

## 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

C#

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

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
	<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

C#

```
public SN () Assembly: 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

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](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

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](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

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](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

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](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**

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](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

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](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

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](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

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](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

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](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

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](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

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](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

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](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 Sockets support
#*
*****

#   System definistions and imports

import clr
import System
import IronPython

clr.AddReferenceByPartialName("System")
clr.AddReferenceByPartialName("IronPython")
clr.AddReferenceToFile("SciSnet.dll")

from System          import Boolean,FormatException,Exception,Int32,Byte
from System          import String
from System.Array    import CreateInstance
from System.Console import WriteLine,ReadLine,Write
from System          import *
from SciSnet         import *

#   Main program

def Main():

    try:

#       Local variables definitions

        NetID      = clr.Reference[Int32](0)
        LPort      = clr.Reference[Int32](0)
        RPort      = clr.Reference[Int32](0)
        RTime      = clr.Reference[Int32](0)
        Rlen       = clr.Reference[Int32](0)
        Slen       = clr.Reference[Int32](0)
        Nrmsg      = clr.Reference[Int32](0)
        Nsmg       = clr.Reference[Int32](0)
        Nrcnt      = clr.Reference[Int32](0)
        Nscnt      = clr.Reference[Int32](0)
        Nrerr      = clr.Reference[Int32](0)
        Nserr      = clr.Reference[Int32](0)
        Rerr       = clr.Reference[Int32](0)
        Serr       = clr.Reference[Int32](0)
        Lsterr     = clr.Reference[Int32](0)
        ival       = clr.Reference[Int32](0)
        NBytes     = clr.Reference[Int32](0)
        Limit      = clr.Reference[Int32](0)
        Timeout    = clr.Reference[Int32](0)
        dialog     = System.Boolean
        SFlag      = System.Boolean
        RName      = clr.Reference[String]("")
        Rname      = clr.Reference[String]("")
        Raddr      = clr.Reference[String]("")
        Lname      = clr.Reference[String]("")
        Laddr      = clr.Reference[String]("")
        Lstexc     = clr.Reference[String]("")
        Lstrtn     = clr.Reference[String]("")

```

```

SValue   = clr.Reference[String]("")
TmpBuf   = CreateInstance(Byte,1)
BytBuf   = CreateInstance(Byte,1)
IntBuf   = CreateInstance(Int32,16)

#   SciSnet Class must be instantiated

SN()

#   Set initial values

bc       = " "
oc       = "O"
cc       = "C"
qc       = "Q"
sc       = "S"
rc       = "R"
wc       = "W"
fc       = "F"
mc       = "M"
nc       = "N"
reply    = ""
line     = ""
pline    = ""
j1       = 0
j2       = 0
n        = 0
i        = 0
j        = 0
Code     = 0
Func     = 0
status   = 0

NetID.Value   = 0
LPort.Value   = 0
RPort.Value   = 0
RTime.Value   = 0
Rlen.Value    = 0
Slen.Value    = 0
Nrmsg.Value   = 0
Nsmsg.Value   = 0
Nrcnt.Value   = 0
Nscnt.Value   = 0
Nrerr.Value   = 0
Nserr.Value   = 0
Rerr.Value    = 0
Serr.Value    = 0
Lsterr.Value  = 0
ival.Value    = 0
NBytes.Value  = 0
Limit.Value   = 80
Timeout.Value = 0

#   Start Main

#   Dialog loop

dialog = True
SFlag = False
while dialog:

```

```

#      Function Request Message

Console.WriteLine("\n TNET Functions:\n")
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("\n Enter Function: ")

#      Get function code

line = Console.ReadLine().ToUpper()
if line.Length == 0:
#      Nothing entered. loop
    continue
reply = line.Substring(0,1)

#      Process reply

if reply == bc :

#      Blank character
    continue

if reply == mc :

#      Menu command

    continue

if reply == nc :

#      Network server

    Console.WriteLine("\n      Network Server Functions:\n")
    Console.WriteLine("        1 = Start Network Server")
    Console.WriteLine("        2 = Resume Network Server")
    Console.WriteLine("        3 = Stop Network Server")

    if SFlag == True :
#      Network server is started
        Code = 3
        Console.Write("\n      Enter Function [3]: ")
        SFlag = False
    else :
#      Network server is down
        Code = 1
        Console.Write("\n      Enter Function [1]: ")
        SFlag = True
    line = Console.ReadLine()
    if line.Length > 0 :
        Int32.TryParse(line,ival)
#      Set select code value
        Code = ival.Value

```

```

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

    continue

if reply == oc :

# Open network connection

# Get NetID
NetID.Value = 1
Console.Write("\n Enter Network Connection " +
"ID [1]: ")
line = Console.ReadLine()
if line.Length > 0:
    Int32.TryParse(line,ival)
# Set select port value
NetID.Value = ival.Value
Console.Write(" Enter Remote Host " +
"Name : ")
line = Console.ReadLine()
if line.Length > 0:
    RName.Value = line
else:
# Nothing entered
continue
# Get remote port
RPort.Value = 8001

```

```

Console.Write("    Enter Remote Host " +
              "Receive Port          [8001]: ")
line = Console.ReadLine()
if line.Length > 0:
    Int32.TryParse(line,ival)
#    Set select remote port value

    RPort.Value = ival.Value
#    Get remote host connection timeout value
RTime.Value = 0
Console.Write("    Enter Remote Host " +
              "Connection Timeout Value  [0]: ")
line = Console.ReadLine()
if line.Length > 0:
    Int32.TryParse(line,ival)
#    Set select remote host connection timeout value

    RTime.Value = ival.Value
#    Open connection to remote host
status = SN.NETOPN(NetID.Value,RName.Value,RPort.Value,
                  RTime.Value)

if status != 0:
    Console.WriteLine("\n        NETOPN Errors: \n")
    Console.WriteLine("        Status = {0}",status)
    Console.WriteLine("        NetID   = {0}",NetID.Value)
    Console.WriteLine("        RName  = {0}",RName.Value)
    Console.WriteLine("        RPort  = {0}",RPort.Value)
    Console.WriteLine("        RTime  = {0}",RTime.Value)
    continue

if reply == cc :

#    Close serial port

    Console.WriteLine("\n        Close Functions:\n")
    Console.WriteLine("        1 = Purge input and" +
                    " output messages")
    Console.WriteLine("        2 = Purge input,wait " +
                    "for output messages")
    Console.Write("\n        Enter Function[1]: ")
    Code = 1
    line = Console.ReadLine()
    if line.Length > 0:
        Int32.TryParse(line,ival)
#        Set select code value
        Code = ival.Value
#
#    Get NetID
NetID.Value = 1
Console.Write("\n        Enter Network Connection ID  [1]: ")
line = Console.ReadLine()
if line.Length > 0:
    Int32.TryParse(line,ival)
#    Set select port value
NetID.Value = ival.Value
status = SN.NETCLS(NetID.Value,Code)
if status != 0:
    Console.WriteLine("\n        NETCLS Errors: \n")
    Console.WriteLine("        Status = {0}",status)
    Console.WriteLine("        NetID   = {0}",NetID.Value)
    Console.WriteLine("        Code    = {0}",Code)

```

```

        continue

    if reply == sc :

#       Network connection status

#       Get NetID
        NetID.Value = 1
        Console.WriteLine("\n      Enter Network Connection ID [1]: ")
        line = Console.ReadLine()
        if line.Length > 0:
            Int32.TryParse(line,ival)
#           Set select port value
            NetID.Value = ival.Value
        status = SN.NETSTS(NetID.Value,Lname,Laddr,Rname,Raddr,
                           Nrmsg,Nsmsg,Nrcnt,Nscnt,Nrerr,Nserr,
                           Rerr,Serr)

        if status == 0:
            Console.WriteLine("\n          NetID = {0}",NetID.Value)
            Console.WriteLine("          Lname = {0}",Lname.Value)
            Console.WriteLine("          Laddr = {0}",Laddr.Value)
            Console.WriteLine("          Rname = {0}",Rname.Value)
            Console.WriteLine("          Raddr = {0}",Raddr.Value)
            Console.WriteLine("          Nrmsg = {0}",Nrmsg.Value)
            Console.WriteLine("          Nsmsg = {0}",Nsmsg.Value)
            Console.WriteLine("          Nrcnt = {0}",Nrcnt.Value)
            Console.WriteLine("          Nscnt = {0}",Nscnt.Value)
            Console.WriteLine("          Nrerr = {0}",Nrerr.Value)
            Console.WriteLine("          Nserr = {0}",Nserr.Value)
            Console.WriteLine("          Rerr = {0}",Rerr.Value)
            Console.WriteLine("          Serr = {0}",Serr.Value)
        else:
            Console.WriteLine("\n          Status = {0}",status)
            Console.WriteLine("          NetID = {0}",NetID.Value)
            Console.WriteLine("          Lname = {0}",Lname.Value)
            Console.WriteLine("          Laddr = {0}",Laddr.Value)
            Console.WriteLine("          Rname = {0}",Rname.Value)
            Console.WriteLine("          Raddr = {0}",Raddr.Value)
            Console.WriteLine("          Nrmsg = {0}",Nrmsg.Value)
            Console.WriteLine("          Nsmsg = {0}",Nsmsg.Value)
            Console.WriteLine("          Nrcnt = {0}",Nrcnt.Value)
            Console.WriteLine("          Nscnt = {0}",Nscnt.Value)
            Console.WriteLine("          Nrerr = {0}",Nrerr.Value)
            Console.WriteLine("          Nserr = {0}",Nserr.Value)
            Console.WriteLine("          Rerr = {0}",Rerr.Value)
            Console.WriteLine("          Serr = {0}",Serr.Value)

        continue

    if reply == rc :

#       Read bytes from receive buffer

        Console.WriteLine("\n      Receive Functions:\n")
        Console.WriteLine("          1 = Normal Read")
        Console.WriteLine("          2 = Read with Timeout")
        Console.WriteLine("\n      Enter Function[1]: ")
        Code = 1
        line = Console.ReadLine()
        if line.Length > 0:

```

```

        Int32.TryParse(line,ival)
#       Set select code value
        Code = ival.Value
Timeout.Value = 0
if Code == 1 :
#       Get NetID
        NetID.Value = 1
        Console.WriteLine("\n          Enter Network Connection ID [1]: ")
        line = Console.ReadLine()
        if line.Length > 0:
            Int32.TryParse(line,ival)
#           Set select port value
            NetID.Value = ival.Value
if Code == 2 :
#       Get NetID
        NetID.Value = 1
        Console.WriteLine("\n          Enter Network Connection ID [1]: ")
        line = Console.ReadLine()
        if line.Length > 0:
            Int32.TryParse(line,ival)
#           Set select port value
            NetID.Value = ival.Value
#       Get Timeout value
        Console.WriteLine("          Enter Read Timeout Value          : ")
        line = Console.ReadLine()
        if line.Length > 0:
            Int32.TryParse(line,ival)
#           Set read timeout value
            Timeout.Value = ival.Value
TmpBuf = CreateInstance(Byte,Limit.Value)
status = SN.NETREC(NetID.Value,Code,Timeout.Value,
                  TmpBuf,Limit.Value,NBytes)
if status == 0:
#       Display bytes received
        if NBytes.Value == 0:
            Console.WriteLine("\n          No Data Available")
            Console.WriteLine("          NetID = {0}",NetID.Value)
            Console.WriteLine("          Code = {0}",Code)
        if NBytes.Value > 0:
            BytBuf = CreateInstance(Byte,NBytes.Value)
#           Move number of bytes read into Buffer
            for i in range(NBytes.Value):
                BytBuf[i] = TmpBuf[i]
            Console.WriteLine("\n          Count: {0}",NBytes.Value)
#           Convert bytes read to ASCII string
            Code = 5
            status = SN.NETENC(Code,BytBuf,IntBuf,SValue)
#           Display ASCII string
            for i in range(0,NBytes.Value,32):
                n = Math.Min(32,NBytes.Value-i)
                pline = SValue.Value.Substring(i,n)
                if i == 0:
                    Console.WriteLine("          ASCII: {0}",pline)
                else:
                    Console.WriteLine("          : {0}",pline)
#           Display Hex string
            for i in range(0,NBytes.Value,11):
                j1 = i
                j2 = i+Math.Min(11,NBytes.Value-i)
                if i == 0:

```

```

        Console.WriteLine("          Hex : ")
        for j in range(j2) :
            Console.WriteLine("{0,2:X2} ",BytBuf[j])
        Console.WriteLine(" ")
    else:
        Console.WriteLine("          : ")
        for j in range(j2) :
            Console.WriteLine("{0,2:X2} ",BytBuf[j])
        Console.WriteLine(" ")
else:
    Console.WriteLine("\n          NETREC Errors: \n")
    Console.WriteLine("          Status = {0}",status)
    Console.WriteLine("          NetID = {0}",NetID.Value)
    Console.WriteLine("          Code = {0}",Code)
    Console.WriteLine("          Timeout = {0}",Timeout.Value)
continue

if reply == wc :

#       Write out bytes to send buffer

Console.WriteLine("\n          Write Functions:\n")
Console.WriteLine("          1 = Send Message")
Console.WriteLine("          2 = Wait on Message to be sent")
Console.WriteLine("          3 = Timestamp Message")
Console.WriteLine("          4 = Wait + TimeTimestamp")
Console.Write("\n          Enter Function[1]: ")
Code = 1
line = Console.ReadLine()
if line.Length > 0:
    Int32.TryParse(line,ival)
#       Set select Code value
    Code = ival.Value
#
#       Get NetID
NetID.Value = 1
Console.Write("\n          Enter Network Connection ID [1]: ")
line = Console.ReadLine()
if line.Length > 0:
    Int32.TryParse(line,ival)
#       Set select port value
    NetID.Value = ival.Value
Console.Write("          Enter Output String          : ")
line = Console.ReadLine()
if line.Length == 0:
#       Nothing to send, loop
    continue
#
#       Convert string to ASCII bytes
Func          = 3
NBytes.Value = line.Length
BytBuf = CreateInstance(Byte,NBytes.Value)
SValue.Value = line
status = SN.NETENC(Func,BytBuf,IntBuf,SValue)
#       Send the ASCII bytes
status = SN.NETSND(NetID.Value,Code,BytBuf,NBytes.Value)
if status != 0:
    Console.WriteLine("\n          NETSND Errors: \n")
    Console.WriteLine("          Status = {0}",status)
    Console.WriteLine("          NetID = {0}",NetID.Value)
    Console.WriteLine("          Code = {0}",Code)
    Console.WriteLine("          line = {0}",SValue.Value)

```

```

        Console.WriteLine("                NBytes = {0}",NBytes.Value)

if reply == fc :

#       Flush receive and send buffers

        Console.WriteLine("\n        Flush Functions:\n")
        Console.WriteLine("                1 = Flush Receive Messages")
        Console.WriteLine("                2 = Flush Send Messages")
        Console.WriteLine("                3 = Flush Receive/Send Messages")
        Console.Write("\n        Enter Function[1]: ")
        Code = 1
        line = Console.ReadLine()
        if line.Length > 0:
            Int32.TryParse(line,ival)
#           Set select Code value
            Code = ival.Value
#       Get NetID
        NetID.Value = 1
        Console.Write("\n                Enter Network Connection ID  [1]: ")
        line = Console.ReadLine()
        if line.Length > 0:
            Int32.TryParse(line,ival)
#           Set select port value
            NetID.Value = ival.Value
        status = SN.NETFLS(NetID.Value,Code)
        if status <> 0:
            Console.WriteLine("\n                NETFLS Errors: \n")
            Console.WriteLine("                Status = {0}",status)
            Console.WriteLine("                NetID  = {0}",NetID.Value)
            Console.WriteLine("                Code   = {0}",Code)

        continue

if reply == qc :

#       Exit TNET program

        dialog = False

        continue

#       Dialog process loop

#       Report any errors

        status = SN.NETERR(Lstexc,Lstrtn,Lsterr)
        if Lsterr.Value == 0:
#           Report normal completion
            Console.WriteLine("\n        Normal completion\n")
        else:
#           Report Last Errors
            Console.WriteLine("\n        Last Exception                : {0}",
                Lstexc.Value)
            Console.WriteLine("        Last Member in Error        : {0}",
                Lstrtn.Value)
            Console.WriteLine("        Last Member Error Code     : {0}",
                Lsterr.Value)
            Console.WriteLine(" ")

```

```
    except FormatException, e:
#       Report exception in Main Program
        Console.WriteLine(" Exception {0} ",e)

    except Exception, e:
#       Report exception in Main Program
        Console.WriteLine(" Exception {0} ",e)

#   Exit application

Main()
```