch (NDB NETTYPE) of NDB)))
        [COND
           ((NOT \IP.READY)
            (\RELEASE.ETHERPACKET IP))
           ([NOT (\IP.CHECKSUM.OK (\IPCHECKSUM IP (ffetch (IP IPBASE) of IP)
                                        (TIMES (ffetch (IP IPHEADERLENGTH) of IP)
                                               BYTESPERCELL]
            (AND IPTRACEFLG (PRINTPACKET IP 'GET IPTRACEFILE "[Packet dropped--bad IP header checksum]"))
            (\RELEASE.ETHERPACKET IP))
           ((ZEROP (ffetch (IP IPTIMETOLIVE) of IP))
            (\ICMP.TIME.EXCEEDED IP \ICMP.TRANSIT.TIME.EXCEEDED)
            (\RELEASE.ETHERPACKET IP))
           ((\IP.PROCESS.OPTIONS IP)
            (COND
               ((NOT (\IP.LOCAL.DESTINATION IP))
                (\FORWARD.IP IP))
               [(\IP.FRAGMENTED.PACKET IP)
                (COND
                   ((SETQ IP (\HANDLE.RAW.IP.FRAGMENT IP))
                    [COND
                       (IPTRACEFLG (COND
                                      ((EQ IPTRACEFLG T)
                                       (PRINTIP IP 'GETFRAGMENT IPTRACEFILE NIL T))
                                      (T (PRIN1 "+" IPTRACEFILE]
                    (\IP.HAND.TO.PROTOCOL IP]
               (T [COND
                     (IPTRACEFLG (COND
                                    ((EQ IPTRACEFLG T)
                                     (PRINTIP IP 'GET IPTRACEFILE NIL T))
                                    (T (PRIN1 "+" IPTRACEFILE]
                  (\IP.HAND.TO.PROTOCOL IP]
        (RETURN T])
```

(\FORWARD.IP

```
[LAMBDA (IP)                                                    (* ejs%: "10-Feb-86 11:32")
   (DECLARE (GLOBALVARS \IP.GATEWAY.FLG \IP.GATEWAY.FORWARDING.FUNCTIONS))
   (COND
      [\IP.GATEWAY.FLG (LET* ((DESTADDRESS (ffetch (IP IPDESTINATIONADDRESS) of IP))
                              (NETADDRESS (\IPNETADDRESS DESTADDRESS))
                              (NDB (fetch (ETHERPACKET EPNETWORK) of IP))
                              (SOURCEADDRESS (fetch NDBIPHOST# of NDB))
                              (SUBNETMASK (CDR (SASSOC SOURCEADDRESS \IP.SUBNET.MASKS)))
                              SUBNETINUSE ROUTE FORWARDING.FUNCTION)
                             [COND
                                [(AND NDB SUBNETMASK (OR (EQP (LOGAND SOURCEADDRESS SUBNETMASK)
                                                              (LOGAND DESTADDRESS SUBNETMASK))
                                                         (PROGN (SETQ SUBNETINUSE T)
                                                                NIL]
                                ((NULL NDB)
                                 (COND
                                    ((SETQ ROUTE (CDR (SASSOC NETADDRESS \IP.ROUTING.TABLE)))
                                     (SETQ NDB (CDR (SASSOC (\IPNETADDRESS ROUTE)
                                                            \IP.LOCAL.NETWORKS]
                             (COND
                                [NDB (replace EPREQUEUE of IP with 'FREE)
                                     (add (ffetch (IP IPTIMETOLIVE) of IP)
                                          -1)
                                     [SETQ NETADDRESS (COND
                                                         (SUBNETINUSE (LOGAND DESTADDRESS SUBNETMASK))
                                                         (T (BITCLEAR DESTADDRESS (\IPHOSTADDRESS DESTADDRESS]
                                     (COND
                                        ((SETQ FORWARDING.FUNCTION (CDR (SASSOC NETADDRESS
                                                                                \IP.GATEWAY.FORWARDING.FUNCTIONS
                                                                                )))
                                         (APPLY* FORWARDING.FUNCTION IP NDB NETADDRESS ROUTE))
                                        (T (\RELEASE.ETHERPACKET IP]
                                (T (\ICMP.REDIRECT IP \ICMP.REDIRECT.NET]
      (T (\RELEASE.ETHERPACKET IP])
```

(\IP.LOCAL.DESTINATION

```
[LAMBDA (IP)                                                    (* ejs%: "25-Mar-86 16:03")

          (* * Return T if IP packet is destined for us)

   (UNINTERRUPTABLY
       (\BLT \IP.ADDRESS.BOX (LOCF (fetch (IP IPDESTINATIONADDRESS) of IP))
             WORDSPERCELL)
       [LET [(LOCALNETADDRESS (fetch NDBIPNET# of (fetch EPNETWORK of IP]
            (COND
               ((MEMBER \IP.ADDRESS.BOX \IP.LOCAL.ADDRESSES)
                T)
```

```
                ((AND (\IP.BROADCAST.ADDRESS \IP.ADDRESS.BOX)
                       (EQP LOCALNETADDRESS (\IPNETADDRESS \IP.ADDRESS.BOX)))
                 T)
                ((NOT (\IP.LEGAL.ADDRESS \IP.ADDRESS.BOX))        (* Bogus destination address)
                 NIL)
                ((EQP 0 (\IPNETADDRESS \IP.ADDRESS.BOX))          (* Source doesn't know its network?)
                 (SELECTQ (INTEGERLENGTH LOCALNETADDRESS)
                      (8 (\PUTBASEBYTE \IP.ADDRESS.BOX 0 LOCALNETADDRESS))
                      (16 (\PUTBASE \IP.ADDRESS.BOX 0 LOCALNETADDRESS))
                      (24 [for I from 0 to 2 do (\PUTBASEBYTE \IP.ADDRESS.BOX I
                                                    (LOGAND 255 (LRSH LOCALNETADDRESS (ITIMES 8
                                                                                              (IDIFFERENCE 2 I])
                      NIL)
                 (COND
                    ((\IP.BROADCAST.ADDRESS \IP.ADDRESS.BOX)
                     T)
                    ((MEMBER \IP.ADDRESS.BOX \IP.LOCAL.ADDRESSES)
                     T])])
```

(\IPCHECKSUM

```
  [LAMBDA (ETHERPACKET CHECKSUMBASE NBYTES IGNOREDWORD)          (* ejs%: "31-Dec-84 13:53")
```

(* * Compute a general checksum for a packet starting at CHECKSUMBASE and extending NBYTES.
If NBYTES is odd, a 0 byte is padded on the end. The IGNOREDWORD field is the LOCF of the field which will contain the checksum, and is to be considered 0 for the calculation.)

```
    (PROG ((MAXINDEX (SUB1 (FOLDHI NBYTES BYTESPERWORD)))
           (CHECKSUM 0)
           (ODDFLG (ODDP NBYTES))
           DIFF WORDCONTENTS)
          (AND IGNOREDWORD (\PUTBASE IGNOREDWORD 0 0))
          [for WORD from 0 to MAXINDEX do (SETQ CHECKSUM (COND
                                                           [(AND ODDFLG (EQ WORD MAXINDEX))
                                                            (COND
                                                               ([ILEQ CHECKSUM
                                                                      (SETQ DIFF
                                                                       (IDIFFERENCE MAX.SMALL.INTEGER
                                                                              (SETQ WORDCONTENTS
                                                                               (LOGAND (\GETBASE CHECKSUMBASE
                                                                                               WORD)
                                                                                       (MASK.1'S 8 8]
                                                                (IPLUS CHECKSUM WORDCONTENTS))
                                                               (T (IDIFFERENCE CHECKSUM DIFF]
                                                           (T (COND
                                                                 ([ILEQ CHECKSUM (SETQ DIFF
                                                                                   (IDIFFERENCE MAX.SMALL.INTEGER
                                                                                          (SETQ WORDCONTENTS
                                                                                           (\GETBASE CHECKSUMBASE
                                                                                             WORD]
                                                                  (IPLUS CHECKSUM WORDCONTENTS))
                                                                 (T (IDIFFERENCE CHECKSUM DIFF]
          (RETURN CHECKSUM])
```

(\IP.CHECKSUM.OK

```
  [LAMBDA (CHECKSUM)                                             (* ejs%: "28-Dec-84 19:40")
    (OR (EQ CHECKSUM (MASK.1'S 0 16))
        (EQ CHECKSUM 0])
```

(\IP.SET.CHECKSUM

```
  [LAMBDA (PACKET CHECKSUMBASE NBYTES CHECKSUMWORD)              (* ejs%: " 4-Jun-85 22:47")
    (PROG ((CHECKSUM (\IPCHECKSUM PACKET CHECKSUMBASE NBYTES CHECKSUMWORD)))
          (\PUTBASE CHECKSUMWORD 0 (COND
                                     ((EQ CHECKSUM (MASK.1'S 0 16))
                                      CHECKSUM)
                                     (T (LOGAND (LOGNOT CHECKSUM)
                                                (MASK.1'S 0 16])

)

;; Protocol Distribution

(DECLARE%: DONTCOPY

;; FOLLOWING DEFINITIONS EXPORTED

(RPAQQ IPPROTOCOLTYPES ((\ICMP.PROTOCOL 1)
                        (\TCP.PROTOCOL 6)
                        (\UDP.PROTOCOL 17)))

(DECLARE%: EVAL@COMPILE

(RPAQQ \ICMP.PROTOCOL 1)
```

```
(RPAQQ \TCP.PROTOCOL 6)

(RPAQQ \UDP.PROTOCOL 17)

(CONSTANTS (\ICMP.PROTOCOL 1)
       (\TCP.PROTOCOL 6)
       (\UDP.PROTOCOL 17))
)
)

;; END EXPORTED DEFINITIONS

(RPAQ? \IP.PROTOCOLS )

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \IP.PROTOCOLS)
)

(DEFINEQ
```

(\IP.HAND.TO.PROTOCOL

```
  [LAMBDA (IP)                                                    (* ejs%: "31-Mar-86 15:39")
    (PROG ((PROTOCOL (ffetch (IP IPPROTOCOL) of IP))
           PROTOCOLCHAIN IPSOCKET)
          (COND
             ((NOT (SETQ PROTOCOLCHAIN (\IP.FIND.PROTOCOL PROTOCOL \IP.PROTOCOLS)))
              (OR (\IP.BROADCAST.ADDRESS (fetch (IP IPDESTINATIONADDRESS) of IP))
                  (\ICMP.DEST.UNREACHABLE IP \ICMP.PROTOCOL.UNREACHABLE)))
             ((NOT (SETQ IPSOCKET (\IP.FIND.PROTOCOL.SOCKET IP PROTOCOLCHAIN)))
              (APPLY* (ffetch (IPSOCKET IPSNOSOCKETFN) of PROTOCOLCHAIN)
                      IP))
             (T (APPLY* (ffetch (IPSOCKET IPSINPUTFN) of (COND
                                                           ((type? IPSOCKET IPSOCKET)
                                                            IPSOCKET)
                                                           (T PROTOCOLCHAIN)))
                        IP IPSOCKET])
```

(\IP.DEFAULT.INPUTFN

```
  [LAMBDA (IP IPSOCKET)                                           (* ejs%: " 3-Feb-85 19:19")
    (COND
       ((EQ (fetch (IPSOCKET IPSQUEUELENGTH) of IPSOCKET)
            (fetch (IPSOCKET IPSQUEUEALLOC) of IPSOCKET))
        (\RELEASE.ETHERPACKET IP))
       (T (UNINTERRUPTABLY
             (\ENQUEUE (fetch (IPSOCKET IPSQUEUE) of IPSOCKET)
                       IP)
             (add (fetch (IPSOCKET IPSQUEUELENGTH) of IPSOCKET)
                  1)
             (NOTIFY.EVENT (fetch (IPSOCKET IPSEVENT) of IPSOCKET)))])
```

(\IP.DEFAULT.NOSOCKETFN

```
  [LAMBDA (IP)                                                    (* ejs%: " 2-Feb-86 11:38")
    (COND
       ([OR (NEQ 0 (fetch (IP IPDESTINATIONHOST) of IP))
            (NOT (\IP.BROADCAST.ADDRESS (fetch (IP IPDESTINATIONADDRESS) of IP]
        (\ICMP.DEST.UNREACHABLE IP \ICMP.PORT.UNREACHABLE))
       (T (\RELEASE.ETHERPACKET IP])
```

(\IP.ADD.PROTOCOL

```
  [LAMBDA (PROTOCOL SOCKETCOMPAREFN NOSOCKETFN INPUTFN ICMPFN)    ; Edited 25-Aug-88 12:10 by bvm

;;; Find an existing protocol, or create a new one, and return the socket chain head.  If the protocol already exists, the remaining arguments redefine the
;;; current slots.

    (LET* [(FOUND (find SOCKET in \IP.PROTOCOLS suchthat (EQ (fetch (IPSOCKET PROTOCOL) of SOCKET)
                                                             PROTOCOL)))
           (SOCKET (OR FOUND (create IPSOCKET
                                     PROTOCOL _ PROTOCOL
                                     IPSQUEUE _ NIL
                                     IPSQUEUEALLOC _ 0
                                     IPSEVENT _ NIL]
          (replace (IPSOCKET IPSDESTSOCKETCOMPAREFN) of SOCKET with SOCKETCOMPAREFN)
          (replace (IPSOCKET IPSINPUTFN) of SOCKET with (OR INPUTFN (FUNCTION \IP.DEFAULT.INPUTFN)))
          (replace (IPSOCKET IPSNOSOCKETFN) of SOCKET with (OR NOSOCKETFN (FUNCTION \IP.DEFAULT.NOSOCKETFN)))
          (replace (IPSOCKET IPSICMPFN) of SOCKET with (OR ICMPFN (FUNCTION \RELEASE.ETHERPACKET)))
          (if (NOT FOUND)
              then                                                ; Now that it's all filled in, add it to the protocol set
                   (push \IP.PROTOCOLS SOCKET))
          SOCKET])
```

(\IP.DELETE.PROTOCOL

```
  [LAMBDA (PROTOCOL)                                              (* ejs%: "10-Apr-85 16:24")
```

```
    (LET ((PROTOCOLCHAIN (\IP.FIND.PROTOCOL PROTOCOL)))
         (COND
            (PROTOCOLCHAIN (until (NULL (fetch (IPSOCKET IPSLINK) of PROTOCOLCHAIN))
                                  do (\IP.CLOSE.SOCKET (fetch (IPSOCKET IPSOCKET) of (fetch (IPSOCKET IPSLINK)
                                                                                       of PROTOCOLCHAIN))
                                                       PROTOCOL))
                           (SETQ \IP.PROTOCOLS (DREMOVE PROTOCOLCHAIN \IP.PROTOCOLS))
                           T])
```

(\IP.FIND.PROTOCOL

```
  [LAMBDA (PROTOCOL)                                                    (* ejs%: "27-Dec-84 11:18")

          (* * Find the protocol chain for this protocol#)

    (CAR (SOME \IP.PROTOCOLS (FUNCTION (LAMBDA (IPSOCKET)
                                         (EQ (ffetch (IPSOCKET PROTOCOL) of IPSOCKET)
                                             PROTOCOL])
```

(\IP.FIND.PROTOCOL.SOCKET

```
  [LAMBDA (IP PROTOCOLCHAIN)                                            ; Edited 26-Aug-88 12:44 by bvm
    ;; Find the socket specified by IP packet.  PROTOCOLCHAIN is the head of the socket chain for this protocol; if NIL we look it up.
    (LET ([SOCKET (OR PROTOCOLCHAIN (\IP.FIND.PROTOCOL (ffetch (IP IPPROTOCOL) of IP]
          RESULT)
         ;; Note that we start the comparisons with the dummy head, even though we expect that to fail.  This is so that a socketless protocol, such
         ;; as ICMP can use this dummy head as the sole handler of the protocol.
         (AND SOCKET (when (SETQ RESULT (CL:FUNCALL (ffetch (IPSOCKET IPSDESTSOCKETCOMPAREFN) of SOCKET)
                                                    IP SOCKET))
                     do (RETURN (COND
                                   ((EQ RESULT T)
                                    SOCKET)
                                   (T                                   ; This is a little strange.  Non-T comparison result will be passed
                                                                        ; as the second arg to the chain head's inputfn when a packet
                                                                        ; arrives here.
                                      RESULT)))
                     repeatwhile (SETQ SOCKET (ffetch (IPSOCKET IPSLINK) of SOCKET])
```

(\IP.FIND.SOCKET

```
  [LAMBDA (SOCKET# SOCKETCHAIN)                                         (* ejs%: "27-Dec-84 11:39")

          (* * Called to find the socket open on the socketchain, or NIL if no such open socket.
          Socketchain comes from \IP.FIND.PROTOCOL)

    (while SOCKETCHAIN until (COND
                               ((EQUAL SOCKET# (ffetch (IPSOCKET IPSOCKET) of SOCKETCHAIN))
                                SOCKETCHAIN)
                               (T (SETQ SOCKETCHAIN (ffetch (IPSOCKET IPSLINK) of SOCKETCHAIN))
                                  NIL))
       finally (RETURN SOCKETCHAIN])
```

(\IP.OPEN.SOCKET

```
  [LAMBDA (PROTOCOL SOCKET NOERRORFLG DESTSOCKETCOMPAREFN NOSOCKETFN INPUTFN ICMPFN)
                                                                        ; Edited 25-Aug-88 12:43 by bvm

;;; Open a new socket for a protocol.  The last 4 fns default to those specified when the protocol was enabled.
    ;; Keeping NOSOCKETFN for back compatibility, but it doesn't really make any sense --bvm.
    (LET ((MASTERSOC (\IP.FIND.PROTOCOL PROTOCOL))
          OLDSOC NEWSOC)
         (COND
            [(NOT (type? IPSOCKET MASTERSOC))
             (COND
                ((NOT NOERRORFLG)
                 (ERROR "Attempt to open socket in unknown protocol" PROTOCOL SOCKET]
            [(if SOCKET
                 then (SETQ OLDSOC (\IP.FIND.SOCKET SOCKET MASTERSOC))
               else                                                     ; Pick a random socket that is smallp but not very small, so as to
                                                                        ; avoid well-known sockets
                    (SETQ SOCKET (LOGOR (LOGAND (DAYTIME)
                                                65535)
                                        32768))
                    (while (\IP.FIND.SOCKET SOCKET MASTERSOC) do (SETQ SOCKET (- SOCKET 1)))
                    NIL)
             (COND
                (NOERRORFLG OLDSOC)
                (T (ERROR "Attempt to open an existing socket" OLDSOC]
            (T [SETQ NEWSOC (create IPSOCKET
                                    IPSLINK _ (ffetch (IPSOCKET IPSLINK) of MASTERSOC)
                                    IPSOCKET _ SOCKET
                                    PROTOCOL _ PROTOCOL
                                    IPSDESTSOCKETCOMPAREFN _ (OR DESTSOCKETCOMPAREFN (ffetch (IPSOCKET
                                                                                              IPSDESTSOCKETCOMPAREFN
```

```
                                                                                    )
                                                                              of MASTERSOC))
                                        IPSNOSOCKETFN _ (OR NOSOCKETFN (ffetch (IPSOCKET IPSNOSOCKETFN) of MASTERSOC))
                                        IPSINPUTFN _ (OR INPUTFN (ffetch (IPSOCKET IPSINPUTFN) of MASTERSOC))
                                        IPSICMPFN _ (OR ICMPFN (ffetch (IPSOCKET IPSICMPFN) of MASTERSOC]
                (freplace (IPSOCKET IPSLINK) of MASTERSOC with NEWSOC)
                NEWSOC])
```

(\IP.CLOSE.SOCKET

```
  [LAMBDA (SOCKET PROTOCOL NOERRORFLG)                    ; Edited 26-Aug-88 12:33 by bvm
```

;;; Close the given socket. Call this only after the higher level protocol has finished doing its closing operations.

;; For some silly reason, this fn was defined to take not an IPSOCKET object but rather the socket number, or whatever was in the socket slot. For ;; backward compatibility, let's do both (sigh).

```
    (LET ((PREV (\IP.FIND.PROTOCOL PROTOCOL))
          NEXT)
         (COND
            [(AND PREV (while (SETQ NEXT (ffetch (IPSOCKET IPSLINK) of PREV))
                          do (if (OR (EQ SOCKET NEXT)
                                     (EQ SOCKET (ffetch (IPSOCKET IPSOCKET) of NEXT)))
                                 then                          ; Found it, so splice it out
                                      (freplace (IPSOCKET IPSLINK) of PREV with (ffetch (IPSOCKET IPSLINK)
                                                                                   of NEXT))
                                      (freplace (IPSOCKET IPSLINK) of NEXT with NIL)
                                      (RETURN T))
                               (SETQ PREV NEXT]
            ((NOT NOERRORFLG)
             (ERROR "Socket not found" SOCKET])

)

;; Fragmentation Handling

(DECLARE%: DONTCOPY

;; FOLLOWING DEFINITIONS EXPORTED

(DECLARE%: EVAL@COMPILE

(RECORD AssemblyRecord (Packet FirstHole Fragments Timeout)
       Packet _ (\ALLOCATE.ETHERPACKET)
       FirstHole _ 0)

(RECORD FragmentRecord (Start Length LastFragment))

(RECORD FragmentID (AssemblyRecord SourceAddress ID Protocol . DestinationAddress))
)
)

;; END EXPORTED DEFINITIONS

(RPAQ? \IP.FRAGMENT.LIST )

(RPAQ? \IP.FRAGMENT.LOCK (CREATE.MONITORLOCK "IP Fragment Processing Lock"))

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \IP.FRAGMENT.LIST \IP.FRAGMENT.LOCK)
)

(DECLARE%: EVAL@COMPILE

(RPAQQ \IP.FRAGMENTATION.UNIT 8)

(CONSTANTS (\IP.FRAGMENTATION.UNIT 8))
)

(DEFINEQ
```

(\HANDLE.RAW.IP.FRAGMENT

```
  [LAMBDA (IP)                                            (* ejs%: " 1-Feb-86 14:24")
```

(* * Add the next fragment to a packet under assembly. If this fragment completes a packet, return the completed packet to be processed by higher-level protocol routines.)

```
    (WITH.MONITOR \IP.FRAGMENT.LOCK
        (LET ((AssemblyRecord (\IP.FIND.MATCHING.FRAGMENTS IP)))
             (COND
                (AssemblyRecord (\IP.ADD.FRAGMENT AssemblyRecord IP))
                (T (\IP.NEW.FRAGMENT.LST IP)
                   NIL)))))])
```

(\IP.NEW.FRAGMENT.LST

```
  [LAMBDA (IP)                                                    (* ejs%: " 3-Feb-86 10:57")

          (* * Add a new fragment to the fragment list)

    (PROG ((Source (ffetch (IP IPSOURCEADDRESS) of IP))
           (Dest (ffetch (IP IPDESTINATIONADDRESS) of IP))
           (Protocol (ffetch (IP IPPROTOCOL) of IP))
           (ID (ffetch (IP IPID) of IP))
           NewFragmentID FragmentRecord AssemblyPacket AssemblyRecord)
          [SETQ NewFragmentID (create FragmentID
                                      SourceAddress _ Source
                                      ID _ ID
                                      Protocol _ Protocol
                                      DestinationAddress _ Dest
                                      AssemblyRecord _ (SETQ AssemblyRecord
                                                             (create AssemblyRecord
                                                                     Timeout _ (SETUPTIMER (ITIMES 1000
                                                                                                   (ffetch (IP
                                                                                                            IPTIMETOLIVE
                                                                                                            )
                                                                                                      of IP)))
                                                                     Fragments _
                                                                     (LIST (SETQ FragmentRecord
                                                                                 (create FragmentRecord
                                                                                         Start _ (UNFOLD (ffetch (IP
                                                                                                                  IPFRAGMENTOFFSET
                                                                                                                  )
                                                                                                            of IP)

                                                                                                          \IP.FRAGMENTATION.UNIT
                                                                                                          )
                                                                                         Length _
                                                                                         (IDIFFERENCE
                                                                                          (ffetch (IP IPTOTALLENGTH)
                                                                                             of IP)
                                                                                          (UNFOLD (ffetch (IP IPHEADERLENGTH)
                                                                                                     of IP)
                                                                                                BYTESPERCELL]
          (COND
             ((EQ IPTRACEFLG T)
              (\IP.PRINT.FRAGMENT NewFragmentID IP IPTRACEFILE)))
          (SETQ AssemblyPacket (fetch (AssemblyRecord Packet) of AssemblyRecord))
          (\IP.COPY.FRAGMENT.HEADER.TO.PACKET.HEADER AssemblyPacket IP)

          (* * Copy the packet data to the packet)

          (\BLT (\ADDBASE (\IPDATABASE AssemblyPacket)
                          (FOLDLO (fetch (FragmentRecord Start) of FragmentRecord)
                                  BYTESPERWORD))
                (\IPDATABASE IP)
                (FOLDLO (fetch (FragmentRecord Length) of FragmentRecord)
                        BYTESPERWORD))
          (\RELEASE.ETHERPACKET IP)
          (push \IP.FRAGMENT.LIST NewFragmentID])
```

(\IP.COPY.FRAGMENT.HEADER.TO.PACKET.HEADER

```
  [LAMBDA (Packet Fragment)                                       (* ejs%: " 1-Feb-86 14:14")

          (* * Copy information from the header of the fragment packet into the header of the reassembled packet)

    (\MOVEBYTES (fetch (IP IPBASE) of Fragment)
           0
           (fetch (IP IPBASE) of Packet)
           0
           (UNFOLD (fetch (IP IPHEADERLENGTH) of Fragment)
                  BYTESPERCELL])
```

(\IP.ADD.FRAGMENT

```
  [LAMBDA (FragmentID NewIP)                                      (* ejs%: " 1-Feb-86 18:41")

          (* * Called to add a fragment to a fragment list. The fragment is added in order.
          If the fragment completes a fragmented IP packet, a new packet is assembled and returned, else NIL is returned)

    (LET* ((AssemblyRecord (fetch (FragmentID AssemblyRecord) of FragmentID))
           [NewFrag (create FragmentRecord
                            Start _ (UNFOLD (ffetch (IP IPFRAGMENTOFFSET) of NewIP)
                                            \IP.FRAGMENTATION.UNIT)
                            Length _ (IDIFFERENCE (ffetch (IP IPTOTALLENGTH) of NewIP)
                                                  (UNFOLD (ffetch (IP IPHEADERLENGTH) of NewIP)
                                                          BYTESPERCELL))
                            LastFragment _ (NOT (fetch (IP IPMOREFRAGMENTS) of NewIP]
           (Fragments (fetch (AssemblyRecord Fragments) of AssemblyRecord))
           Status NextHole AssemblyPacket)
          (COND
             ((EQ IPTRACEFLG T)
```

```
            (\IP.PRINT.FRAGMENT FragmentID NewIP IPTRACEFILE)))
        (SETQ AssemblyPacket (fetch (AssemblyRecord Packet) of AssemblyRecord))
        (replace (AssemblyRecord Timeout) of AssemblyRecord with (SETUPTIMER (ITIMES 1000 (ffetch (IP IPTIMETOLIVE
                                                                                                    )
                                                                                          of NewIP))
                                                                                (fetch (AssemblyRecord Timeout)
                                                                                   of AssemblyRecord)))
        [SETQ Status (COND
                        ((ILESSP (fetch (FragmentRecord Start) of NewFrag)
                                 (fetch (FragmentRecord Start) of (CAR Fragments)))
                                                                  (* Earlier than the earliest existing fragment)
                         (SETQ Fragments (push (fetch (AssemblyRecord Fragments) of AssemblyRecord)
                                               NewFrag))
                         'INSERTED.FRAGMENT)
                        ((EQ (fetch (FragmentRecord Start) of NewFrag)
                             (fetch (FragmentRecord Start) of (CAR Fragments)))
                                                                  (* Duplicate of earliest fragment)
                         'DUPLICATE)
                        (T                                        (* Have to search)
                           (for OldFragTail on Fragments while (CDR OldFragTail)
                              thereis (COND
                                         ((EQ (fetch (FragmentRecord Start) of NewFrag)
                                              (fetch (FragmentRecord Start) of (CADR OldFragTail)))
                                                                  (* Duplicate)
                                          (SETQ Status 'DUPLICATE)
                                          T)
                                         ((ILESSP (fetch (FragmentRecord Start) of NewFrag)
                                                  (fetch (FragmentRecord Start) of (CADR OldFragTail)))
                                                                  (* Found the hole to insert)
                                          T))
                              finally (COND
                                         (Status                  (* Duplicate)
                                                 (RETURN Status))
                                         ((CDR OldFragTail)       (* Inserted in middle of list)
                                          (RPLACD OldFragTail (CONS NewFrag (CDR OldFragTail)))
                                          (RETURN 'INSERTED.FRAGMENT))
                                         (T                       (* Inserted at end of list)
                                            (NCONC1 OldFragTail NewFrag)
                                            (RETURN 'INSERTED.FRAGMENT]
        (PROG1 (SELECTQ Status
                    (DUPLICATE NIL)
                    (INSERTED.FRAGMENT                            (* Copy bytes into assembly)
                         (\MOVEBYTES (\IPDATABASE NewIP)
                                0
                                (\IPDATABASE AssemblyPacket)
                                (fetch (FragmentRecord Start) of NewFrag)
                                (fetch (FragmentRecord Length) of NewFrag))
                         (add (ffetch (IP IPTOTALLENGTH) of AssemblyPacket)
                              (fetch (FragmentRecord Length) of NewFrag))
                                                                  (* Update Assembly record)
                         [COND
                            ((ILESSP (fetch (FragmentRecord Start) of NewFrag)
                                     (fetch (AssemblyRecord FirstHole) of AssemblyRecord))
                             (ERROR "Error in IP fragment reassembly!" NewFrag))
                            (T (COND
                                  ((EQ [bind End Status for FragTail on Fragments while (CDR FragTail)
                                       thereis [COND
                                                  ((NEQ [SETQ End (IPLUS (fetch (FragmentRecord Start)
                                                                                of (CAR FragTail))
                                                                         (fetch (FragmentRecord Length)
                                                                            of (CAR FragTail]
                                                        (fetch (FragmentRecord Start) of (CADR FragTail)))
                                                   (replace (AssemblyRecord FirstHole) of AssemblyRecord
                                                      with End)
                                                   (SETQ Status 'FOUND.HOLE]
                                       finally (RETURN (COND
                                                          [(NULL Status)
                                                           (COND
                                                              ((fetch (FragmentRecord LastFragment)
                                                                  of (CAR FragTail))
                                                               (COND
                                                                  ((EQ IPTRACEFLG T)
                                                                   (printout IPTRACEFILE T "Complete IP
                                                                           Fragment received" T)))
                                                               'COMPLETE.PACKET)
                                                              (T (replace (AssemblyRecord FirstHole)
                                                                    of AssemblyRecord with End)
                                                                 'INCOMPLETE.BUT.NO.HOLES]
                                                          (T Status]
                                     'COMPLETE.PACKET)
                                  (\IP.DELETE.FRAGMENT FragmentID)
                                  AssemblyPacket])
                    NIL)
                 (\RELEASE.ETHERPACKET NewIP])
```

(\IP.FIND.MATCHING.FRAGMENTS

```
[LAMBDA (IP)                                                    (* ejs%: " 1-Feb-86 14:41")

        (* * Find the list of fragments matching this IP packet, or NIL if none exists)

   (DECLARE (GLOBALVARS \IP.FRAGMENT.LIST))
   (LET* ((Source (ffetch (IP IPSOURCEADDRESS) of IP))
          (Dest (ffetch (IP IPDESTINATIONADDRESS) of IP))
          (Protocol (ffetch (IP IPPROTOCOL) of IP))
          (ID (ffetch (IP IPID) of IP))
          (FragmentEntry))
         (for FragmentID in \IP.FRAGMENT.LIST thereis (AND (EQP (fetch (FragmentID SourceAddress) of FragmentID)
                                                                Source)
                                                           (EQ (fetch (FragmentID ID) of FragmentID)
                                                               ID)
                                                           (EQ (fetch (FragmentID Protocol) of FragmentID)
                                                              Protocol)
                                                           (EQP (fetch (FragmentID DestinationAddress) of
                                                                                                   FragmentID
                                                                       )
                                                                Dest])
```

(\IP.FRAGMENTED.PACKET

```
[LAMBDA (IP)                                                    (* ejs%: " 1-Feb-86 16:50")

        (* * Return T if IP packet is a fragment)

   (OR (ffetch (IP IPMOREFRAGMENTS) of IP)
       (NEQ 0 (ffetch (IP IPFRAGMENTOFFSET) of IP])
```

(\IP.CHECK.REASSEMBLY.TIMEOUTS

```
[LAMBDA NIL                                                     (* ejs%: " 3-Feb-86 11:00")

        (* * Kill any fragments in the process of reassembly if their timeout has expired.
        Report timeout via ICMP)

   (WITH.MONITOR \IP.FRAGMENT.LOCK
       (bind AssemblyRecord for Fragment in \IP.FRAGMENT.LIST when [TIMEREXPIRED? (fetch (AssemblyRecord Timeout)
                                                                                      of (SETQ AssemblyRecord
                                                                                           (fetch (FragmentID
                                                                                                   AssemblyRecord
                                                                                                  )
                                                                                             of Fragment]
         do (COND
               ((EQ IPTRACEFLG T)
                (printout IPTRACEFILE T "IP Fragment timeout expired" T)))
            (\ICMP.TIME.EXCEEDED (fetch (AssemblyRecord Packet) of AssemblyRecord)
                   \ICMP.FRAGMENT.TIME.EXCEEDED)
            (\IP.DELETE.FRAGMENT Fragment T)))])
```

(\IP.DELETE.FRAGMENT

```
[LAMBDA (FragmentID FreePacketToo)                              (* ejs%: " 3-Feb-86 10:59")

        (* * Delete FragmentID from the list of Fragment ID's)

   (PROG [(IP (fetch (AssemblyRecord Packet) of (fetch (FragmentID AssemblyRecord) of FragmentID]
         (SETQ \IP.FRAGMENT.LIST (DREMOVE FragmentID \IP.FRAGMENT.LIST))
         (AND FreePacketToo (\RELEASE.ETHERPACKET IP])
```

(\IP.PRINT.FRAGMENT

```
[LAMBDA (FragmentID IPFragment File)                            (* ejs%: " 2-Feb-86 10:39")

        (* * Print information about this fragement to File)

   (printout File T "Received IP Fragment:" T "Source " (\IP.ADDRESS.TO.STRING (fetch (FragmentID SourceAddress)
                                                                                  of FragmentID))
          " Dest "
          (\IP.ADDRESS.TO.STRING (fetch (FragmentID DestinationAddress) of FragmentID))
          T "Protocol ")
   (PRINTCONSTANT (fetch (FragmentID Protocol) of FragmentID)
          IPPROTOCOLTYPES File)
   (printout File " ID " (fetch (FragmentID ID) of FragmentID)
          T "Covering [" (UNFOLD (ffetch (IP IPFRAGMENTOFFSET) of IPFragment)
                                 \IP.FRAGMENTATION.UNIT)
          ".."
          (IPLUS (UNFOLD (ffetch (IP IPFRAGMENTOFFSET) of IPFragment)
                         \IP.FRAGMENTATION.UNIT)
                 (IDIFFERENCE (ffetch (IP IPTOTALLENGTH) of IPFragment)
                        (UNFOLD (ffetch (IP IPHEADERLENGTH) of IPFragment)
                               BYTESPERCELL)))
          "]" T)
   (bind C for I from 0 to [SUB1 (IMIN 40 (IDIFFERENCE (ffetch (IP IPTOTALLENGTH) of IPFragment)
                                                        (UNFOLD (ffetch (IP IPHEADERLENGTH) of IPFragment)
                                                               BYTESPERCELL]
```

```
       do (SETQ C (\GETBASEBYTE (\IPDATABASE IPFragment)
                          I))
          (COND
             ((AND (IGEQ C (CHARCODE SPACE))
                   (ILEQ C 126))
              (BOUT File C))
             (T (printout File "[" C "]"])

)
```

;; Option Processing

```
(DECLARE%: DONTCOPY
```

;; FOLLOWING DEFINITIONS EXPORTED

```
(RPAQQ IPOPTIONTYPES ((IPOPT.END 0)
                      (IPOPT.NOP 1)
                      (IPOPT.SECURITY 2)
                      (IPOPT.LSRR 3)
                      (IPOPT.TIMESTAMP 4)
                      (IPOPT.RECRT 7)
                      (IPOPT.STREAMID 8)
                      (IPOPT.SSSR 9)))

(DECLARE%: EVAL@COMPILE

(RPAQQ IPOPT.END 0)

(RPAQQ IPOPT.NOP 1)

(RPAQQ IPOPT.SECURITY 2)

(RPAQQ IPOPT.LSRR 3)

(RPAQQ IPOPT.TIMESTAMP 4)

(RPAQQ IPOPT.RECRT 7)

(RPAQQ IPOPT.STREAMID 8)

(RPAQQ IPOPT.SSSR 9)

(CONSTANTS (IPOPT.END 0)
       (IPOPT.NOP 1)
       (IPOPT.SECURITY 2)
       (IPOPT.LSRR 3)
       (IPOPT.TIMESTAMP 4)
       (IPOPT.RECRT 7)
       (IPOPT.STREAMID 8)
       (IPOPT.SSSR 9))
)

(DECLARE%: EVAL@COMPILE

(RPAQ IP.OPTION.NUMBER.BYTESPEC (BYTE 5 0))

(CONSTANTS (IP.OPTION.NUMBER.BYTESPEC (BYTE 5 0)))
)
)
```

;; END EXPORTED DEFINITIONS

```
(DEFINEQ
```

(\IP.PROCESS.OPTIONS

```
  [LAMBDA (IP)                                          ; Edited 20-Jan-89 12:24 by bvm
```

;;; Process option fields in IP header. Return T if OK, else handle internally needed actions like redirection or reporting of parameter problems

```
    (bind (OPTIONSSTART _ (LOCF (ffetch (IP IPOPTIONSSTART) of IP)))
          (INDEX _ 0)
          (RESULT _ T)
          REROUTING OPTION until (OR (>= INDEX (- (UNFOLD (fetch (IP IPHEADERLENGTH) of IP)
                                                          BYTESPERCELL)
                                                  \IPOVLEN))
                                     (EQ (SETQ OPTION (LDB (BYTE 5 0)
                                                           (\GETBASEBYTE OPTIONSSTART INDEX)))
                                         IPOPT.END))
       do (if (EQ OPTION IPOPT.NOP)
              then                                      ; This is the only one-byte option we know of other than
                                                        ; IPOPT.END
                   (add INDEX 1)
            else (SELECTC OPTION
                       ((LIST IPOPT.LSRR IPOPT.SSSR)
                            (COND
```

```
                        (REROUTING (SETQ RESULT INDEX))
                        ((NEQ (SETQ RESULT (\IP.OPTION.STRICT.SOURCE.ROUTE IP INDEX))
                              'REROUTE)
                         (SETQ REROUTING T))))
                   (IPOPT.RECRT (SETQ RESULT (\IP.OPTION.RECORD.ROUTE IP INDEX)))
                   (IPOPT.TIMESTAMP
                        (\IP.OPTION.TIMESTAMP IP INDEX))
                   (IPOPT.SECURITY)
                   (IPOPT.STREAMID)
                   (PROGN                                    ; Unknown option code-- we can't continue, since it could be
                                                             ; some unknown 1-byte option
                          (RETURN NIL)))
               (COND
                  ((NUMBERP RESULT)
                   ;; If the result is a number then there was a parameter problem. We could process them here.
                   (RETURN NIL)))
               (add INDEX (\GETBASEBYTE OPTIONSSTART (ADD1 INDEX)))
                                                             ; Increment by the length field
               )
      finally (RETURN RESULT])
```

(\IP.OPTION.RECORD.ROUTE

```
[LAMBDA (IP INDEX)                                           ; Edited 2-Aug-88 14:57 by atm
  (LET* [(OPTIONSSTART (LOCF (ffetch (IP IPOPTIONSSTART) of IP)))
         (LENGTH (\GETBASEBYTE OPTIONSSTART (IPLUS INDEX 1)))
         (PTR (\GETBASEBYTE OPTIONSSTART (IPLUS INDEX 2]
        ;; From RFC 791: If the route data area is already full just forward. If there is room , but not enough for a full address to be inserted, signal
        ;; an ICMP error. Otherwise insert the address into the datagram and update PTR.
        (COND
           ((IGREATERP PTR LENGTH)
            NIL)
           ((ILESSP (IDIFFERENCE LENGTH PTR)
                    3)
            INDEX)
           (T (\PUTBASEFIXP OPTIONSSTART (IPLUS INDEX PTR)
                     (CAR \IP.LOCAL.ADDRESSES))
              (\PUTBASEBYTE OPTIONSSTART (IPLUS INDEX 2)
                     (LDB (BYTE 8 0)
                          (IPLUS PTR 4)))
              T])
```

(\IP.OPTION.STRICT.SOURCE.ROUTE

```
[LAMBDA (IP INDEX)                                           ; Edited 8-Aug-88 12:05 by atm
  (LET* ((OPTIONSSTART (LOCF (ffetch (IP IPOPTIONSSTART) of IP)))
         (LENGTH (\GETBASEBYTE OPTIONSSTART (IPLUS INDEX 1)))
         (PTR (\GETBASEBYTE OPTIONSSTART (IPLUS INDEX 2)))
         (DESTINATIONADDRESSLOC (LOCF (ffetch (IP IPDESTINATIONADDRESS) of IP)))
         (DESTINATIONADDRESS (\GETBASEFIXP DESTINATIONADDRESSLOC 0)))
        ;; From RFC 791: If the address in the destination field has been reached and PTR is not greater than LENGTH, the next address in the
        ;; source route replaces the address in the destination address field, and the recorded route address replaces the source address just
        ;; used, and PTR is increased by four.
        (COND
           ((IGREATERP PTR LENGTH)
            NIL)
           ((ILESSP (IDIFFERENCE LENGTH PTR)
                    3)
            INDEX)
           (T (COND
                 ((MEMBER DESTINATIONADDRESS \IP.LOCAL.ADDRESSES)
                  (\PUTBASEFIXP OPTIONSSTART (IPLUS PTR INDEX 4)
                         DESTINATIONADDRESS)
                  (\PUTBASEFIXP DESTINATIONADDRESSLOC 0 (\GETBASEFIXP OPTIONSSTART (IPLUS PTR INDEX)))
                  (\PUTBASEBYTE OPTIONSSTART (IPLUS INDEX 2)
                         (LDB (BYTE 8 0)
                              (IPLUS PTR 4)))
                  'REROUTE)
                 (T])
```

(\IP.OPTION.TIMESTAMP

```
[LAMBDA (IP INDEX)                                           ; Edited 8-Aug-88 12:08 by atm
  (LET* ((OPTIONSSTART (LOCF (ffetch (IP IPOPTIONSSTART) of IP)))
         (LENGTH (\GETBASEBYTE OPTIONSSTART (IPLUS INDEX 1)))
         (PTR (\GETBASEBYTE OPTIONSSTART (IPLUS INDEX 2)))
         (OFLW/FLG (\GETBASEBYTE OPTIONSSTART (IPLUS INDEX 3)))
         FLAG)
        ;; From RFC 791: If the timestamp area is already full then increment the overflow flag and forward the datagram without inserting the
        ;; timestamp. If there is room but not enough for a full timestamp to be inserted then signal an ICMP error. Otherwise insert the timestamp
        ;; or the timestamp and the internet address depending on the flag; 0 indicates timestamp only, 1 indicates timestamp and address, 3
        ;; indicates that the address is prespecified.
        (COND
```

```
             ((IGREATERP PTR LENGTH)
              (\PUTBASEBYTE OPTIONSSTART (IPLUS INDEX 3)
                     (IPLUS OFLW/FLG (LSH 1 4)))
              T)
             (T (SELECTQ (LOGAND 15 OFLW/FLG)
                     (0 (COND
                           ((ILESSP (IDIFFERENCE LENGTH PTR)
                                    3)
                            INDEX)
                           (T (\PUTBASEFIXP OPTIONSSTART (IPLUS INDEX PTR)
                                     (\CLOCK0 (\CREATECELL \FIXP)))
                              (\PUTBASEBYTE OPTIONSSTART (IPLUS INDEX 2)
                                     (LDB (BYTE 8 0)
                                          (IPLUS PTR 4)))
                              T)))
                     (1 (COND
                           ((IGREATERP 8 (IDIFFERENCE LENGTH (SUB1 PTR)))
                            INDEX)
                           (T (\PUTBASEFIXP OPTIONSSTART (IPLUS INDEX PTR)
                                     (CAR \IP.LOCAL.ADDRESSES))
                              (\PUTBASEFIXP OPTIONSSTART (IPLUS INDEX PTR 4)
                                     (\CLOCK0 (\CREATECELL \FIXP)))
                              (\PUTBASEBYTE OPTIONSSTART (IPLUS INDEX 2)
                                     (LDB (BYTE 8 0)
                                          (IPLUS PTR 8)))
                              T)))
                     (3 [COND
                           ((IGREATERP 8 (IDIFFERENCE LENGTH (SUB1 PTR)))
                            INDEX)
                           (T (COND
                                 ((MEMBER (\GETBASEFIXP OPTIONSSTART (IPLUS INDEX PTR))
                                          \IP.LOCAL.ADDRESSES)
                                  (\PUTBASEFIXP OPTIONSSTART (IPLUS INDEX PTR 4)
                                          (\CLOCK0 (\CREATECELL \FIXP)))
                                  (\PUTBASEBYTE OPTIONSSTART (IPLUS INDEX 2)
                                          (LDB (BYTE 8 0)
                                               (IPLUS PTR 8)))
                                  T)
                                 (T NIL])
                     INDEX])

)

;; Packet Transmission and routing

(RPAQ? \IP.ROUTING.TABLE (CONS))

(RPAQ? \IP.DEFAULT.GATEWAY )

(RPAQ? \IP.LOCAL.NETWORKS )

(RPAQ? \IP.GATEWAY.FORWARDING.FUNCTIONS )

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \IP.ROUTING.TABLE \IP.DEFAULT.GATEWAY \IP.LOCAL.NETWORKS \IP.GATEWAY.FORWARDING.FUNCTIONS)
)

(DEFINEQ
```

(\IP.SETUPIP

```
  [LAMBDA (IP DESTHOST ID SOCKET REQUEUE)                    (* ejs%: "31-Mar-86 15:01")
```

(* * Initialize IP header of packet.)

```
    (OR IP (SETQ IP (\ALLOCATE.ETHERPACKET)))
    (replace (IP IPVERSION) of IP with \IP.PROTOCOLVERSION)
    (freplace (IP IPHEADERLENGTH) of IP with (FOLDHI \IPOVLEN BYTESPERCELL))
    (freplace (IP IPTOTALLENGTH) of IP with \IPOVLEN)
    [freplace (IP IPID) of IP with (OR (SMALLP ID)
                                       (LOGAND (DAYTIME)
                                               (MASK.1'S 0 16]
    (freplace (IP IPMOREFRAGMENTS) of IP with NIL)
    (freplace (IP IPFRAGMENTOFFSET) of IP with 0)
    (freplace (IP IPTIMETOLIVE) of IP with \IP.DEFAULT.TIME.TO.LIVE)
    (freplace (IP IPPROTOCOL) of IP with (fetch (IPSOCKET PROTOCOL) of SOCKET))
    (freplace (IP IPSOURCEADDRESS) of IP with (CAR \IP.LOCAL.ADDRESSES))
    (freplace (IP IPDESTINATIONADDRESS) of IP with DESTHOST)
    (freplace EPREQUEUE of IP with REQUEUE)
    IP])
```

(\IP.TRANSMIT

```
  [LAMBDA (IP ROUTINGREADONLY)                               (* ejs%: "27-Jan-86 15:59")
```

(* * Sends an IP packet, after first computing the IP header checksum)

```
    (PROG (NDB)
          (SETQ IP (\DTEST IP 'ETHERPACKET))
          (until \IP.READY do (AWAIT.EVENT \IP.READY.EVENT))
          (\RCLK (LOCF (ffetch EPTIMESTAMP of IP)))
          (replace EPTYPE of IP with \EPT.IP)
          (RETURN (COND
                     ((ffetch EPTRANSMITTING of IP)
                      (AND IPTRACEFLG (printout IPTRACEFILE "[Put fails--packet already being transmitted]"))
                      'AlreadyQueued)
                     ((NOT (SETQ NDB (\IP.ROUTE.PACKET IP ROUTINGREADONLY)))
                      (AND IPTRACEFLG (PRINTPACKET IP 'PUT IPTRACEFILE "[Put fails--no routing]"))
                      (\REQUEUE.ETHERPACKET IP)
                      'NoRouting)
                     (T (\IP.SET.CHECKSUM IP (ffetch (IP IPBASE) of IP)
                                (LLSH (ffetch (IP IPHEADERLENGTH) of IP)
                                      2)
                                (LOCF (ffetch (IP IPHEADERCHECKSUM) of IP)))
                        [COND
                           (IPTRACEFLG (COND
                                          ((EQ IPTRACEFLG T)
                                           (PRINTPACKET IP 'PUT IPTRACEFILE))
                                          (T (PRIN1 "!" IPTRACEFILE]
                        (TRANSMIT.ETHERPACKET NDB IP)
                        NIL])
```

(\IP.ROUTE.PACKET

```
  [LAMBDA (IP READONLY)                                                    ; Edited 19-Jan-89 18:00 by bvm
```

;; Encapsulates XIP, choosing the right network and immediate destination host. Returns an NDB for the transmission. Unless READONLY is ;; true, defaults source and destination nets if needed

```
    (DECLARE (GLOBALVARS \10MBLOCALNDB \3MBLOCALNDB \IP.LOCAL.NETWORKS \IP.DEFAULT.GATEWAY))
    (PROG ((DESTADDRESS (fetch (IP IPDESTINATIONADDRESS) of IP))
           DESTNET SUBNETMASK SOURCEHOSTADDRESS SUBNETINUSE PDH ROUTE NDB EPTYPE BROADCASTP)
          (SETQ DESTNET (\IPNETADDRESS DESTADDRESS))
```

;; Try to resolve a destination network of 0.0 If we have two attached networks, fail.

```
          [COND
             ((AND (EQ 0 DESTADDRESS)
                   \10MBLOCALNDB \3MBLOCALNDB)
              (RETURN))
             ((EQ 0 DESTADDRESS)
              '[SETQ DESTADDRESS (\IP.MAKE.BROADCAST.ADDRESS (fetch NDBIPHOST# of (OR \10MBLOCALNDB \3MBLOCALNDB
                                                                                     ]
              (SETQ DESTADDRESS -1)
              (SETQ BROADCASTP T)
              '(SETQ DESTNET (\IPNETADDRESS DESTADDRESS))
              (SETQ DESTNET (CAAR \IP.LOCAL.NETWORKS]
```

;; First see if the destination network is one of our local networks

```
          [COND
             [(AND (SETQ NDB (CDR (SASSOC DESTNET \IP.LOCAL.NETWORKS)))
                   (SETQ SUBNETMASK (CDR (SASSOC (SETQ SOURCEHOSTADDRESS (fetch (NDB NDBIPHOST#) of NDB))
                                                 \IP.SUBNET.MASKS)))
                   (OR (AND (\IP.BROADCAST.ADDRESS DESTADDRESS)
                            (SETQ BROADCASTP T))
                       (EQP (LOGAND SOURCEHOSTADDRESS SUBNETMASK)
                            (LOGAND DESTADDRESS SUBNETMASK))
                       (PROGN (SETQ SUBNETINUSE T)
                              NIL)))
```

;; A local net. Try to find the Ethernet address of the host

```
              (COND
                 [(SETQ PDH (SELECTQ (fetch (NDB NETTYPE) of NDB)
                                 (10 (SETQ EPTYPE \EPT.IP)
                                     (COND
                                        (BROADCASTP BROADCASTNSHOSTNUMBER)
                                        (T (\AR.TRANSLATE.TO.10MB DESTADDRESS))))
                                 (3 (SETQ EPTYPE \EET.IP)
                                    (\AR.TRANSLATE.TO.3MB DESTADDRESS))
                                 (SHOULDNT]
                  (T                                                       ; Nope
                     (RETURN]
              (T ;; The host is not on a local net.  See if we have a route to that host, or use the default route if necessary
                 (COND
                    [(SETQ ROUTE (OR [COND
                                       (SUBNETINUSE (CDR (SASSOC (LOGAND DESTADDRESS SUBNETMASK)
                                                                 \IP.ROUTING.TABLE)))
                                       (T (CDR (SASSOC DESTNET \IP.ROUTING.TABLE]
                                     \IP.DEFAULT.GATEWAY))
```

;; We've got the IP address of the gateway

```
                     (COND
                        [(SETQ NDB (CDR (SASSOC (\IPNETADDRESS ROUTE)
                                                \IP.LOCAL.NETWORKS)))
```

;; We know what network it's on

```
                        (COND
                           [(SETQ PDH (SELECTQ (fetch (NDB NETTYPE) of NDB)
                                              (10 (SETQ EPTYPE \EPT.IP)
                                                  (\AR.TRANSLATE.TO.10MB ROUTE))
                                              (3 (SETQ EPTYPE \EET.IP)
                                                 (\AR.TRANSLATE.TO.3MB ROUTE))
                                              (SHOULDNT]
                              (T (RETURN]
                           (T (ERROR "IP routing table contains non-local gateway address for network" DESTNET]
                     (T (RETURN]
               (freplace EPNETWORK of IP with NDB)
               (ENCAPSULATE.ETHERPACKET NDB IP PDH (ffetch (IP IPTOTALLENGTH) of IP)
                      EPTYPE)
               (replace EPTYPE of IP with EPTYPE)
               [COND
                  ((NOT READONLY)
                   (COND
                      ((EQ 0 (fetch (IP IPDESTINATIONADDRESS) of IP))
                       (freplace (IP IPDESTINATIONADDRESS) of IP with DESTADDRESS)))
                   (freplace (IP IPSOURCEADDRESS) of IP with (fetch NDBIPHOST# of NDB]
               (RETURN NDB])

)

(DEFINEQ
```

(IP.GET

```
  [LAMBDA (IPSOCKET WAIT)                                    (* ejs%: "31-Mar-86 14:30")
```

(* * Returns the next IP packet on the queue, or NIL if none exist and WAIT is NIL.
If WAIT is T, this function waits forever. If WAIT is an integer, it is interpreted as the number of milliseconds to wait before returning NIL or a packet which arrives during that time. This function therefore is like GETXIP and GETPUP)

```
    (PROG ((QUEUE (fetch (IPSOCKET IPSQUEUE) of IPSOCKET))
           IP TIMER)
      LP   (UNINTERRUPTABLY
               (COND
                  ((SETQ IP (\DEQUEUE QUEUE))
                   (add (fetch (IPSOCKET IPSQUEUELENGTH) of IPSOCKET)
                        -1))))
           [COND
              ((NULL IP)
               (COND
                  (WAIT (COND
                           ((EQ WAIT T))
                           [TIMER (COND
                                     ((TIMEREXPIRED? TIMER)
                                      (RETURN]
                           (T (OR (FIXP WAIT)
                                  (LISPERROR "NON-NUMERIC ARG" WAIT))
                              (SETQ TIMER (SETUPTIMER WAIT))
                              T))
                        (AWAIT.EVENT (fetch (IPSOCKET IPSEVENT) of IPSOCKET)
                               TIMER T)
                        (GO LP))
                  (T (BLOCK]
           (RETURN IP])
```

(IP.SEND

```
  [LAMBDA (IP)                                               (* ejs%: "31-Mar-86 15:07")
    (\IP.TRANSMIT IP])
```

(IP.PACKET.WATCHER

```
  [LAMBDA (IPSOCKET PACKET.FUNCTION)                         (* ejs%: "31-Mar-86 15:50")
```

(* * Infinite loop which waits for packet on IPSOCKET, and calls PACKET.FUNCTION whenever one arrives)

```
    (COND
       ((NOT (type? IPSOCKET IPSOCKET))
        (ERROR "ARG NOT IPSOCKET" IPSOCKET))
       ((NOT (FNTYP PACKET.FUNCTION))
        (ERROR "UNDEFINED FUNCTION" PACKET.FUNCTION))
       (T (while T do (APPLY* PACKET.FUNCTION (IP.GET IPSOCKET T)
                              IPSOCKET])

)

(DECLARE%: EVAL@COMPILE

(PUTPROPS IP.SEND MACRO [LAMBDA (IP)                         (* ejs%: "31-Mar-86 15:07")
                          (\IP.TRANSMIT IP])
)
```

;; Client functions for building packets

```
(DEFINEQ
```

(\IP.APPEND.BYTE

```
  [LAMBDA (IP BYTE INHEADER)                                    (* ejs%: "28-Dec-84 08:23")
```

(* * Append a byte to an IP packet. If INHEADER is not NIL, we adjust the header length field as well.)

```
    (PROG (NEWLENGTH)
          (\PUTBASEBYTE (fetch (IP IPBASE) of IP)
                 (fetch (IP IPTOTALLENGTH) of IP)
                 BYTE)
          (SETQ NEWLENGTH (add (ffetch (IP IPTOTALLENGTH) of IP)
                               1))
          [COND
             (INHEADER (freplace (IP IPHEADERLENGTH) of IP with (FOLDHI NEWLENGTH 4]
          (RETURN NEWLENGTH])
```

(\IP.APPEND.CELL

```
  [LAMBDA (IP CELL INHEADER)                                    (* ejs%: "28-Dec-84 08:33")
```

(* * Append a cell to an IP packet. If INHEADER is not NIL, we adjust the header length field as well.)

```
    (PROG (NEWLENGTH (OFFSET (fetch (IP IPTOTALLENGTH) of IP)))
          [COND
             ((EVENP OFFSET)
              (\PUTBASEFIXP (fetch (IP IPBASE) of IP)
                     (FOLDLO OFFSET 2)
                     CELL))
             (T (\PUTBASEBYTE (fetch (IP IPBASE) of IP)
                       OFFSET
                       (LDB (BYTE 8 24)
                            CELL))
                (\PUTBASEBYTE (fetch (IP IPBASE) of IP)
                       (\ADDBASE OFFSET 1)
                       (LDB (BYTE 8 16)
                            CELL))
                (\PUTBASEBYTE (fetch (IP IPBASE) of IP)
                       (\ADDBASE OFFSET 2)
                       (LDB (BYTE 8 8)
                            CELL))
                (\PUTBASEBYTE (fetch (IP IPBASE) of IP)
                       (\ADDBASE OFFSET 3)
                       (LDB (BYTE 8 0)
                            CELL]
          (SETQ NEWLENGTH (add (ffetch (IP IPTOTALLENGTH) of IP)
                               4))
          (COND
             (INHEADER (add (ffetch (IP IPHEADERLENGTH) of IP)
                            1)))
          (RETURN NEWLENGTH])
```

(\IP.APPEND.STRING

```
  [LAMBDA (IP STRING)                                           (* ejs%: " 9-Feb-85 19:44")
    (PROG ((LENGTH (fetch (STRINGP LENGTH) of STRING)))
          (\MOVEBYTES (fetch (STRINGP BASE) of STRING)
                 (fetch (STRINGP OFFST) of STRING)
                 (fetch (IP IPBASE) of IP)
                 (fetch (IP IPTOTALLENGTH) of IP)
                 LENGTH)
          (RETURN (add (ffetch (IP IPTOTALLENGTH) of IP)
                       LENGTH])
```

(\IP.APPEND.WORD

```
  [LAMBDA (IP WORD INHEADER)                                    (* ejs%: "28-Dec-84 08:28")
```

(* * Append a word to an IP packet. If INHEADER is not NIL, we adjust the header length field as well.)

```
    (PROG (NEWLENGTH (OFFSET (fetch (IP IPTOTALLENGTH) of IP)))
          [COND
             ((EVENP OFFSET)
              (\PUTBASE (fetch (IP IPBASE) of IP)
                     (FOLDLO OFFSET 2)
                     WORD))
             (T (\PUTBASEBYTE (fetch (IP IPBASE) of IP)
                       OFFSET
                       (LDB (BYTE 8 8)
                            WORD))
                (\PUTBASEBYTE (fetch (IP IPBASE) of IP)
                       (\ADDBASE OFFSET 1)
                       (LDB (BYTE 8 0)
                            WORD]
          (SETQ NEWLENGTH (add (ffetch (IP IPTOTALLENGTH) of IP)
                               2))
```

```
        [COND
           (INHEADER (freplace (IP IPHEADERLENGTH) of IP with (FOLDHI NEWLENGTH 4]
        (RETURN NEWLENGTH])
```

(\IP.GET.BYTE

```
[LAMBDA (IP BYTE INHEADER)                                    (* ejs%: "30-Mar-86 14:49")
```

(* * Retrieve a byte from an IP packet. If INHEADER is T, BYTE is an offset from the start of the packet, else it's an offset from the start of the IP data section)

```
  (\GETBASEBYTE (COND
                   (INHEADER (fetch (IP IPBASE) of IP))
                   (T (\IPDATABASE IP)))
        BYTE])
```

(\IP.GET.CELL

```
[LAMBDA (IP CELL INHEADER)                                    (* ejs%: "30-Mar-86 15:07")
```

(* * Retrieve a cell from an IP packet. If INHEADER is not NIL, the cell is written to the header portion of the IP packet, else it's written to the data portion. CELL is the offset, in 16-bit units)

```
  (\GETBASEFIXP (COND
                   (INHEADER (fetch (IP IPBASE) of IP))
                   (T (\IPDATABASE IP)))
        CELL])
```

(\IP.GET.STRING

```
[LAMBDA (IP BYTEOFFSET NCHARS INHEADER)                       (* ejs%: "30-Mar-86 15:13")
```

(* * Retrieve a string from an IP packet. If INHEADER is T, BYTEOFFSET is an offset from the start of the packet, else it's an offset from the start of the IP data section)

```
  (\GETBASESTRING (COND
                     (INHEADER (fetch (IP IPBASE) of IP))
                     (T (\IPDATABASE IP)))
        BYTEOFFSET NCHARS])
```

(\IP.GET.WORD

```
[LAMBDA (IP WORD INHEADER)                                    (* ejs%: "30-Mar-86 14:51")
```

(* * Retrieve a word from an IP packet. If INHEADER is T, WORD is an offset from the start of the packet, else it's an offset from the start of the IP data section)

```
  (\GETBASE (COND
               (INHEADER (fetch (IP IPBASE) of IP))
               (T (\IPDATABASE IP)))
        WORD])
```

(\IP.PUT.BYTE

```
[LAMBDA (IP BYTE VALUE INHEADER)                              (* ejs%: "30-Mar-86 14:52")
```

(* * Store a byte in an IP packet. If INHEADER is T, BYTE is an offset from the start of the packet, else it's an offset from the start of the IP data section)

```
  (\PUTBASEBYTE (COND
                   (INHEADER (fetch (IP IPBASE) of IP))
                   (T (\IPDATABASE IP)))
        BYTE VALUE])
```

(\IP.PUT.CELL

```
[LAMBDA (IP CELL VALUE INHEADER)                              (* ejs%: "30-Mar-86 15:06")
```

(* * Store a cell in an IP packet. If INHEADER is not NIL, the cell is written to the header portion of the IP packet, else it's written to the data portion. CELL is the offset, in 16-bit units)

```
  (\PUTBASEFIXP (COND
                   (INHEADER (fetch (IP IPBASE) of IP))
                   (T (\IPDATABASE IP)))
        CELL VALUE])
```

(\IP.PUT.STRING

```
[LAMBDA (IP BYTEOFFSET STRING INHEADER)                       (* ejs%: "30-Mar-86 15:13")
```

(* * Store a string ib an IP packet. If INHEADER is T, BYTEOFFSET is an offset from the start of the packet, else it's an offset from the start of the IP data section)

```
  (\PUTBASESTRING (COND
                     (INHEADER (fetch (IP IPBASE) of IP))
                     (T (\IPDATABASE IP)))
        BYTEOFFSET STRING])
```

(\IP.PUT.WORD

```
  [LAMBDA (IP WORD VALUE INHEADER)                    (* ejs%: "30-Mar-86 14:50")
```

(* * Store a word in an IP packet. If INHEADER is T, WORD is an offset from the start of the packet, else it's an offset from the start of the IP data section)

```
    (\PUTBASE (COND
                 (INHEADER (fetch (IP IPBASE) of IP))
                 (T (\IPDATABASE IP)))
              WORD VALUE])

)

(DECLARE%: EVAL@COMPILE

(PUTPROPS \IP.GET.BYTE DMACRO [LAMBDA (IP BYTE INHEADER)          (* ejs%: "30-Mar-86 14:49")
```

(* * Retrieve a byte from an IP packet. If INHEADER is T, BYTE is an offset from the start of the packet, else it's an offset from the start of the IP data section)

```
                                (\GETBASEBYTE (COND
                                                 (INHEADER (fetch (IP IPBASE) of IP))
                                                 (T (\IPDATABASE IP)))
                                      BYTE])

(PUTPROPS \IP.GET.CELL DMACRO [LAMBDA (IP CELL INHEADER)          (* ejs%: "30-Mar-86 15:07")
```

(* * Retrieve a cell from an IP packet. If INHEADER is not NIL, the cell is written to the header portion of the IP packet, else it's written to the data portion. CELL is the offset, in 16-bit units)

```
                                (\GETBASEFIXP (COND
                                                 (INHEADER (fetch (IP IPBASE) of IP))
                                                 (T (\IPDATABASE IP)))
                                      CELL])

(PUTPROPS \IP.GET.STRING DMACRO [LAMBDA (IP BYTEOFFSET NCHARS INHEADER)
                                                                  (* ejs%: "30-Mar-86 15:13")
```

(* * Retrieve a string from an IP packet. If INHEADER is T, BYTEOFFSET is an offset from the start of the packet, else it's an offset from the start of the IP data section)

```
                                  (\GETBASESTRING (COND
                                                     (INHEADER (fetch (IP IPBASE) of IP))
                                                     (T (\IPDATABASE IP)))
                                        BYTEOFFSET NCHARS])

(PUTPROPS \IP.GET.WORD DMACRO [LAMBDA (IP WORD INHEADER)          (* ejs%: "30-Mar-86 14:51")
```

(* * Retrieve a word from an IP packet. If INHEADER is T, WORD is an offset from the start of the packet, else it's an offset from the start of the IP data section)

```
                                 (\GETBASE (COND
                                              (INHEADER (fetch (IP IPBASE) of IP))
                                              (T (\IPDATABASE IP)))
                                       WORD])

(PUTPROPS \IP.PUT.BYTE DMACRO [LAMBDA (IP BYTE VALUE INHEADER)    (* ejs%: "30-Mar-86 14:52")
```

(* * Store a byte in an IP packet. If INHEADER is T, BYTE is an offset from the start of the packet, else it's an offset from the start of the IP data section)

```
                                (\PUTBASEBYTE (COND
                                                 (INHEADER (fetch (IP IPBASE) of IP))
                                                 (T (\IPDATABASE IP)))
                                      BYTE VALUE])

(PUTPROPS \IP.PUT.CELL DMACRO [LAMBDA (IP CELL VALUE INHEADER)    (* ejs%: "30-Mar-86 15:06")
```

(* * Store a cell in an IP packet. If INHEADER is not NIL, the cell is written to the header portion of the IP packet, else it's written to the data portion. CELL is the offset, in 16-bit units)

```
                                (\PUTBASEFIXP (COND
                                                 (INHEADER (fetch (IP IPBASE) of IP))
                                                 (T (\IPDATABASE IP)))
                                      CELL VALUE])

(PUTPROPS \IP.PUT.STRING DMACRO [LAMBDA (IP BYTEOFFSET STRING INHEADER)
                                                                  (* ejs%: "30-Mar-86 15:13")
```

(* * Store a string ib an IP packet. If INHEADER is T, BYTEOFFSET is an offset from the start of the packet, else it's an offset from the start of the IP data section)

```
                                  (\PUTBASESTRING (COND
                                                     (INHEADER (fetch (IP IPBASE) of IP))
                                                     (T (\IPDATABASE IP)))
```

```
                                        BYTEOFFSET STRING])

(PUTPROPS \IP.PUT.WORD DMACRO [LAMBDA (IP WORD VALUE INHEADER)  (* ejs%: "30-Mar-86 14:50")

          (* * Store a word in an IP packet. If INHEADER is T, WORD is an offset from the start of the packet, else it's an offset from
          the start of the IP data section)

                                    (\PUTBASE (COND
                                                (INHEADER (fetch (IP IPBASE) of IP))
                                                (T (\IPDATABASE IP)))
                                              WORD VALUE])
)

(MOVD? 'NILL 'IP.DEFAULT.CONFIGURATION)

(DECLARE%: EVAL@COMPILE DONTCOPY

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \IP.LOCAL.NETWORKS \IP.DEFAULT.GATEWAY \IP.INIT.FILE \IP.SUBNET.MASKS \PROCESS.AFTEREXIT.EVENT
       \PROC.READY \AR.IP.TO.10MB.ALIST)
)
)

(PUTPROPS TCPLLIP COPYRIGHT ("Xerox Corporation" 1985 1986 1987 1988 1989 1990))
```

FUNCTION INDEX

CONSTANT INDEX

VARIABLE INDEX

{MEDLEY}<obsolete>tcp>TCPLLIP.;1

MACRO INDEX

RECORD INDEX

PROPERTY INDEX
