1. Identification of the proposed change

1.1. Title

Sockets Binding

1.2. MDC Proposer and Sponsor

This proposal originates from Mark Berryman of SAIC. Since the original inception of this proposal, the "baton has been passed" a number of times. Intermediate contributions were offered by Peter Cudhea of InterSystems and David Marcus of Micronetics. The current document editors are Nancy Adams and Ed de Moel.

Motions regarding the status of this document will be made by Taskgroup 14 (Networking) of Subcommittee 12 (Environment).

Ed de Moel can be reached at:

· 800 Nelson Street, Rockville, Maryland 20850-2051
· home phone: 301 762 8333
· office phone: 301 330 7558 extension 759
· telefax: 301 762 8999
· email: demoel@radix.net

1.3. Motion

This document supercedes X11/SC/12/1998-1 as the final print-out of an MDC Type A extension.

1.4. History of MDC actions

Date	Document	Action
June 1998	X11/1998-14	Final print-out
March 1998	X11/SC12/1998-1	Presented for elevation to MDC Type A, accepted 23:0:2.
September 1997	X11/SC12/TG14/97-3	Presented for elevation to SC#12 Type A, accepted 13:0:4.
19 March 1997	X11/SC14/TG6.96-4	Presented for elevation to SC 12 Type B. Accepted 12:0:2.
29 Sept 1996	X11/SC14/TG6/96-3	Presented for elevation to SC 14 Type B. Accepted 7:0.
8 Apr 1996	X11/SC14/TG6/96-1	Discussion document containing comments
Dec 1995	X11/SC14/TG6/95-10	Discussion document containing comments
26 October 1995	X11/SC14/TG6/95-8	Presented for elevation to SC 14 Type B. Accepted 7:0:1.
September 1995	X11/SC14/TG6/95-7	Supported by MDCC-E
4 June 1995	X11/SC14/TG6/95-4	Presented for elevation to SC 14 Type B. Accepted without dissent.
April 1995	X11/SC14/TG6/95-3	Intermediate version
29 January 1995	X11/SC14/TG6/95-1	Presented for elevation to SC 14 Type B. Accepted 7:0:1.
12 June 1994	X11/SC14/TG6/94-6	Presented for elevation to SC 14 Type B. Accepted 10:0.
2 May 1994	X11/SC14/TG6/94-5	Intermediate version
27 February 1994	X11/SC14/TG6/94-3	Presented for rescission. Rescission passed 10:0.
27 February 1994	X11/SC14/TG6/94-1	Presented for rescission. Rescission passed 10:0.
27 February 1994	X11/SC14/TG6/94-2	Presented for elevation to SC 14 Type B. Accepted as amended 10:0.
27 February 1994	X11/SC14/TG6/94-2	Presented for elevation to amend current document. Accepted 9:0:1.
27 February 1994	X11/SC9/91-5	Presented for demotion to SC 14 Type B. Demotion passed 10:0.
25 October 1993	X11/SC14/TG6/93-3	Presented for elevation to SC 14 Type B. Accepted 6:0:1.
25 October 1993	X11/SC14/TG6/93-2	Presented for elevation to SC 14 Type B. Accepted 6:0:1.
20 June 1993	X11/93-29	Presented for elevation to SC 14 Type C, Accepted passed 13:0.
20 June 1993	X11/SC9/91-5	Withdrawn from consideration as MDC Type A. Withdrawal passed 12:0:1.
2 June 1991	X11/SC9/91-5	Presented for elevation to SC 9 Type A. Accepted 7:1:8.
January 1991	X11/SC9/91-2	Hand-out at meeting
9 October 1990	X11/SC9/90-8	Presented for elevation to SC 9 Type A. Accepted 4:0:8.
10 June 1990	X11/SC9/90-8	Presented for elevation to SC 9 Type B, Accepted unanimously.
15 May 1989	X11/SC9/89-16	Initial presentation

1.5. Dependencies

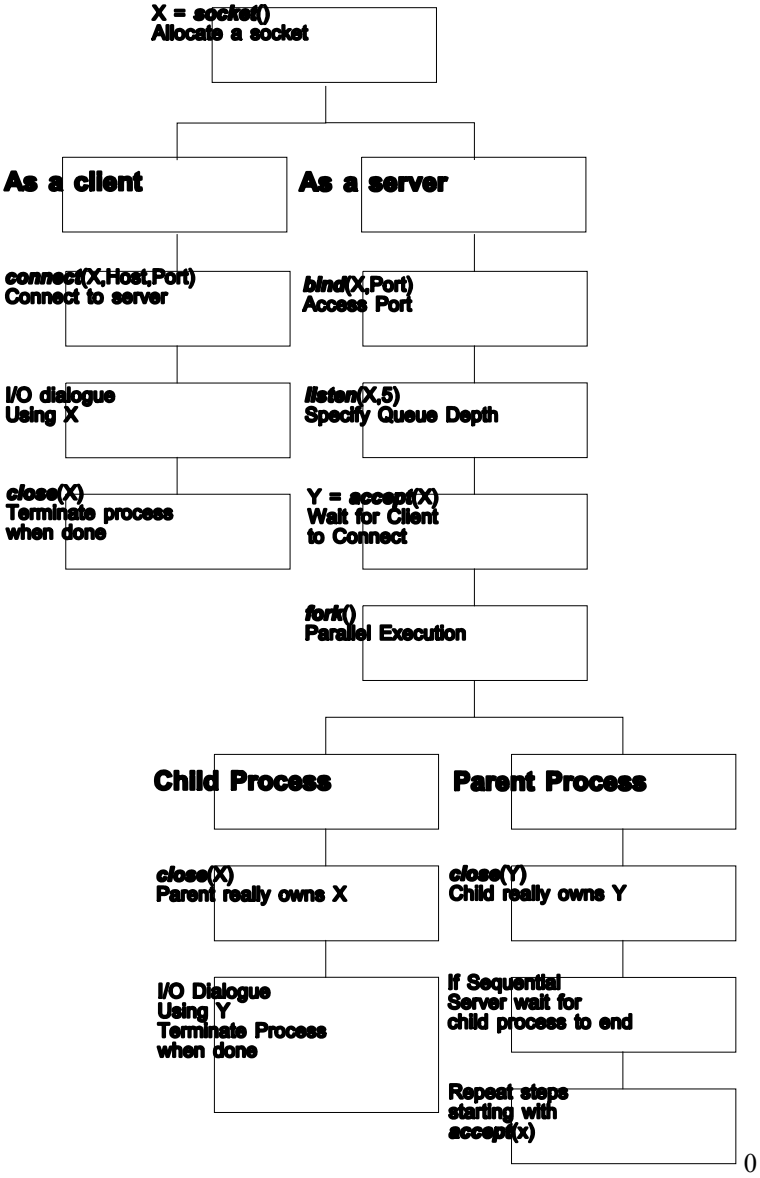
None.

2. Justification of Proposed Change

2.1. Needs

In the computing industry, client/server technology using the TCP/IP and other protocols abound. The MUMPS Language needs to provide **full** access to these networking services. M[UMPS] applications need to be able to fully participate as clients or servers in networks. Work has been done to bind specifically to TCP/IP sockets. The rest of the industry has chosen not to bind to a protocol (i.e. TCP/IP) but to a socket mechanism that can support many protocols. This then is what this proposal attempts to accomplish.

In the industry, sockets are used in client or server mode. The following diagram illustrates the “flow” associated with socket usage.



A client session typically proceeds as follows:

- A socket is allocated
- The socket is used to “connect” to a server. The host name or Network address as well as any subconnection information are specified.
- After a successful connect, an I/O dialogue ensues.
- Close socket, terminate process

A server session typically proceeds as follows:

- A socket is allocated
- The socket is associated with (bound to) a local port number using the ‘bind’ primitive.
- The ‘listen’ primitive is used to establish a queue depth for incoming client connection. UNIX, for example, has a limit of 5.
- The server then uses the ‘accept’ primitive to wait for an actual incoming client connection. This

creates a second socket which represents the actual connection to the client. The original socket is available for accepting additional connections.
- The 'fork' primitive is used to initiate parallel execution.
- The child process closes the original (bound) socket since it is really owned and used by the parent. It then conducts an I/O dialogue with the client using the 'accepted' socket. Finally it closes the socket and terminates.
- The parent process closes the ?accepted' socket since it is really owned and used by the child process. Sequential servers wait for the child process to terminate. Concurrent servers proceed without waiting for the child process to terminate. In both cases, the server repeats the steps starting with the ?accept' (waiting for new clients)

As an objective, an M[UMPS] application should be able to serve as a sequential or a concurrent server (a-la UNIX's *inetd* daemon). This includes the ability to process multiple concurrent clients.

2.2. Existing Practice in Area of the Proposed Change
Currently, no standardized techniques exist. Some implementations have implemented the mnemonicspace defined by document X11/91-5 which is in the process of being demoted into oblivion..

3. Description of the proposed change

3.1. General Description of the Proposed Change
The binding to sockets is accomplished via the ability to associate sockets with M[UMPS] devices. Access to the sockets is provided via controlmnemonics to provide full functionality.

A client process carries out a conversation with a server over one socket. A server process may monitor several different sockets for communication requests from clients.

3.2. Annotated Examples of Use
3.2.1 Client Process
```
; connect to any server providing the service
s timeout=10
o dev:(connect="myservice"):timeout:"SOCKET"
i ?$t w !,"Unable to connect to server" q
u dev
;==> WRITE/READ dialogue with server ...
c dev q

; restrict the search to TCP/IP servers
o dev:(connect="myservice:IP:TCP")::"SOCKET"
i ?$t w !,"Unable to connect to server" q
s sockindx=^$D(dev,"SOCKETINDEX")
s address=^$D(dev,sockindx,"REMOTEADDRESS")
u $P w !,"Connected to server at ",address
u dev
;==> WRITE/READ dialogue with server ...
c dev q

; connect to a specific server and port
o dev:(connect="128.200.1.5:IP:TCP:2001")::"SOCKET" u dev
;==> WRITE/READ dialogue with server ...
c dev q
```

3.2.2 Serial Server Process
```
s timeout=10
```

```
; obtain a socket
o dev:(listen=“:IP:TCP:2001"):timeout:”SOCKET”
i ‘$t w !,”Unable to open server port” q
; establish a queue depth of 3
u dev w /listen(3)
; wait for a client connection
f  u dev w /wait(timeout) q:$device  i $key’=““ d
. ;==> READ/WRITE dialogue with client ...
. s sockindx=^$D(dev,”SOCKETINDEX”)
. s sockhndl=^$D(dev,sockindx,”SOCKETHANDLE”)
. ; close current device
. c dev:(socket=sockhndl)
c dev q
```

3.2.3 Concurrent Server Process

```
s timeout=10
; obtain a socket
o dev:(listen=“:IP:TCP:2001"):timeout:”SOCKET”
i ‘$t w !,”Unable to open server port” q
; establish a queue depth of 3
u dev w /listen(3)
; wait for a client connection
f  u dev w /wait(timeout) q:$device  i $key’=““ d
. s sockindx=^$D(dev,”SOCKETINDEX”)
. s sockhndl=^$D(dev,sockindx,”SOCKETHANDLE”)
. u dev:(detach=sockhndl)
. j server^srv(sockhndl)
c dev q
```

Routine: ^srv

```
            ...
server(sockhndl) ;
            ...
            o dev:(attach=sockhndl)::”SOCKET”
            u dev
            ;==> READ/WRITE dialogue with client
            c dev q
```

3.2.4 Multi Protocol Server Process

```
s timeout=10
o dev:::”SOCKET”  ; acquire M[UMPS] device
u dev:(listen=“:IP:TCP:2001") ; Establish TCP protocol
I $device u $p w !,”Unable to listen for TCP connections” c dev q
s tcpindx=^$D(dev,”SOCKETINDEX”)
s tcphndl=^$D(dev,tcpindx,”SOCKETHANDLE”)
u dev:(listen=“:SPX:1025") ; Establish SPX protocol
I $device u $p w !,”Unable to listen for SPX connections” c dev q
s spxindx=^$D(dev,”SOCKETINDEX”)
s spxhndl=^$D(dev,spxindx,”SOCKETHANDLE”)
; wait for a client connection on either protocol
f  u dev w /wait(timeout) q:$device  i $key’=““ d
. s sockindx=^$D(dev,”SOCKETINDEX”)
. s sockhndl=^$D(dev,sockindx,”SOCKETHANDLE”)
. i $key’=“CONNECT” u $p w “Shouldn’t happen” c dev:(socket=sockhndl) q
. u dev:(detach=sockhndl)
. j server^srv(sockhndl) ; Same server as above example
c dev q
```

3.3. Formalization
Insert the text from Appendix 9 as an informative annex to X11.1.

4. Implementation impacts

4.1. Impact on Existing User Practices and Investments
Existing programs could be modified to take advantage of this proposal.

4.2. Impact on Existing Vendor Practices and Investments
Existing implementations would need to be modified to implement this proposal.

4.3. Techniques and Costs for Compliance Verification
Developing applications which would rely on correct implementation of this proposal.

4.4. Legal considerations
None.

5. Closely related standards activities

5.1. Other X11 Proposals (Type A or Type B) Under Consideration
X11/SC14/TG6/95-7.

5.2. Other Related Standards Efforts
None.

5.3. Recommendations for Co-ordinating Liaison
None.

6. List of Associated Documents

Department of Defence, *RFC # 793, Transmission Control Protocol*; DARPA Internet program Protocol Specification; September 1981.

7. Issues, Pros and Cons, and Discussion

7.1. 10 June 1990, Orlando, Florida
A motion before Subcommittee 9 to accept X11/SC9/90-8, TCP Binding, version 2 as a Type B document of Subcommittee 9 superseding X11/SC9/89-16, TCP Binding, version 1 passed unanimously.

7.2. 9 October 1990, Amsterdam, The Netherlands
A motion before Subcommittee 9 to accept X11/SC9/90-8, TCP Binding, version 2 as amended as a Type A document of Subcommittee 9 superseding X11/SC9/89-16, TCP Binding, version 1 passed by a vote of 4 in favor, none opposed and 8 abstaining.

7.3. 2 June 1991, New Orleans, Louisiana
A motion before Subcommittee 9 to accept X11/SC9/91-5, TCP Binding, version 4 as a Type A document of Subcommittee 9 superseding X11/SC9/91-2, TCP Binding, version 3 passed by a vote of 7 in favor, 1 opposed and 8 abstaining.

7.4. 20 June 1993, Washington, DC

A motion before Subcommittee 14 to withdraw X11/SC9/91-5, TCP Binding, version 4 for consideration as MDC Type A superseding X11/SC9/91-2, TCP Binding, version 3 passed by a vote of 12 in favor, none opposed and 1 abstaining.

A motion before Subcommittee 14 to accept X11/93-29, TCP/IP Binding as a Type C document of Subcommittee 14 passed 13 in favor and none opposed.

7.5. 25 October 1993, Dublin, Ireland

A motion before Subcommittee 14 to accept X11/SC14/TG6/93-2, Revised TCP Binding Cover-Letter, version 1 as a Type B document of Subcommittee 14 superseding X11/93-29, TCP/IP Binding passed by a vote of 6 in favor, none opposed and 1 abstaining.

A motion before Subcommittee 14 to accept X11/SC14/TG6/93-3, TCP-MUMPS Binding Formalization, version 1 as a Type B document of Subcommittee 14 passed by a vote of 6 in favor, none opposed and 1 abstaining.

7.6. 27 February 1994, Houston, Texas

A motion before Subcommittee 14 to accept X11/SC9/91-5, TCP Binding, version 4 (demotion) as a Type B document of Subcommittee 15 superseding X11/SC9/91-2, TCP Binding, version 3 passed 10 in favor and none opposed.

A motion before Subcommittee 14 to amend X11/SC14/TG6/94-2, TCP-MUMPS Binding Formalization, version 2 superseding X11/SC14/TG6/93-3, TCP-MUMPS Binding Formalization, version 1, X11/SC14/TG6/94-1, Revised TCP Binding Cover-Letter, version 2 and X11/SC14/TG6/94-3, TCP-MUMPS Functions, version 1 passed by a vote of 9 in favor, none opposed and 1 abstaining.

A motion before Subcommittee 14 to accept X11/SC14/TG6/94-2, TCP-MUMPS Binding Formalization, version 2 as amended as a Type B document of Subcommittee 14 superseding X11/SC14/TG6/93-3, TCP-MUMPS Binding Formalization, version 1, X11/SC14/TG6/94-1, Revised TCP Binding Cover-Letter, version 2 and X11/SC14/TG6/94-3, TCP-MUMPS Functions, version 1 passed 10 in favor and none opposed.

A motion before Subcommittee 14 to rescind X11/SC14/TG6/94-1, Revised TCP Binding Cover-Letter, version 2 (superseded by X11/SC14/TG6/94-2) superseding X11/SC14/TG6/93-2, Revised TCP Binding Cover-Letter, version 1 passed 10 in favor and none opposed.

A motion before Subcommittee 14 to rescind X11/SC14/TG6/94-3, TCP-MUMPS Functions, version 1 (superseded by X11/SC14/TG6/94-2) passed 10 in favor and none opposed.

7.7. 12 June 1994, Reno, Nevada

A motion before Subcommittee 14 to accept X11/SC14/TG6/94-6, TCP-MUMPS Functions, version 2 as a Type B document of Subcommittee 14 superseding X11/SC9/91-5, TCP Binding, version 4, X11/SC14/TG6/94-3, TCP-MUMPS Functions, version 1 and X11/SC14/TG6/94-5, TCP-MUMPS Binding Formalization, version 3 passed 10 in favor and none opposed.

Summary of changes:
- Reformatted to conform to current MDC formatting standards
- Allow for enabling I/O error trapping on OPEN and USE commands

• Define OPEN and USE syntax, include option for delimiters
• Allow sockets to be selected by host name in addition to IP address
• Update the limit on *port* from 255 to 65,535
• In the definition of **tcp.socket**, define *type* and *protocol* using standard (non-host-specific) terms
• Generate I/O errors on OPEN, USE, READ or WRITE to a socket that is no longer active
• Define **tcp.errno**()
• Add a timeout option to **tcp.send**() and **tcp.listen**()
• Define **tcp.select**()
• Correct the CLOSE command semantics to reference **tcp.close**()
• Clarify the action of the READ command with respect to timeout, fixed length read, read delimiters and "plain" reads
• Update the examples to reflect the changes and the capabilities of the proposal
• The indications of errors have been changed. **tcp.errno**() returns an empty string if no error has occurred, otherwise it returns information pertinent to the error condition.

7.8. 29 January 1995, Albuquerque, New Mexico

A motion before Subcommittee 14 to accept X11/SC14/TG6/95-1, TCP-MUMPS Binding as a Type B document of Subcommittee 14 superseding X11/SC14/TG6/94-6, TCP-MUMPS Functions, version 2 passed by a vote of 7 in favor, none opposed and 1 abstaining.

7.9. April 1995

X11/SC14/TG6/95-2 presented as a total replacement for the document that is currently being considered for elevation to Type A.

Summary of changes:
• Eliminate TCP package
• Limit the TCP binding to the TCP mnemonicspace

7.10. 4 June 1995, Chicago, Illinois

A motion before Subcommittee 14 to accept X11/SC14/TG6/95-4, TCP/IP-MUMPS Mnemonics Binding as amended as a Type B document of Subcommittee 14 superseding X11/SC14/TG6/95-1, TCP-MUMPS Binding and X11/SC14/TG6/95-3, TCP/IP-MUMPS Binding passed without dissent.

Summary of changes:
• Corrected examples to remove references to `&TCP`
• Updated deviceparam to allow (L expr)
• Annex made normative
• Define as "reserved" IP addresses elements not in the 0-255 range
• Specify the initial settings of DELIMITERS to be "empty"
• Miscellaneous cleanup and rewording

7.11. 26 October 1995, New Orleans, Louisiana

A motion before Subcommittee 14 to accept X11/SC14/TG6/95-8, Sockets-MUMPS Mnemonics Binding as a Type B document of Subcommittee 14 superseding X11/SC14/TG6/95-7, TCP-MUMPS Mnemonics Binding passed by a vote of 7 in favor, none opposed and 1 abstaining.

Summary of changes:
• Removed /SHUTDOWN controlmnemonic
• Streamlined the number of socket states

- Added additional introductory explanations
- Removed /SELECT controlmnemonic
- Removed references to STREAM, DATAGRAM and RAW, since TCP is STREAM mode only (UDP is DATAGRAM, etcetera)
- Error conditions added for State Errors
- LISTEN devicekeyword allowed
- LISTEN, CONNECT or SOCKET required when TCP mnemonicspace allocated to device
- DISCONNECT made default on CLOSE
- ^$DEVICE section included
- Miscellaneous cleanup and clarifications

7.12. 29 September 1996, Toronto, Canada

A motion before Subcommittee 14 to accept this document as a Type B document of Subcommittee 14 superseding X11/SC14/TG6/96-1, Socket Requirements Comments and X11/SC14/TG6/96-3, Sockets-MUMPS mnemonics binding, version 2 passed 7 in favor and none opposed.

7.13. 19 March 1997, San Diego, California

A motion before Subcommittee 12 to accept this document as a Type B document of Subcommittee 12 superseding X11/SC14/TG6/96-1, Socket Requirements Comments and X11/SC14/TG6/96-3, Sockets-MUMPS mnemonics binding, version 2 passed by a vote of 12 in favor, none opposed and 2 abstaining.

7.14. September 1997, Chicago, Illinois

No modifications made.
Pro: 1. Long awaited functionality
No cons.
Elevated to SC#12 Type A: 13:0:4.

Appendix - M[UMPS] Language Sockets Binding

Annex X - Sockets Binding

X.1 Introduction

Sockets are used to represent and manage a communication channel between two entities on a network. The channel can be connection-oriented, in which the two entities establish a session for the duration of the conversation, or it can be connectionless, in which messages are simply sent out to the intended recipient.

X.2 General

Socket communications are accessed by the use of controlmnemonics and deviceparameters. This binding uses the SOCKET mnemonicspace.

Socket identifiers (referred to simply as ?sockets') are used by the implementation to identify the ?socket handle' used by the underlying implementation. The actual mapping between socket identifiers and the underlying sockets is implementation-specific.

Sockets are accessed and manipulated via a socket device. The socket device can contain a collection of sockets. At any one time, a single socket from the collection is the current socket. Any socket in the collection can be designated to be the current socket. Furthermore, sockets can be attached (added) and detached (deleted) from the collection of sockets.

X.3 Commands and Deviceparameters

For the SOCKET mnemonicspace, the following deviceattributes are defined. All deviceattributes and devicekeywords beginning with the letter Z (or z) are reserved for the implementation. All others are reserved. Names differing only in the use of corresponding upper and lower case letters are equivalent.

Once a device is successfully OPENed, the structured system variable ^$DEVICE reflects the current settings of the deviceattributes.

An attempt to modify a socket when none is current will result in an error with **ecode = M99** (invalid operation for context). An attempt to specify an invalid argument to a deviceattribute will result in an error with **ecode = M47** (invalid attribute value).

X.3.1 OPEN and USE Commands

The OPEN and USE commands allow sockets to be associated with devices after specifying a mnemonicspace equal to "SOCKET".

The following deviceattributes are valid on an OPEN or USE command.

X.3.1.1 ATTACH = expr

expr specifies an implementation-specific socket identifier. It specifies an existing socket that should be added to this device's collection of sockets. If the socket is attached to any other process or device, the ATTACH will fail with **ecode = M98** (resource unavailable). Otherwise, if the operation is successful, the

attached socket will become the current socket for the device.

X.3.1.2 CONNECT = expr

expr specifies implementation-specific connection information. A client connection will be established with a server, using the connection information to locate the server. A new socket will be allocated for the client connection and will become the current socket for the device.

X.3.1.3 DELIMITER = | expr |
| (L expr) |

expr specifies an I/O delimiter. Each usage of this deviceattribute replaces the existing set of I/O delimiters with a new set (which may be empty). The set is empty if all exprs have the value of the empty string.

If no DELIMITER is specified, the initial set of I/O delimiters for the socket is empty.

X.3.1.4 IOERROR = expr

expr specifies the I/O error trapping mode.

A value equal to "NOTRAP" specifies that I/O errors on a device do not raise error conditions. A value equal to "TRAP" specifies that I/O errors on a device do raise error conditions with an ecode value associated with the error. Values beginning with Z (or z) are reserved for the implementation. All other values are reserved. Values differing only in the use of corresponding upper and lower case letters are equivalent.

If no IOERROR is specified, the initial I/O error trapping mode for a socket is "NOTRAP".

X.3.1.5 LISTEN = expr

expr specifies implementation and protocol specific information. This command causes the device to allocate a new socket and prepare it for listening for incoming requests for connection to a server. The new socket is made the current socket for the device. Requests for connections will not be accepted until a WRITE /controlmnemonic is issued

The following deviceattributes are valid on the USE but not the OPEN command.

X.3.1.6 DETACH = expr

expr specifies an implementation-specific socket identifier. The specified socket is detached from the device without affecting the sockets existing connection. The socket may then be attached to another socket device using the ATTACH deviceattribute.

X.3.1.7 SOCKET = expr

expr specifies an implementation-specific socket identifier. The specified socket becomes the current socket.

X.3.2 CLOSE Command

The following deviceattributes are valid on the CLOSE command.

X.3.2.1 SOCKET = expr

expr specifies an implementation-specific socket identifier which is the socket associated with the device that is to be closed. If this deviceattribute is specified then any other sockets associated with the device are not closed and the M[UMPS] device is not released.

If the SOCKET deviceattribute is omitted then all sockets associated with the device are closed and the M[UMPS] device is released.

X.3.3 READ Command

The READ command may be used to obtain data from a socket.

A READ operation will terminate if any of the following are detected, in the order specified:

- Error condition. $DEVICE reflects the error, $KEY is assigned the empty string. The value returned by the READ command is implementation specific.
- READ timeout. $KEY is assigned the empty string. The READ command returns data received up to the timeout.
- READ delimiter. $KEY is assigned the delimiter string which terminated the READ. The READ command returns data received up to, but not including, the delimiter.
- Fixed-length READ requirements are satisfied. This occurs only after the specified number of characters are received. $KEY is assigned the empty string. The READ command returns the characters received.
- For a stream-oriented protocol, when the buffer is empty the READ waits. When there is at least one character, the READ command returns available characters, up to the maximum string length for the implementation. Note that the number of characters returned is not predictable except to be within the range from one to the maximum string length. $KEY is assigned the empty string.
- For a message-oriented protocol, when a complete message is received, READ returns the message. $KEY is assigned the empty string.

For multi-character I/O delimiters, the possibility exists due to the stream nature of transmissions, that characters which would otherwise match an I/O delimiter may actually be spread across multiple ?packets'. In the event that the last ?*n*' characters received (*n* > 0) match a prefix of one or more I/O delimiters, the implementation must determine if any of the additionally expected characters complete the match with the I/O delimiter(s). One implementation would be to internally issue a timed READ. A timeout of this internally issued timed READ does not affect $TEST. The time associated with this internal timed READ is implementation specific and is not included in the timeout which may have been optionally specified on the actual READ command.

X.3.4 WRITE Command

The WRITE command may be used to send data to a socket.

Data being transmitted is sent using the urgency mode currently in effect for the socket. The definition and usage of urgency mode is implementation-specific.

WRITE ! appends the first I/O delimiter (see X 3.1.3), if specified, to the internal output buffers for the current device. The process then immediately transfers the internally buffered output data to the underlying binding services. This command does not affect internally buffered input data. $X is set to 0. $Y is incremented by 1.

WRITE # causes the process to immediately transfer any internally buffered output data for the current device to the underlying binding services. No I/O delimiters are implicitly added to the internal output buffer. This command does not affect internally buffered input data. $X and $Y are set to 0.

X.4 controlmnemonics

controlmnemonic names differing only in the use of corresponding upper and lower case letters are equivalent.

X.4.1 LISTEN [(expr)]

The use of this controlmnemonic causes the process to establish a queue depth for incoming client connections.

If expr is omitted then the queue depth established will take on an implementation specific value.

X.4.2 WAIT [(numexpr)]

numexpr is a timeout value.

If the optional numexpr is present, the value must be nonnegative. If it is negative, the value 0 is used. numexpr denotes a *t*-second timeout, where *t* is the value of numexpr. If *t* = 0, the condition is tested. If *t* is positive, execution is suspended until the connection is made, but in any case no longer than *t* seconds.

The use of this controlmnemonic causes the process to wait for an event to occur on any socket associated with the device, subject to timeout. When this operation completes, $KEY contains a value identifying the event that occurred.

In the event of a timeout or an error the empty string is returned in $KEY.

If a listening server socket receives a connection request, $KEY will contain the value “CONNECT”. A new socket will be allocated to handle the connection with the client, and the new socket will become the current socket of the device.

If a message is received by a connectionless protocol, $KEY will contain the value “READ”. The socket which received the message will become the current socket of the device.

X.5 ^$DEVICE

The following nodes are defined in ^$DEVICE for the SOCKET mnemonicspace:

`^$DEVICE(device,”SOCKET”)` = intexpr

Each device has a collection of sockets associated with it. Each new socket is identified by a socket identifier which is assigned an index number in the collection of sockets. This node of ^$DEVICE defines the index number of the current socket.

`^$DEVICE(device,”SOCKET”,index,”DELIMITER”)` = intexpr

This provides the number of I/O delimiters, as defined using the DELIMITER deviceattribute, in effect for the device/socket. (See X.3.1.3)

^$DEVICE(device,"SOCKET",index,"DELIMITER",*n*) = expr
This provides the *n*-th I/O delimiter string. (See X.3.1.3)

^$DEVICE(device,"SOCKET",index,"IOERROR") = expr
I/O error trapping mode. (See X.3.1.4)

^$DEVICE(device,"SOCKET",index,"LOCALADDRESS") = expr
This provides the local network node address of the connection

^$DEVICE(device,"SOCKET",index,"PROTOCOL") = expr
This provides the network protocol used for the connection

^$DEVICE(device,"SOCKET",index,"REMOTEADDRESS") = expr
This provides the remote network node address of the connection

^$DEVICE(device,"SOCKET",index,"SOCKETHANDLE") = expr
The value of this node is an implementation-specific string that provides the socket identifier of the indicated socket.