

## Namespaces

Namespaces

Namespaces

Namespace	Description
<a href="#">SciSnet</a>	

Send comments on this topic to [support@microglyph.com](mailto:support@microglyph.com)

(c) 2007 MicroGlyph Systems. All rights reserved.

## SciSnet Namespace

Namespaces > SciSnet

Syntax

Visual Basic

Namespace SciSnet  Types


All Types

Classes

Structures

Interfaces

Enumerations

Icon	Type	Description
	<b>SN</b>	SN Class - SciSnet Sockets Communication Interface Methods

Send comments on this topic to [support@microglyph.com](mailto:support@microglyph.com)

(c) 2007 MicroGlyph Systems. All rights reserved.

# SN Class

Namespaces > SciSnet > SN

SN Class - SciSnet Sockets Communication Interface Methods

## Syntax

Visual Basic

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

## Inheritance

### Hierarchy

Object



Assembly: SCISNET (Module: SCISNET)

Send comments on this topic to [support@microglyph.com](mailto:support@microglyph.com)

(c) 2007 MicroGlyph Systems. All rights reserved.

## SN Constructor

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

SN Class - constructor

**Syntax**

Visual Basic

**Public Sub** NewAssembly: SCISNET (Module: SCISNET)

Send comments on this topic to [support@microglyph.com](mailto: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

Visual Basic

```
Public Shared Function NETCLS ( _  
    netid As Integer, _  
    code As Integer _  
) As Integer
```

### 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](mailto: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

Visual Basic

```
Public Shared Function NETDTM ( _  
    ByRef month As Integer, _  
    ByRef day As Integer, _  
    ByRef year As Integer, _  
    ByRef hour As Integer, _  
    ByRef min As Integer, _  
    ByRef sec As Integer, _  
    ByRef msec As Integer _  
) As Integer
```

### 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](mailto: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

Visual Basic

```
Public Shared Function NETENC ( _  
    code As Integer, _  
    bytbuf As Byte(), _  
    intbuf As Integer(), _  
    ByRef svalue As String _  
) As Integer
```

### 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](mailto: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

Visual Basic

```
Public Shared Function NETERR ( _  
    ByRef Istexc As String, _  
    ByRef Istrtn As String, _  
    ByRef Isterr As Integer _  
) As Integer
```

### [-] 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](mailto: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

Visual Basic

```
Public Shared Function NETFLS ( _  
    netid As Integer, _  
    code As Integer _  
) As Integer
```

### 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](mailto: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

Visual Basic

```
Public Shared Function NETOPN ( _  
    netid As Integer, _  
    rname As String, _  
    rport As Integer, _  
    rtime As Integer _  
) As Integer
```

### 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](mailto: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

Visual Basic

```
Public Shared Function NETREC ( _  
    netid As Integer, _  
    code As Integer, _  
    timeout As Integer, _  
    outbuf As Byte(), _  
    limit As Integer, _  
    ByRef nbytes As Integer _
```

) As Integer  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](mailto: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

Visual Basic

```
Public Shared Function NETSND ( _  
    netid As Integer, _  
    code As Integer, _  
    outbuf As Byte(), _  
    nbytes As Integer _  
) As Integer
```

### 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](mailto: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

Visual Basic

```
Public Shared Function NETSRV ( _  
    code As Integer, _  
    lport As Integer, _  
    rlen As Integer, _  
    slen As Integer _  
) As Integer
```

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](mailto: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

Visual Basic

```
Public Shared Function NETSTS ( _  
    netid As Integer, _  
    ByRef lname As String, _  
    ByRef laddr As String, _  
    ByRef rname As String, _  
    ByRef raddr As String, _  
    ByRef nrmsg As Integer, _  
    ByRef nsmsg As Integer, _  
    ByRef nrcnt As Integer, _  
    ByRef nscnt As Integer, _  
    ByRef nrerr As Integer, _  
    ByRef nserr As Integer, _  
    ByRef rerr As Integer, _  
    ByRef serr As Integer _  
)
```

As Integer

### 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](mailto: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

Visual Basic

```
Public Shared Function NETSUS ( _  
    tmeout As Integer _  
) As Integer
```

### 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](mailto:support@microglyph.com)

(c) 2007 MicroGlyph Systems. All rights reserved.

## NETTMR Method

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

Routine to get current time in milliseconds.

### **Syntax**

Visual Basic

**Public Shared Function NETTMR As Integer**  **Return Value**

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

Assembly: SCISNET (Module: SCISNET)

Send comments on this topic to [support@microglyph.com](mailto: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 test .NET 2005 TCP/IP Sockets support
'*
*****

Imports System
Imports SciSnet

Public Class TNET

'   Instantiate SciSnet Class

    Public Shared Dim SNClass As SciSnet.SN = New SciSnet.SN()

'   Main Program

    Public Shared Sub Main ()

'   Local variables

        Dim status,i,j,NetID,LPort,RPort,RTime,Rlen,Slen,Code,Nrmsg,
            Nsmg,Nrcnt,Nscnt,Nrerr,Nserr,Rerr,Serr,Lsterr,ival,NBytes,
            Func,Limit,Value,j1,j2,n As Integer
        Dim IntBuf(15) As Integer
        Dim RName,Rnme,Raddr,Lname,Laddr,Lstexc,Lstrtn,reply,line,
            pline,bc,oc,cc,qc,sc,rc,wc,fc,mc,nc As String
        Dim dialog,SFlag As Boolean

'   Initial values

        Limit   = 80
        bc      = " "
        oc      = "O"
        cc      = "C"
        qc      = "Q"
        sc      = "S"
        rc      = "R"
        wc      = "W"
        fc      = "F"
        mc      = "M"
        nc      = "N"
        Lstexc  = ""
        Lstrtn  = ""
        Rnme    = ""
        Lname   = ""
        Laddr   = ""
        Raddr   = ""

'   Start Main

        Try

'   Dialog processing loop

            dialog = true
            SFlag  = false
            While (dialog)

```

```

' Main Function Message

Console.WriteLine(vbNewLine + " TNET Functions:" + vbNewLine)
Console.WriteLine("    N = Server      (NETSRV)")
Console.WriteLine("    O = Open      (NETOPN)")
Console.WriteLine("    C = Close     (NETCLS)")
Console.WriteLine("    S = Status    (NETSTS)")
Console.WriteLine("    R = Read      (NETREC)")
Console.WriteLine("    W = Write     (NETSND)")
Console.WriteLine("    F = Flush     (NETFLS)")
Console.WriteLine("    M = Display Menu")
Console.WriteLine("    Q = Quit/Exit")
Console.Write(vbNewLine + " Enter Function: ")
'
Get function code
line = Console.ReadLine().ToUpper()
If (line.Length = 0) Then
'    Nothing entered, loop
    Continue While
End If
reply = line.Substring(0,1)

' Process reply

Select Case (reply)

    Case bc

'        Blank line

    Case mc

'        Menu command

    Case nc

'        Network server

        Console.WriteLine(vbNewLine +
                           "    Network Server Functions:" +
                           vbNewLine)
        Console.WriteLine("        1 = Start Network Server")
        Console.WriteLine("        2 = Resume Network Server")
        Console.WriteLine("        3 = Stop Network Server")
        If (SFlag)
'            Then
            Network server is started
            Code = 3
            Console.Write(vbNewLine + "            Enter Function [3]: ")
            SFlag = false
        Else
'            Network server is down
            Code = 1
            Console.Write(vbNewLine +
                           "            Enter Function
                           [1]: ")
            SFlag = true
        End If
        line = Console.ReadLine()
        If (line.Length > 0) Then
            Int32.TryParse(line,ival)
            Set select code value

```

```

        Code = ival
    End If
    Start Network Server parameters
    If (Code = 1) Then
        Set local listen port
        LPort = 8001
        Console.Write(
            "        Enter Local Listen Port    [8001]: ")
        line = Console.ReadLine()
        If (line.Length > 0) Then
            Int32.TryParse(line,ival)
            Set selected local port value
            LPort = ival
        End If
        Set TCP/IP receive buffer length
        Rlen = 1440
        Console.Write(
            "        Enter Receive Buffer Length[1440]: ")
        line = Console.ReadLine()
        If (line.Length > 0) Then
            Int32.TryParse(line,ival)
            Set selected receive buffer length
            Rlen = ival
        End If
        Set TCP/IP send buffer length
        Slen = 1440
        Console.Write(
            "        Enter Send Buffer Length    [1440]: ")
        line = Console.ReadLine()
        If (line.Length > 0) Then
            Int32.TryParse(line,ival)
            Set selected send buffer length
            Slen = ival
        End If
    End If
    Call NETSRV routine
    status = SciSnet.SN.NETSRV(Code,LPort,Rlen,Slen)
    If (status <> 0) Then
        Console.WriteLine(vbNewLine +
            " NETSRV error: ({0})" + vbNewLine,
            status)
        Console.WriteLine("        Code    = {0}",Code)
        Console.WriteLine("        LPort   = {0}",LPort)
        Console.WriteLine("        Rlen    = {0}",Rlen)
        Console.WriteLine("        Slen    = {0}",Slen)
    End If

Case oc

    Open network connection

    Get NetID
    NetID = 1
    Console.Write(vbNewLine +
        "        Enter Network Connection ID    [1]: ")
    line = Console.ReadLine()
    If (line.Length > 0) Then
        Int32.TryParse(line,ival)
        Set NetID value
        NetID = ival
    End If

```

```

End If
'
Get remote host name
Console.Write(
    "        Enter Remote Host Name                : ")
line = Console.ReadLine()
If (line.Length > 0)
    Then
'
        Set remote host name
        RName = line
    Else
'
        Nothing entered
        Continue While
    End If
'
Get remote port
RPort = 8001
Console.Write(
    "        Enter Remote Host Receive Port        [8001]: ")
line = Console.ReadLine()
If (line.Length > 0) Then
    Int32.TryParse(line,ival)
'
    Set selected remote port value
    RPort = ival
End If
'
Get remote host connect timeout value
RTime = 0
Console.Write(
    "        Enter Remote Host Connect Timeout Value[0]: ")
line = Console.ReadLine()
If (line.Length > 0) Then
    Int32.TryParse(line,ival)
'
    Set remote host connect timeout value
    RTime = ival
End If
'
Open connection to remote host
status = SciSnet.SN.NETOPN(NetID,RName,RPort,RTime)
If (status <> 0) Then
    Console.WriteLine(vbNewLine +
        " NETOPN error: ({0})" + vbNewLine,status)
    Console.WriteLine("        NetID = {0}",NetID)
    Console.WriteLine("        RName = {0}",RName)
    Console.WriteLine("        RPort = {0}",RPort)
    Console.WriteLine("        RTime = {0}",RTime)
End If

Case cc

'
    Close network connection

'
    Get function code
    Console.WriteLine(vbNewLine +
        "        Close Functions:" + vbNewLine)
    Console.WriteLine(
        "            1 = Purge input and output messages")
    Console.WriteLine(
        "            2 = Purge input,wait for output messages")
    Console.Write(vbNewLine +
        "        Enter Function                [1]: ")
    Code = 1
    line = Console.ReadLine()
    If (line.Length > 0) Then

```

```

        Int32.TryParse(line,ival)
    '
    ' Set select code value
    ' Code = ival
    End If
    '
    ' Get NetID
    ' NetID = 1
    ' Console.WriteLine("        Enter Network Connection ID[1]: ")
    ' line = Console.ReadLine()
    ' If (line.Length > 0) Then
    '     Int32.TryParse(line,ival)
    '     Set NetID value
    '     NetID = ival
    ' End If
    '
    ' Close network connection
    ' status = SciSnet.SN.NETCLS(NetID,Code)
    ' If (status <> 0) Then
    '     Console.WriteLine(vbNewLine +
    '         " NETCLS error: ({0})" + vbNewLine,status)
    '     Console.WriteLine("        NetID = {0}",NetID)
    '     Console.WriteLine("        Code = {0}",Code)
    ' End If

Case sc

    '
    ' Network connection status
    '
    '
    ' Get NetID
    ' NetID = 1
    ' Console.WriteLine(vbNewLine +
    '     "        Enter Network Connection ID[1]: ")
    ' line = Console.ReadLine()
    ' If (line.Length > 0) Then
    '     Int32.TryParse(line,ival)
    '     Set NetID value
    '     NetID = ival
    ' End If
    '
    ' Get status for this network connection
    ' status = SciSnet.SN.NETSTS(NetID,Lname,Laddr,Rnme,Raddr,
    '                             Nrmsg,Nsmg,Nrcnt,Nscnt,Nrerr,
    '                             Nserr,Rerr,Serr)
    '
    ' If (status = 0)
    ' Then
    '     Console.WriteLine(vbNewLine +
    '         "        NetID = {0}",NetID)
    '     Console.WriteLine("        Lname = {0}",Lname)
    '     Console.WriteLine("        Laddr = {0}",Laddr)
    '     Console.WriteLine("        Rname = {0}",Rnme)
    '     Console.WriteLine("        Raddr = {0}",Raddr)
    '     Console.WriteLine("        Nrmsg = {0}",Nrmsg)
    '     Console.WriteLine("        Nsmg = {0}",Nsmg)
    '     Console.WriteLine("        Nrcnt = {0}",Nrcnt)
    '     Console.WriteLine("        Nscnt = {0}",Nscnt)
    '     Console.WriteLine("        Nrerr = {0}",Nrerr)
    '     Console.WriteLine("        Nserr = {0}",Nserr)
    '     Console.WriteLine("        Rerr = {0}",Rerr)
    '     Console.WriteLine("        Serr = {0}",Serr)
    ' Else
    '     Console.WriteLine(vbNewLine +
    '         " NETSTS error: ({0})" + vbNewLine,status)
    '     Console.WriteLine("        NetID = {0}",NetID)

```

```

        Console.WriteLine("      Lname = {0}",Lname)
        Console.WriteLine("      Laddr = {0}",Laddr)
        Console.WriteLine("      Rname = {0}",Rnme)
        Console.WriteLine("      Raddr = {0}",Raddr)
        Console.WriteLine("      Nrmsg = {0}",Nrmsg)
        Console.WriteLine("      Nsmg = {0}",Nsmsg)
        Console.WriteLine("      Nrcnt = {0}",Nrcnt)
        Console.WriteLine("      Nscnt = {0}",Nscnt)
        Console.WriteLine("      Nrerr = {0}",Nrerr)
        Console.WriteLine("      Nserr = {0}",Nserr)
        Console.WriteLine("      Rerr = {0}",Rerr)
        Console.WriteLine("      Serr = {0}",Serr)
    End If

```

Case rc

```

'      Read bytes from receive buffer
'
'      Get function code
        Console.WriteLine(vbNewLine +
            "      Receive Functions:" + vbNewLine)
        Console.WriteLine("          1 = Normal Read")
        Console.WriteLine("          2 = Read with Timeout")
        Console.Write(vbNewLine +
            "          Enter Function          [1]: ")
        Code = 1
        line = Console.ReadLine()
        If (line.Length > 0) Then
            Int32.TryParse(line,ival)
            Set select code value
            Code = ival
        End If
'      Get timeout value
        Value = 0
        If (Code = 2) Then
            Console.Write(vbNewLine +
                "          Enter Read Timeout Value: ")
            line = Console.ReadLine()
            If (line.Length > 0) Then
                Int32.TryParse(line,ival)
                Set read timeout value
                Value = ival
            End If
        End If
'      Get NetID
        NetID = 1
        Console.Write("          Enter Network Connection ID[1]: ")
        line = Console.ReadLine()
        If (line.Length > 0) Then
            Int32.TryParse(line,ival)
            Set NetID value
            NetID = ival
        End If
'      Read a message from a remote host
        Dim TmpBuf(Limit-1) As Byte
        status = SciSnet.SN.NETREC(NetID,Code,Value,TmpBuf,Limit,
            NBytes)
        If (status = 0)
            Then
                If (NBytes = 0) Then

```

```

        Console.WriteLine(vbNewLine +
            " NETREC: No Data Available" + vbNewLine)
        Console.WriteLine("      NetID = {0}",NetID)
        Console.WriteLine("      Code  = {0}",Code)
    End If
    If (NBytes > 0) Then
        Dim BytBuf(NBytes-1) As Byte
        Move number of bytes read into byte buffer
        For i = 0 To (NBytes-1)
            BytBuf(i) = TmpBuf(i)
        Next i
        Console.WriteLine(vbNewLine +
            "      Count: {0}",NBytes)
        Convert bytes received to ASCII string
        Func      = 5
        line      = ""
        status = SciSnet.SN.NETENC(Func,BytBuf,IntBuf,line)
        Display ASCII string
        For i = 0 To (NBytes-1) Step 32
            n = Math.Min(32,NBytes-i)
            pline = line.Substring(i,n)
            If (i = 0)
                Then
                    Console.WriteLine(vbNewLine +
                        "      ASCII: {0}",pline)
                Else
                    Console.WriteLine(
                        "      : {0}",pline)
                End If
            Next i
            Display Hex string
            For i = 0 To (NBytes-1) Step 11
                j1 = i
                j2 = i+Math.Min(11,NBytes-i)
                If (i = 0)
                    Then
                        Console.Write("      Hex : ")
                        For j = j1 To (j2-1)
                            Console.Write("{0,2:X2} ",BytBuf(j))
                        Next j
                        Console.WriteLine(" ")
                    Else
                        Console.Write("      : ")
                        For j = j1 To (j2-1)
                            Console.Write("{0,2:X2} ",BytBuf(j))
                        Next j
                        Console.WriteLine(" ")
                    End If
                Next i
            End If
        Else
            Console.WriteLine(vbNewLine + " NETREC error: ({0})" +
                vbNewLine,status)
            Console.WriteLine("      NetID = {0}",NetID)
            Console.WriteLine("      Code  = {0}",Code)
            Console.WriteLine("      Value = {0}",Value)
        End If
    Case wc

```

```

' Write out bytes to send buffer
'
' Get function code
Console.WriteLine(vbNewLine + " Send Functions:" +
vbNewLine)
Console.WriteLine(" 1 = Send Message")
Console.WriteLine(" 2 = Wait on Message to be sent")
Console.WriteLine(" 3 = Timestamp Message")
Console.WriteLine(" 4 = Wait + Timestamp")
Console.WriteLine(vbNewLine +
" Enter Function [1]: ")
Code = 1
line = Console.ReadLine()
If (line.Length > 0) Then
    Int32.TryParse(line,ival)
    Set select code value
    Code = ival
End If
'
' Get NetID
NetID = 1
Console.WriteLine(" Enter Network Connection ID[1]: ")
line = Console.ReadLine()
If (line.Length > 0) Then
    Int32.TryParse(line,ival)
    Set NetID value
    NetID = ival
End If
'
' Get output line
Console.WriteLine(" Enter Output String : ")
line = Console.ReadLine()
If (line.Length = 0) Then
    Nothing to send, loop
    Continue While
End If
'
' Output ASCII string on network connection
Func = 3
NBytes = line.Length
Dim BytBuf(NBytes-1) As Byte
status = SciSnet.SN.NETENC(Func,BytBuf,IntBuf,line)
'
' Send ASCII bytes
status = SciSnet.SN.NETSND(NetID,Code,BytBuf,NBytes)
If (status <> 0) Then
    Console.WriteLine(vbNewLine + " NETSND error: ({0})" +
vbNewLine,status)
    Console.WriteLine(" NetID = {0}",NetID)
    Console.WriteLine(" Code = {0}",Code)
    Console.WriteLine(" line = {0}",line)
    Console.WriteLine(" NBytes = {0}",NBytes)
End If
'
Case fc
'
' Flush receive and send buffers
'
' Get function code
Console.WriteLine(vbNewLine + " Flush Functions:" +
vbNewLine)
Console.WriteLine(" 1 = Flush Receive Messages")
Console.WriteLine(" 2 = Flush Send Messages")
Console.WriteLine(" 3 = Flush Receive + Send Messages")

```

```

Console.WriteLine(vbNewLine +
    "          Enter Function                               [1]: ")
Code = 1
line = Console.ReadLine()
If (line.Length > 0) Then
    Int32.TryParse(line,ival)
    Set select code value
    Code = ival
End If
Get NetID
NetID = 1
Console.WriteLine("          Enter Network Connection ID[1]: ")
line = Console.ReadLine()
If (line.Length > 0) Then
    Int32.TryParse(line,ival)
    Set NetID value
    NetID = ival
End If
Flush buffers
status = SciSnet.SN.NETFLS(NetID,Code)
If (status <> 0) Then
    Console.WriteLine(vbNewLine + " NETFLS error: ({0})" +
        vbNewLine,status)
    Console.WriteLine("          NetID = {0}",NetID)
    Console.WriteLine("          Code = {0}",Code)
End If

Case qc

    Exit TNET program

    dialog = false

Case Else

    Unknown command

End Select

Dialog process loop

End While

Report any errors

status = SciSnet.SN.NETERR(Lstexc,Lstrtn,Lsterr)
If (Lsterr = 0)
    Then
        Report normal completion
        Console.WriteLine(vbNewLine + "    Normal completion" +
            vbNewLine)
    Else
        Report Last Errors
        Console.WriteLine(vbNewLine +
            "          Last Exception           : {0}",Lstexc)
        Console.WriteLine(" Last Member in Error       : {0}",Lstrtn)
        Console.WriteLine(" Last Member Error Code    : {0}",Lsterr)
    End If

Catch e As FormatException

```

```
'      Report exception in Main Program
      Console.WriteLine(" Exception {0} ",e)
      return

Catch e As Exception
'      Report exception in Main Program
      Console.WriteLine(" Exception {0} ",e)
      return

End Try

'      Exit application

End Sub 'Main

End Class
```