

Namespaces

Namespaces

Namespaces

Namespace	Description
SciSnet	

Send comments on this topic to support@microglyph.com

(c) 2007 MicroGlyph Systems. All rights reserved.

SciSnet Namespace

Namespaces > SciSnet

Syntax

C#

namespace SciSnet Types

All Types

Classes

Structures

Interfaces

Enumerations

Icon

Type

Description



SN

SN Class - SciSnet Sockets Communication Interface Methods

Send comments on this topic to support@microglyph.com

(c) 2007 MicroGlyph Systems. All rights reserved.

SN Class

Namespaces > SciSnet > SN

SN Class - SciSnet Sockets Communication Interface Methods

Syntax

C#

```
public class SN
```

All Members	Constructors	Methods	Properties	Fields
<input checked="" type="checkbox"/> Public		<input checked="" type="checkbox"/> Instance		<input checked="" type="checkbox"/> Decla
<input checked="" type="checkbox"/> Protected		<input checked="" type="checkbox"/> Static		<input checked="" type="checkbox"/> Inher

Icon	Member	Description
	SN()	SN Class - constructor
	NETCLS(Int32, Int32)	Routine to close a network connection
	NETDTM(Int32, Int32, Int32, Int32, Int32, Int32, Int32)	Routine to get date and time.
	NETENC(Int32, Byte[], Int32[], String)	Routine to encode bytes, integers, or strings
	NETERR(String, String, Int32)	Routine to get error messages and error code.
	NETFLS(Int32, Int32)	Routine to flush network messages
	NETOPN(Int32, String, Int32, Int32)	Routine to open a network connections
	NETREC(Int32, Int32, Int32, Byte[], Int32, Int32)	Routine to receive a message buffer
	NETSND(Int32, Int32, Byte[], Int32)	Routine to send a message buffer
	NETSRV(Int32, Int32, Int32, Int32)	Routine to start/stop local host server
	NETSTS(Int32, String, String, String, Int32, Int32, Int32, Int32, Int32, Int32, Int32, Int32)	Routine to get network connection status
	NETSUS(Int32)	Routine to sleep the executing thread.
	NETTMR()	Routine to get current time in milliseconds.

Inheritance

Hierarchy

Object



Assembly: SCISNET (Module: SCISNET)

Send comments on this topic to support@microglyph.com

(c) 2007 MicroGlyph Systems. All rights reserved.

SN Constructor

[Namespaces](#) > [SciSnet](#) > [SN](#) > **SN()**

SN Class - constructor

Syntax

C#

```
public SN () Assembly: SCISNET (Module: SCISNET)
```

Send comments on this topic to support@microglyph.com

(c) 2007 MicroGlyph Systems. All rights reserved.

NETCLS Method (netid, code)

Namespaces > SciSnet > SN > NETCLS(Int32, Int32)

Routine to close a network connection

Syntax

C#

```
public static int NETCLS (  
    int netid,  
    int code  
)
```

Parameters

netid (**Int32**)

Network connection ID
1 thru 100

code (**Int32**)

Function code

- 1 - Purge input and output messages
- 2 - Purge input, wait for output messages

Return Value

Status is returned as an int value.

- 0 - Normal return
- 1 - Invalid arguments
- 2 - System exception
- 3 - SN Class not instantiated
- 4 - Connection not open
- 5 - Server not running

Assembly: SCISNET (Module: SCISNET)

Send comments on this topic to support@microglyph.com

(c) 2007 MicroGlyph Systems. All rights reserved.

NETDTM Method (month, day, year, hour, min, sec, msec)

Namespaces > SciSnet > SN > NETDTM(Int32, Int32, Int32, Int32, Int32, Int32, Int32)



Routine to get date and time.

Syntax

C#

```
public static int NETDTM (  
    ref int month,  
    ref int day,  
    ref int year,  
    ref int hour,  
    ref int min,  
    ref int sec,  
    ref int msec  
)
```

Parameters

month (**Int32**)

Month

day (**Int32**)

Day

year (**Int32**)

Year

hour (**Int32**)

Hour

min (**Int32**)

Minute

sec (**Int32**)

Second

msec (**Int32**)

Millisecond

Return Value

Status is returned as an int value.

- 0 - Normal return
- 1 - System exception
- 2 - SN Class not instantiated

Assembly: SCISNET (Module: SCISNET)

Send comments on this topic to support@microglyph.com

(c) 2007 MicroGlyph Systems. All rights reserved.

NETENC Method (code, bytbuf, intbuf, svalue)

Namespaces > [SciSnet](#) > [SN](#) > **NETENC(Int32, Byte[], Int32[], String)**

Routine to encode bytes, integers, or strings

[-] Syntax

C#

```
public static int NETENC (  
    int code,  
    byte[] bytbuf,  
    int[] intbuf,  
    ref string svalue
```

[-] Parameters

code (**Int32**)

Type of encoding

- 1 - int to Byte
- 2 - Byte to int
- 3 - string to Byte (ASCII encoding)
- 4 - string to Byte (Unicode encoding)
- 5 - Byte to string (ASCII encoding)
- 6 - Byte to string (Unicode encoding)

bytbuf (**Byte**[])

Byte array(4 times intbuf size)

intbuf (**Int32**[])

Integer array

svalue (**String**)

String value

[-] Return Value

Status is returned as an int value.

- 0 - Normal return
- 1 - Invalid arguments
- 2 - System exception
- 3 - SN Class not instantiated

Assembly: SCISNET (Module: SCISNET)

Send comments on this topic to support@microglyph.com

(c) 2007 MicroGlyph Systems. All rights reserved.

NETERR Method (Istexc, Istrtn, Isterr)

Namespaces > [SciSnet](#) > [SN](#) > NETERR(String, String, Int32)

Routine to get error messages and error code.

[-] Syntax

C#

```
public static int NETERR (  
    ref string Istexc,  
    ref string Istrtn,  
    ref int Isterr
```

[-] Parameters

Istexc (**String**)

Last SciSnet Exception

Istrtn (**String**)

Last SciSnet routine in error

Isterr (**Int32**)

Last SciSnet routine error code

[-] Return Value

Status is returned as an int value.

- 0 - Normal return
- 1 - Invalid arguments
- 2 - SN Class not instantiated
- 3 - System exception

[-] Remarks

Method clears current error messages and code.

Assembly: SCISNET (Module: SCISNET)

Send comments on this topic to support@microglyph.com

(c) 2007 MicroGlyph Systems. All rights reserved.

NETFLS Method (netid, code)

Namespaces > [SciSnet](#) > [SN](#) > NETFLS(Int32, Int32)

Routine to flush network messages

Syntax

C#

```
public static int NETFLS (  
    int netid,  
    int code  
)
```

Parameters

netid (**Int32**)

Network connection ID
1 thru 100

code (**Int32**)

Function code

- 1 - Flush receive messages
- 2 - Flush send messages
- 3 - Flush receive and send messages

Return Value

Status is returned as an int value.

- 0 - Normal return
- 1 - Invalid arguments
- 2 - System exception
- 3 - SN Class not instantiated
- 4 - Connection not open
- 5 - Server not running

Assembly: SCISNET (Module: SCISNET)

Send comments on this topic to support@microglyph.com

(c) 2007 MicroGlyph Systems. All rights reserved.

NETOPN Method (netid, rname, rport, rtime)

Namespaces > [SciSnet](#) > [SN](#) > NETOPN(Int32, String, Int32, Int32)

Routine to open a network connections

Syntax

C#

```
public static int NETOPN (  
    int netid,  
    string rname,  
    int rport,  
    int rtime  
)
```

Parameters

netid (**Int32**)

Network connection ID (1 - 100)

rname (**String**)

Remote host name or IP address

rport (**Int32**)

Remote host receive port(1-65535)

rtime (**Int32**)

Remote host connection timeout(ms)

Return Value

Status is returned as an int value.

- 0 - Normal return
- 1 - Invalid arguments
- 2 - System exception
- 3 - SN Class not instantiated
- 4 - Connection already open
- 5 - Server not running
- 6 - Duplicate remote host name
- 7 - Remote host not found
- 8 - Remote host connection failed
- 9 - Remote host connection timed out

Assembly: SCISNET (Module: SCISNET)

Send comments on this topic to support@microglyph.com

(c) 2007 MicroGlyph Systems. All rights reserved.

NETREC Method (netid, code, timeout, outbuf, limit, nbytes)

Namespaces > SciSnet > SN > NETREC(Int32, Int32, Int32, Byte[], Int32, Int32)



Routine to receive a message buffer

Syntax

C#

```
public static int NETREC (  
    int netid,  
    int code,  
    int timeout,  
    byte[] outbuf,  
    int limit,  
    ref int nbytes
```

Parameters

netid (**Int32**)

Network connection ID
1 thru 100

code (**Int32**)

Option code

- 1 - Normal read
- 2 - Read with timeout

timeout (**Int32**)

Timeout value (ms)

outbuf (**Byte[]**)

Output message buffer

limit (**Int32**)

Maximum length for outbuf

nbytes (**Int32**)

Number of bytes received

Return Value

Status is returned as an int value.

- 0 - Normal return
- 1 - Invalid arguments
- 2 - System exception
- 3 - SN Class not instantiated
- 4 - Connection not open
- 5 - Message truncated to limit
- 6 - Transmission errors detected
- 7 - Receive timed out
- 8 - Server not running

Assembly: SCISNET (Module: SCISNET)

Send comments on this topic to support@microglyph.com

(c) 2007 MicroGlyph Systems. All rights reserved.

NETSND Method (netid, code, outbuf, nbytes)

Namespaces > [SciSnet](#) > [SN](#) > NETSND(Int32, Int32, Byte[], Int32)

Routine to send a message buffer

Syntax

C#

```
public static int NETSND (  
    int netid,  
    int code,  
    byte[] outbuf,  
    int nbytes  
)
```

Parameters

netid (**Int32**)

Network connection ID
1 thru 100

code (**Int32**)

Option code

- 1 - Send message
- 2 - Wait on message to be sent
- 3 - Timestamp message
- 4 - Wait + timestamp

outbuf (**Byte[]**)

Message buffer bytes array

nbytes (**Int32**)

Number of bytes to send

Return Value

Status is returned as an int value.

- 0 - Normal return
- 1 - Invalid arguments
- 2 - System exception
- 3 - SN Class not instantiated
- 4 - Connection not open
- 5 - Memory exhausted
- 6 - Transmission errors detected
- 7 - Server not running

Assembly: SCISNET (Module: SCISNET)

Send comments on this topic to support@microglyph.com

(c) 2007 MicroGlyph Systems. All rights reserved.

NETSRV Method (code, lport, rlen, slen)

Namespaces > SciSnet > SN > NETSRV(Int32, Int32, Int32, Int32)

Routine to start/stop local host server

Syntax

C#

```
public static int NETSRV (  
    int code,  
    int lport,  
    int rlen,  
    int slen  
)
```

Parameters

code (**Int32**)

Function code

- 1 - Start server
- 2 - Resume server
- 3 - Stop server

lport (**Int32**)

Local server port (1 - 65535)

rlen (**Int32**)

Receive TCP/IP buffer size

slen (**Int32**)

Send TCP/IP buffer size

Return Value

Status is returned as an int value.

- 0 - Normal return
- 1 - Invalid arguments
- 2 - System exception
- 3 - SN Class not instantiated
- 4 - Server already started
- 5 - Server already stopped
- 6 - Receive thread failed to start
- 7 - Send thread failed to start
- 8 - Timeout thread failed to start
- 9 - Receive thread failed to stop
- 0 - Send thread failed to stop
- 1 - Timeout thread failed to stop
- 2 - Memory allocation error

Assembly: SCISNET (Module: SCISNET)

Send comments on this topic to support@microglyph.com

(c) 2007 MicroGlyph Systems. All rights reserved.

NETSTS Method (netid, lname, laddr, rname, raddr, nrmsg, nsmsg, nrcnt, nscnt, nrerr, nserr, rerr, serr)

Namespaces > SciSnet > SN > NETSTS(Int32, String, String, String, String, Int32, Int32,

Int32, Int32, Int32, Int32, Int32, Int32)

Routine to get network connection status

Syntax

C#

```
public static int NETSTS (  
    int netid,  
    ref string lname,  
    ref string laddr,  
    ref string rname,  
    ref string raddr,  
    ref int nrmsg,  
    ref int nsmsg,  
    ref int nrcnt,  
    ref int nscnt,  
    ref int nrerr,  
    ref int nserr,  
    ref int rerr,  
    ref int serr  
)
```

Parameters

netid (**Int32**)
Network connection ID (1-100)

lname (**String**)
Local host name

laddr (**String**)
Local host ip address

rname (**String**)
Remote host name

raddr (**String**)
Remote host ip address

nrmsg (**Int32**)
Number of receive messages

nsmsg (**Int32**)
Number of send messages

nrcnt (**Int32**)
Number bytes receive msgs

nscnt (**Int32**)
Number bytes send msgs

nrerr (**Int32**)
Number receive msgs errors

nserr (**Int32**)
Number send msgs errors

rerr (**Int32**)
Receive message error

serr (**Int32**)
Send message error

Return Value

Status is returned as an int value.

- 0 - Normal return
- 1 - Invalid arguments

- 2 - System exception
- 3 - SN Class not instantiated
- 4 - Connection not open
- 5 - Transmission errors detected
- 6 - Server not running

Assembly: SCISNET (Module: SCISNET)

Send comments on this topic to support@microglyph.com

(c) 2007 MicroGlyph Systems. All rights reserved.

NETSUS Method (tmeout)

Namespaces > SciSnet > SN > NETSUS(Int32)

Routine to sleep the executing thread.

Syntax

C#

```
public static int NETSUS (  
    int tmeout  
)
```

Parameters

tmeout (**Int32**)

Time out value

- 0 - Give up time slice to any other
- xxx - Sleep current thread for xxx

Return Value

Status is returned as an int value.

- 0 - Normal return
- 1 - Argument error
- 2 - System exception
- 3 - SN Class not instantiated

Assembly: SCISNET (Module: SCISNET)

Send comments on this topic to support@microglyph.com

(c) 2007 MicroGlyph Systems. All rights reserved.

NETTMR Method

[Namespaces](#) > [SciSnet](#) > [SN](#) > **NETTMR()**



Routine to get current time in milliseconds.

Syntax

C#

```
public static int NETTMR ()
```

Return Value

Current time (milliseconds) is returned as an int value.

Assembly: SCISNET (Module: SCISNET)

Send comments on this topic to support@microglyph.com

(c) 2007 MicroGlyph Systems. All rights reserved.

Example Programs

Example Programs are provided to demonstrate the usage of SciSnet Library methods.

The following links provide access to the example programs:

[TNET](#)

Use of SciComm Library Methods.

```

!*****
!
!   TNET - Program to demo SciSnet Sockets Communication Library   *
!   *
!*****

!   Main Control Program

        PROGRAM MAIN

!.nca (MAIN,add=System.STAThreadAttribute())

        USE System
        USE System.Text
        USE MicroGlyph.SciSnet

        IMPLICIT NONE

!   Instantiate MicroGlyph.SciSnet Class

        TYPE(SN)          :: SNClass
        TYPE(SN), POINTER :: SNOBJECT
        ALLOCATE(SNOBJECT, SOURCE=SN())

!   Local variables

        INTEGER(4) :: status, i, j, NetID, lport, rport, rtime, rlen, slen, code, &
& nrmsg, nsmsg, nrcnt, nscnt, nrerr, nserr, rerr, serr, lsterr, nbytes, &
& func, limit, tmeout, j1, j2, n, rtnsts, ival
        LOGICAL(4) :: dialog, sflag
        UIINTEGER(KIND=1), POINTER, DIMENSION(:) :: TmpBuf
        UIINTEGER(KIND=1), POINTER, DIMENSION(:) :: BytBuf
        INTEGER(4), POINTER, DIMENSION(:) :: IntBuf
        CHARACTER(1) :: reply
        CHARACTER(2) :: CRLF
        CHARACTER(5) :: ifmt, iline
        TYPE(System.Exception), POINTER :: se
        TYPE(String), POINTER :: line, pline, rname, raddr, lname, laddr, &
& lstexc, lststrn

!   Instantiate arrays

        ALLOCATE(IntBuf(80))
        CRLF = CHAR(13) // CHAR(10)
        ifmt = '(I)'
        limit = 80

!   Start Main

        TRY

!   Dialog loop

        dialog = .TRUE.
        sflag = .FALSE.
        DO WHILE (dialog)

                CALL Console.WriteLine( USTRING( CRLF // " TNET Functions:" &
& // CRLF))

```

```

CALL Console.WriteLine(USTRING(                                     &
&     "    N = Server      (NETSRV) ")                               &
CALL Console.WriteLine(USTRING(                                     &
&     "    O = Open       (NETOPN) ")                               &
CALL Console.WriteLine(USTRING(                                     &
&     "    C = Close      (NETCLS) ")                               &
CALL Console.WriteLine(USTRING(                                     &
&     "    S = Status     (NETSTS) ")                               &
CALL Console.WriteLine(USTRING(                                     &
&     "    R = Read       (NETREC) ")                               &
CALL Console.WriteLine(USTRING(                                     &
&     "    W = Write      (NETSND) ")                               &
CALL Console.WriteLine(USTRING(                                     &
&     "    F = Flush      (NETFLS) ")                               &
CALL Console.WriteLine(USTRING(                                     &
&     "    M = Display Menu"))                                       &
CALL Console.WriteLine(USTRING(                                     &
&     "    Q = Quit/Exit"))                                           &
CALL Console.Write(USTRING(                                         &
&         CRLF // " Enter Function: ")                               &
line => Console.ReadLine()
reply = ASTRING(line)
ival = ICHAR(reply)
IF (ival.gt.96) ival = ival - 32
reply = CHAR(ival)

! Process reply

SELECT CASE (reply)

CASE (' ')

!     Blank character

CASE ('N')

!     Network server

CALL Console.WriteLine(USTRING(CRLF //                               &
&     "    Network Server Functions:" // CRLF))                       &
CALL Console.WriteLine(USTRING(                                     &
&     "    1 = Start Network Server"))                               &
CALL Console.WriteLine(USTRING(                                     &
&     "    2 = Resume Network Server"))                             &
CALL Console.WriteLine(USTRING(                                     &
&     "    3 = Stop Network Server"))                               &
IF (sflag)                                                         &
&     THEN                                                         &
&         CALL Console.Write(USTRING(CRLF //                         &
&         "    Enter Function[3]: "))                                 &
&         code = 3
&         sflag = .FALSE.
&     ELSE
&         CALL Console.Write(USTRING(CRLF //                         &
&         "    Enter Function                                     [1]: ")) &
&         code = 1
&         sflag = .TRUE.
END IF
line => Console.ReadLine()
IF (line.Length.GT.0) THEN

```

```

        iline = ASTRING(line)
        READ(iline,ifmt)ival
!       Set selected value
        code = ival
    END IF
    IF (code.EQ.1) THEN
!       Set local listing port
        CALL Console.Write(USTRING(
&           "      Enter Local Listing Port      [8001]: "))
        lport = 8001
        line => Console.ReadLine()
        IF (line.Length.GT.0) THEN
!           iline = ASTRING(line)
            READ(iline,ifmt)ival
!           Set selected value
            lport = ival
        END IF
!       Set receive buffer length
        CALL Console.Write(USTRING(
&           "      Enter Receive Buffer Length [1440]: "))
        rlen = 1440
        line => Console.ReadLine()
        IF (line.Length.GT.0) THEN
!           iline = ASTRING(line)
            READ(iline,ifmt)ival
!           Set selected value
            rlen = ival
        END IF
!       Set send buffer length
        CALL Console.Write(USTRING(
&           "      Enter Send Buffer Length      [1440]: "))
        slen = 1440
        line => Console.ReadLine()
        IF (line.Length.GT.0) THEN
!           iline = ASTRING(line)
            READ(iline,ifmt)ival
!           Set selected value
            slen = ival
        END IF
    END IF
    status = SN.NETSRV(code,lport,rlen,slen)
    IF (status.NE.0) THEN
&       CALL Console.WriteLine(USTRING(
&           CRLF // "      NETSRV Errors:" // CRLF))
&       CALL Console.WriteLine(USTRING("      Status = {0}"), &
&           status)
&       CALL Console.WriteLine(USTRING("      Lport  = {0}"), &
&           lport)
&       CALL Console.WriteLine(USTRING("      Rlen   = {0}"), &
&           rlen)
&       CALL Console.WriteLine(USTRING("      Slens  = {0}"), &
&           slen)
    END IF

CASE ('O')

!       Open network connection

!       Get NetID
        CALL Console.Write(USTRING(CRLF //
&

```

```

&          "      Enter Network Connection          [1]: ")
NetID = 1
line => Console.ReadLine()
IF (line.Length.GT.0) THEN
    iline = ASTRING(line)
    READ(iline,ifmt)ival
!      Set selected value
    NetID = ival
END IF
!
Get remote host name
CALL Console.WriteLine(USTRING(
&          "      Enter Remote Host Name          : ")
&
line => Console.ReadLine()
IF (line.Length.GT.0)
&
THEN
    rname => line
ELSE
!      Nothing entered
    rname => USTRING("")
END IF
!
Get remote host port
CALL Console.WriteLine(USTRING(
&          "      Enter Remote Host Port          [8001]: ")
&
rport = 8001
line => Console.ReadLine()
IF (line.Length.GT.0) THEN
    iline = ASTRING(line)
    READ(iline,ifmt)ival
!      Set selected value
    rport = ival
END IF
!
Get remote host connect timeout value
CALL Console.WriteLine(USTRING(
&          "      Enter Remote Host Connect Timeout [0]: ")
&
rtime = 0
line => Console.ReadLine()
IF (line.Length.GT.0) THEN
    iline = ASTRING(line)
    READ(iline,ifmt)ival
!      Set selected value
    rtime = ival
END IF
status = SN.NETOPN(NetID,rname,rport,rtime)
IF (status.NE.0) THEN
    CALL Console.WriteLine(USTRING(
&          CRLF // "      NETOPN Errors:" // CRLF))
&
    CALL Console.WriteLine(USTRING("      Status = {0}"),
&
&          status)
    CALL Console.WriteLine(USTRING("      NetID = {0}"),
&
&          NetID)
    CALL Console.WriteLine(USTRING("      Rname = {0}"),
&
&          rname)
    CALL Console.WriteLine(USTRING("      Rport = {0}"),
&
&          rport)
    CALL Console.WriteLine(USTRING("      Rtime = {0}"),
&
&          rtime)
&
END IF

CASE ('C')

```

```

!      Close network connection

      CALL Console.WriteLine(USTRING(CRLF //
&          "      Close Functions:" // CRLF))
      CALL Console.WriteLine(USTRING(
&          "          1 = Purge Input/Output Messages"))
      CALL Console.WriteLine(USTRING(
&          "          2 = Purge Input/Wait on" //
&          " Output Messages"))
      CALL Console.Write(USTRING(CRLF //
&          "      Enter Function          [1]: "))
      code = 1
      line => Console.ReadLine()
      IF (line.Length.GT.0) THEN
          iline = ASTRING(line)
          READ(iline,ifmt)ival
!          Set selected value
          code = ival
      END IF
!      Get NetID
      CALL Console.Write(USTRING(
&          "      Enter Network Connection [1]: "))
      NetID = 1
      line => Console.ReadLine()
      IF (line.Length.GT.0) THEN
          iline = ASTRING(line)
          READ(iline,ifmt)ival
!          Set selected value
          NetID = ival
      END IF
      status = SN.NETCLS(NetID,code)
      IF (status.NE.0) THEN
&          CALL Console.WriteLine(USTRING(
&              CRLF // "      NETCLS Errors:" // CRLF))
&          CALL Console.WriteLine(USTRING("      Status = {0}"),
&              status)
&          CALL Console.WriteLine(USTRING("      NetID = {0}"),
&              NetID)
&          CALL Console.WriteLine(USTRING("      Code = {0}"),
&              code)
      END IF

      CASE ('M')
!          Menu command

      CASE ('S')
!          Network connection status

      CALL Console.Write(USTRING(CRLF //
&          "      Enter Network Connection [1]: "))
      NetID = 1
      line => Console.ReadLine()
      IF (line.Length.GT.0) THEN
          iline = ASTRING(line)
          READ(iline,ifmt)ival
!          Set selected value
          NetID = ival
      END IF

```

```

status = SN.NETSTS (NetID, lname, laddr, rname, raddr, nrmsg,      &
&                                nsmg, nrcnt, nscnt, nrerr, nserr, rerr, serr)
IF (status.EQ.0)                                               &
&     THEN
&         CALL Console.WriteLine (USTRING (CRLF //           &
&             "          NetID = {0}"), NetID)
&         CALL Console.WriteLine (USTRING ("          Lname = {0}"), &
&             lname)
&         CALL Console.WriteLine (USTRING ("          Laddr = {0}"), &
&             laddr)
&         CALL Console.WriteLine (USTRING ("          Rname = {0}"), &
&             rname)
&         CALL Console.WriteLine (USTRING ("          Raddr = {0}"), &
&             raddr)
&         CALL Console.WriteLine (USTRING ("          Nrmsg = {0}"), &
&             nrmsg)
&         CALL Console.WriteLine (USTRING ("          Nsmg = {0}"), &
&             nsmg)
&         CALL Console.WriteLine (USTRING ("          Nrcnt = {0}"), &
&             nrcnt)
&         CALL Console.WriteLine (USTRING ("          Nscnt = {0}"), &
&             nscnt)
&         CALL Console.WriteLine (USTRING ("          Nrerr = {0}"), &
&             nrerr)
&         CALL Console.WriteLine (USTRING ("          Nserr = {0}"), &
&             nserr)
&         CALL Console.WriteLine (USTRING ("          Rerr = {0}"), &
&             rerr)
&         CALL Console.WriteLine (USTRING ("          Serr = {0}"), &
&             serr)
&     ELSE
&         CALL Console.WriteLine (USTRING (                   &
&             CRLF // "          NETSTS Errors:" // CRLF))
&         CALL Console.WriteLine (USTRING ("          Status = {0}"), &
&             status)
&         CALL Console.WriteLine (USTRING ("          NetID = {0}"), &
&             NetID)
&     END IF

CASE ('R')

!     Read bytes from receive messages

&     CALL Console.WriteLine (USTRING (CRLF //           &
&         "          Receive Functions:" // CRLF))
&     CALL Console.WriteLine (USTRING ("          1 = Normal Read"))
&     CALL Console.WriteLine (USTRING (                   &
&         "          2 = Read with Timeout"))
&     CALL Console.WriteLine (USTRING (CRLF //           &
&         "          Enter Function          [1]: "))

code = 1
line => Console.ReadLine()
IF (line.Length.GT.0) THEN
iline = ASTRING(line)
READ(iline, ifmt) ival
!     Set selected value
code = ival
END IF
IF (code.EQ.2) THEN
!     Get timeout value

```

```

CALL Console.WriteLine(USTRING(                                     &
&     "    Enter Read Timeout Value [0]: ")
tmeout = 0
line => Console.ReadLine()
IF (line.Length.GT.0) THEN
    iline = ASTRING(line)
    READ(iline,ifmt)ival
!     Set selected value
    tmeout = ival
    END IF
END IF
! Get network connection ID
CALL Console.WriteLine(USTRING(                                     &
&     "    Enter Network Connection [1]: ")
NetID = 1
line => Console.ReadLine()
IF (line.Length.GT.0) THEN
    iline = ASTRING(line)
    READ(iline,ifmt)ival
!     Set selected value
    NetID = ival
END IF
! Read a message from a remote host
ALLOCATE(TmpBuf(limit))
status = SN.NETREC(NetID,code,tmeout,TmpBuf,limit,nbytes)
IF (status.EQ.0)                                                 &
&     THEN
!     Display bytes received
    IF (nbytes.EQ.0) THEN
&         CALL Console.WriteLine(USTRING(                             &
&             CRLF // "    No data available "                         &
&             // CRLF))
&         CALL Console.WriteLine(USTRING(                             &
&             "    NetID = {0}"),NetID)                               &
&         CALL Console.WriteLine(USTRING(                             &
&             "    Code   = {0}"),code)                               &
    END IF
    IF (nbytes.GT.0) THEN
!     ALLOCATE(BytBuf(nbytes))
        Move number of bytes read into Buffer
        DO i = 1, nbytes
            BytBuf(i) = TmpBuf(i)
        END DO
&         CALL Console.WriteLine(USTRING(CRLF //                         &
&             "    Count: {0}"),nbytes)
!     Convert bytes read to ASCII string
        code = 5
        rtnsts = SN.NETENC(code,BytBuf,IntBuf,line)
!     Display ASCII string
        DO i = 1 , nbytes, 32
            n = Min(32,nbytes-i+1)
            pline => line.Substring(i-1,n)
            IF (i.EQ.1)                                             &
&                 THEN
&                     CALL Console.WriteLine(USTRING(                 &
&                         "    ASCII: {0}"),pline)
&                 ELSE
&                     CALL Console.WriteLine(USTRING(                 &
&                         "    : {0}"),pline)
&                 END IF
        END IF

```

```

        END DO
!       Display Hex string
        DO i = 1 , nbytes, 11
            j1 = i
            j2 = i + Min(11,nbytes-i+1) - 1
            IF (i.EQ.1) &
&           THEN
&               CALL Console.WriteLine(USTRING( &
&                   "           Hex : ")
&               DO j = j1 , j2
&                   CALL Console.WriteLine(USTRING( &
&                       "{0,2:X2} " ),BytBuf(j))
&               END DO
&               CALL Console.WriteLine(USTRING(" "))
            ELSE
&               CALL Console.WriteLine(USTRING( &
&                   "           : ")
&               DO j = j1 , j2
&                   CALL Console.WriteLine(USTRING( &
&                       "{0,2:X2} " ),BytBuf(j))
&               END DO
&               CALL Console.WriteLine(USTRING(" "))
            END IF
        END DO
    END IF
END DO
END IF
ELSE
&   CALL Console.WriteLine(USTRING( &
&       CRLF // "           NETREC Errors:" // CRLF))
&   CALL Console.WriteLine(USTRING("           Status = {0}"), &
&       status)
&   CALL Console.WriteLine(USTRING("           NetID = {0}"), &
&       NetID)
&   CALL Console.WriteLine(USTRING("           Code = {0}"), &
&       code)
END IF

CASE ('W')
!   Write out bytes to send buffer

&   CALL Console.WriteLine(USTRING(CRLF // &
&       "           Write Functions:" // CRLF))
&   CALL Console.WriteLine(USTRING("           1 = Send Message"))
&   CALL Console.WriteLine(USTRING( &
&       "           2 = Wait on Message to be send"))
&   CALL Console.WriteLine(USTRING( &
&       "           3 = Timestamp Message"))
&   CALL Console.WriteLine(USTRING( &
&       "           4 = Wait + Timestamp"))
&   CALL Console.WriteLine(USTRING(CRLF // &
&       "           Enter Function           [1]: "))
&   code = 1
&   line => Console.ReadLine()
&   IF (line.Length.GT.0) THEN
&       iline = ASTRING(line)
&       READ(iline,ifmt)ival
!       Set selected value
&       code = ival
&   END IF
!   Get network connection ID

```

```

CALL Console.WriteLine(USTRING(                                     &
&      "      Enter Network Connection [1]: ")
NetID = 1
line => Console.ReadLine()
IF (line.Length.GT.0) THEN
    iline = ASTRING(line)
    READ(iline,ifmt)ival
!      Set selected value
    NetID = ival
END IF
CALL Console.WriteLine(USTRING(                                     &
&      "      Enter String                                     : ")
!      line => Console.ReadLine()
Convert string to ASCII bytes
func = 3
nbytes = line.Length
ALLOCATE(BytBuf(nbytes))
status = SN.NETENC(func,BytBuf,IntBuf,line)
!      Send the ASCII bytes
status = SN.NETSND(NetID,code,BytBuf,nbytes)
IF (status.NE.0) THEN
    CALL Console.WriteLine(USTRING(                                     &
&          CRLF // "      NETSND Errors:" // CRLF))
    CALL Console.WriteLine(USTRING("      Status = {0}"), &
&          status)
    CALL Console.WriteLine(USTRING("      NetID = {0}"), &
&          NetID)
    CALL Console.WriteLine(USTRING("      Code = {0}"), &
&          code)
END IF

CASE ('F')
!      Flush receive and send buffers

CALL Console.WriteLine(USTRING(CRLF //                                     &
&      "      Flush Functions:" // CRLF))
CALL Console.WriteLine(USTRING(                                     &
&      "      1 = Flush Receive Messages"))
CALL Console.WriteLine(USTRING(                                     &
&      "      2 = Flush Send Messages"))
CALL Console.WriteLine(USTRING(                                     &
&      "      3 = Flush Receive/Send Messages"))
CALL Console.WriteLine(USTRING(CRLF //                                     &
&      "      Enter Function                                     [1]: "))
code = 1
line => Console.ReadLine()
IF (line.Length.GT.0) THEN
    iline = ASTRING(line)
    READ(iline,ifmt)ival
!      Set selected value
    code = ival
END IF
!      Get network connection ID
CALL Console.WriteLine(USTRING(                                     &
&      "      Enter Network Connection [1]: ")
NetID = 1
line => Console.ReadLine()
IF (line.Length.GT.0) THEN
    iline = ASTRING(line)

```

```

        READ(iline,ifmt)ival
!       Set selected value
        NetID = ival
    END IF
    status = SN.NETFLS(NetID,code)
    IF (status.NE.0) THEN
        CALL Console.WriteLine(USTRING(
&           CRLF // "          NETFLS Errors:" // CRLF)) &
        CALL Console.WriteLine(USTRING("          Status = {0}"), &
&           status) &
        CALL Console.WriteLine(USTRING("          NetID  = {0}"), &
&           NetID) &
        CALL Console.WriteLine(USTRING("          Code   = {0}"), &
&           code) &
    END IF

    CASE ('Q')
!       Exit processing

        CALL Console.WriteLine(USTRING(" "))
        dialog = .FALSE.

    CASE DEFAULT
!       Unknown command

    END SELECT

END DO

STOP

CATCH (se)
!       Report exception in Main Program

        CALL Console.WriteLine(USTRING(
&           " Main Program Exception {0}"),se.Message) &

END TRY

END PROGRAM

```