TCPIP

```
PSIONICS FILE - TCPIP
=====================
Interface to TCP/IP
Last modified 1998-02-23
========================
The TCP/IP stack for the Psion 3 series is shipped as a separate productcalled PsiMail; it comes
with various utilities. This file describes theprogramming interface to this stack. It assumes
knowledge of the UnixBerkeley Sockets interface, and terms not defined here have the same meaning
as there.
General conventions
-------------------
Most network functions are carried out by sending messages to objects. Thereare two objects -
the IP object and the DNS object.
Messages are sent to the objects with SEND. This takes from 2 to 5arguments: the object handle,
the message type number, and 0 to 2 additionalitems. These items are called V1, V2, and V3 in
this file, and are described
in OPL terms as an object. Thus, for example, given:
    Message 99 - Activate or deactivate network
    V1: 10 byte buffer described below
    V2: a word holding the value 8
    Returns: zero for success, non-zero for failure
typical code to make the call would be:
    LOCAL bufx%,bufy&,bufz&,x%
    x%=8
    IF SEND(ipobj%, 99, bufx%, x%)<>0
        ALERT("Can't activate network")
    ENDIF
and the variables bufx%, bufy&, and bufz& form the buffer.
Where a constant is given, exactly that value must be used. Where bits arenot mentioned in a
value, they must be set to zero and ignored when read.
Network addresses are 16 bytes long, and have the following format:
  Offset 0 (word): must be 2
```

```
  Offset 2 (word): port number, in network order
  Offset 4 (long): IP address, in network order
  Offset 8 to  15: unusedNote that the address and port numbers are in network order (most
significant byte in the lowest offset).
Accessing the interface
-----------------------
The interface is provided by a library called SOCKDYL.DYL. If there is acopy in the ROM, it will
be in the file ROM::SOCKDYL.DYL and can be invoked
by:
    FINDLIB(tcpipch%, "SOCKDYL.DYL")Otherwise it should be in a directory called \NET on a local
drive. Programs
are recommended to look on all drives and determine the copy with thehighest version number
(stored in a word at offset 24 of the file). Then
invoke the library by:
    LOADLIB(tcpipch%, "file", 1)where the second parameter is the name of the file holding the
copy used.
In either case, the command will return 0 for success; tcpipch% is thecategory handle for the
library.
When the program has finished with the network, it should unload the library
using UNLOADLIB.
Actual access to the network is carried out through two objects - the IPobject and the DNS
object. The IP object is created by:
    ipobj%=NEWOBJH(tcpipch%,0)
    SEND(ipobj%,25,#0,#0)      : REM see below for explanation
The DNS object is created by:
    dnsobj%=NEWOBJH(tcpipch%,2)
Before unloading the library, these objects should be destroyed by sending
them message 0:
    SEND(ipobj%,0)
    SEND(dnsobj%,0)
The IP object
-------------
The IP object provides the following messages.
Messages 2 to 24 have two additional arguments with a standard meaning: thefirst is a control
block of some size, whose format is given instead of theargument descriptions, and the second is
a status word. These calls can be
made synchronously or asynchronously.
```

To make a synchronous call, set the status word to 0 beforehand. If thecall fails, -1 is returned and the error code is in the status word.Otherwise the call succeeded and the returned value is as described.

To make an asynchronous call, set the status word to 1 beforehand. Ifthe call returns -1, it failed and the error code is in the status word.Otherwise, the status word will have been set to -46; when the operationfinishes, the status word will be changed and the semaphore signalled. The new status word should now be passed to message 1. If the operationfailed, -1 is returned and the status word holds the error code. Otherwisethe value returned is the result of the operation.

Message 1 - interpret asynchronous result

V1: status word

See above for how to use this message.

Message 2 - create socket

The control block is 6 bytes:

  Offset 0 (word): must be 2 Offset 2 (word): socket type: 1 = TCP stream, 2 = UDP datagram, 3 = raw

  Offset 4 (word): must be 0

Returns: socket handle for a new socket

Creates a new socket.

Message 3 - bind socket

The control block is 6 bytes:

  Offset 0 (word): socket handle

  Offset 2 (word): address of a 16 byte network address block

  Offset 4 (word): must be 16

Binds a socket. Within the address, IP address 0 may be used to bind toany local interface, and port number 0 may be used to request that a port
is assigned on connection.

Message 4 - listen on socket

The control block is 4 bytes:

  Offset 0 (word): socket handle

  Offset 2 (word): length of the backlog queue

Starts a bound TCP socket listening for new connections.

Message 5 - accept a connection

The control block is 6 bytes:

  Offset 0 (word): socket handle

  Offset 2 (word): address of a 16 byte network address block Offset 4 (word): must be 16 (may be changed by the call)

```
Returns: handle of new socket
Accepts a connection from a remote client, placing it on a new socket. Theoriginal socket
remains open for connections; the new socket can be usedfor data transfer. The address block
will be filled in with the address of
the remote client.
Message 6 - make a connection
The control block is 6 bytes:
  Offset 0 (word): socket handle
  Offset 2 (word): address of a 16 byte network address block
  Offset 4 (word): must be 16
Connects a socket to a remote service. The address block should specifythe address of the remote
service.
Message 7 - get peer name
Message 8 - get socket name
The control block is 6 bytes:
  Offset 0 (word): socket handle
  Offset 2 (word): address of a 16 byte network address block  Offset 4 (word): must be 16 (may
be changed by the call)
Obtains the address of the remote or local end (respectively) of a socket.The address block will
be filled with the relevant address.
Message  9 - get socket option
Message 10 - set socket option
The control block is 10 bytes:
  Offset 0 (word): socket handle
  Offset 2 (word): must be $FFFF
  Offset 4 (word): option code  Offset 6 (word): address of the area the value is taken from or
put in  Offset 8 (word): size in bytes of the area (for message 9, this will
be
                    changed to the actual size of the value)
Gets or sets an option on a socket. In the following table, "bool" means aword that is either
zero (for false or disabled) or non-zero (for true or
enabled).
Available options are:
  option   size   meaning
  $0004    bool   local address can be reused
  $0008    bool   keep connections alive
  $0010    bool   don't route outgoing messages
```

```
  $0020    bool   broadcasting permitted
  $0080      4    linger control block  $0100    bool   out of band data will be received in-
band
  $1001      2    output buffer size
  $1002      2    input buffer size
  $1003      2    transmit low water mark
  $1004      2    receive low water mark
  $1005      2    transmit timeout
  $1006      2    receive timeout  $1007      2    pending error (will clear the error) [can't
be set]
  $1008      2    socket type [can't be set]
The linger control block has the format:
  Offset 0 (word): 0 to disable lingering, 1 to enable  Offset 2 (word): linger time in seconds
Lingering affects the behaviour of a TCP socket where data is waiting tobe sent when the socket
is closed. If it is enabled, the close will waitup to the linger time for the data to be sent.
If it is disabled, the close
will return as soon as possible.
Message 11 - receive data
Message 12 - receive data
The control block is 8 bytes (message 11) or 12 bytes (message 12):  Offset  0 (word): socket
handle
  Offset  2 (word): address of buffer to store data in  Offset  4 (word): maximum number of
bytes to receive
  Offset  6 (word): flags:
    Bit 0: set to process out of band data    Bit 1: if set, the data remains in the input
stream and can be received
           a second time
  Offset  8 (word): address of a 16 byte network address block  Offset 10 (word): must be 16
(may be changed by the call)
Returns: number of bytes received
Receives data from the socket and places them in memory. Either message canbe used with any
socket, but message 12 stores the sender's address in theblock, and so is more useful with
datagram sockets that are not connected to
a single peer.
Message 13 - receive data
@@@@ Documentation not available
Message 14 - select
```

@@@@ Documentation not available

Message 15 - send data

Message 16 - send data to specified addressThe control block is 8 bytes (message 15) or 12 bytes (message 16):

  Offset 0 (word): socket handle

  Offset 2 (word): address of first byte to send

  Offset 4 (word): number of bytes to send

  Offset 6 (word): flags:

    Bit 0: set for out of band data

    Bit 2: set to bypass routing

  Offset 8 (word): address of a 16 byte network address block Offset 10 (word): must be 16

Returns: number of bytes actually sent

Sends data to the remote peer of a connected socket (message 15) or to thespecified address with an unconnected socket (message 16).

Message 17 - send data

@@@@ Documentation not available

Message 18 - partially close a socket

The control block is 4 bytes:

  Offset 0 (word): socket handle Offset 2 (word): 0 = end reception, 1 = end transmission, 2 = end both

Performs a close on one or both directions of a connected TCP socket.

Message 19 - close a socket

The control block is 2 bytes:

  Offset 0 (word): socket handle

Close a socket; the handle may no longer be used.

Message 20 - asynchronous action

@@@@ Documentation not available

Message 21 - configuration

@@@@ Documentation not available

Message 22 - control network

The control block is 6 bytes:

  Offset 0 (long): action:

    1 = connect to network,

    2 = disconnect from network,

    3 = return network status,

    4 = get length of time connected, in seconds

    5 = get local IP address, in network order Offset 4 (word): address of a long to store any

fetched value in
Returns: network status (action 3 only, see below)
Performs some network control action. Note that it is not necessary to usethis function to set up a connection; the first attempt to use the network
will do so automatically.
For action 2 only, the returned value gives the current status:0 = disconnected, 1 = connecting, 2 = connected.
For actions 3 and 4, the result is stored in the long pointed to byoffset 4; the result is only valid if the call was made while connected to
the network.
Message 23 - cancel a pending operation on a socket
The control block is 2 bytes:
  Offset 0 (word): socket handle
Cancels any pending operation on a socket. This call should only be made insynchronous mode, with the status word initialized to zero.
Message 24 - assign socket
@@@@ Documentation not available
@@@@ socket, pid
Message 25 - register network client
No additional arguments
Returns: 0 for success, a negative error code for failure
This call registers the current process as a client of the network system.It must be made when the IP object is created (see above), and can becalled additional times, in which case the calls are counted by the network
server process. Programs do not normally need to make additional calls.
Message 26 - deregister network client
No additional arguments
Returns: 0 for success, a negative error code for failure
This call cancels a registration of the current process as a client of thenetwork system, decrementing the count held by the network server. One suchcancellation is made when the IP object is destroyed, and all registrationsare cancelled when the current process terminates. Programs do not normally
need to call this directly.
The DNS object

The DNS object provides the following messages.
Message 1 - convert host to network (long)

```
Message 2 - convert network to host (long)
Message 3 - convert host to network (word)
Message 4 - convert network to host (word)
V1: 2 byte or 4 byte buffer holding the value
Each of these converts a value between host and network ordering; the
conversion is done in situ.
Message 5 - convert dotted quad to IP address
V1: long to hold resulting address
V2: cstr to be converted
Returns: zero for success or a non-zero error code
Converts a dotted quad name (such as "158.152.1.222") to an IP addresssuch as $9E9801DE, but in
network order. The dotted quad may have two orthree components, and each component may be a C
decimal, octal, or hex
number (so another form for the above is "0x9E.230.478").
Message 6 - test for valid IP address
@@@@ Documentation not available
Message 7 - get host by name
V1: 10 byte buffer which is filled in with the host informationV2: cstr of host name to be
looked up
V3: status word
Returns: zero for success or a non-zero error code
This call is always asynchronous. It looks up a host in the DNS, and, iffound, the buffer is
filled with the following information:
  Offset 0 (word): address of the canonical name (a cstr)  Offset 2 (word): address of a zero-
terminated list of words, each holding                   the address of a cstr giving an alias,
or zero if there
                   are no aliases
  Offset 4 (long): unused  Offset 8 (word): address of a zero-terminated list of words, each
holding
                   the address of a block holding a network addressThe data pointed to is stored
in an area which may be overwritten by any
other call to the interface module.
Message 8 - get host by addressV1: 10 byte buffer which is filled in with the host information
V2: 6 byte control block
V3: status word
Returns: zero for success or a non-zero error code
This call is always asynchronous. It looks up a host in the DNS, and, iffound, the buffer is
```

filled with the same information as for message 7. The
control block has the following format: Offset 0 (word): address of a long holding the IP address in network order
  Offset 2 (word): must be 4
  Offset 4 (word): must be 2
Message 9 - cancel a pending get host operationV1: 10 byte buffer provided to the get host operation
This call cancels any outstanding get host operation (messages 7 and 8).The V1 value must be that provided to the original operation.
Message 10 - extract local component of IP addressMessage 11 - construct IP address from local and network components
Message 12 - extract network component of IP address
These calls are intended to split and recombine IP addresses treated asnetwork and local components, using the old class A, B, and C networkconcepts. They are not documented here, partly because subnets and CIDRhave made them obsolete, and partly because the interfaces are flawed.
Message 13 - convert IP address to dotted quadV1: long holding the IP address in network order
V2: 16 byte buffer filled in with the dotted quad as a cstr
This call converts an IP address such as $9E9801DE (but in network order)to a dotted quad name such as "158.152.1.222".
Error codes

The IP object generates the following error codes:
   -33 TCP/IP is not installed on the system   -32 Incompatible versions of TCP/IP are being used simultaneously
   -16 sockdyl.dyl is corrupt
   -10 Insufficient memory to initialize
     9 Bad socket handle
    11 Out of resources
    14 Bad argument to operation (e.g. not a valid ADDR)   22 Invalid argument (e.g. attempting to re-bind a bound socket)
    35 Operation would block and socket is non-blocking   36 Non-blocking operation in progress but not completed
    37 Previous non-blocking operation still in progress   38 Socket operation on non-socket
    39 Destination address required and socket not connected   40 Message too long and socket type does not allow it to be split

```
    41  Wrong protocol type for this socket type    42  Unknown option for this protocol
level
    43  Protocol not supported in this address domain
    44  Socket type not supported
    45  Operation not supported on this socket type
    46  Protocol family not supported
    47  Address family not supported by protocol family
    48  Address already in use
    49  Can't assign requested address
    50  Network is down
    51  Network is unreachable
    52  Network dropped connection on reset
    53  Software caused connection abort
    54  Connection reset by peer
    55  No buffer space available in the socket server
    56  Socket is already connected
    57  Socket is not connected
    58  Can't send after socket shutdown
@@  59  Too many references: can't splice
    60  Connection timed out
    61  Connection refused
    64  Host is down
    65  No route to host
    66  Network has not been initialized
  1001  Syntax error in connection script
  1002  Got abort string while connecting
  1004  Error in configuration
  1005  Serial port is in use
  1006  Error with serial port
  1007  Error reading connection script file
  1008  Timer timed out while connecting
  1009  Not enough memory to connect
  1010  Couldn't find connection script dyl
  1011  Couldn't load connection script dyl
  1012  ISP modem busy
  1013  No carrier
  1014  No answer from ISP modem
```

```
  1015  Couldn't read dial settings
The DNS object generates the following error codes:
     1  insufficient memory in DNS process
     2  failure creating socket in DNS process
     3  failure during connect to DNS server
     4  failure during send to DNS server
     5  failure during receive from DNS server
     6  DNS lookup returned "no such entry"
     7  timeout waiting for DNS server     8  bad arguments
```

Revision #1
Created Thu, Jan 24, 2019 10:35 AM by Alex
Updated Thu, Jan 24, 2019 10:35 AM by Alex