

Namespaces

Namespaces

Namespaces

Namespace	Description
SciComm	

Send comments on this topic to support@microglyph.com

(c) 2007 MicroGlyph Systems. All rights reserved.

SciComm Namespace

Namespaces > SciComm

Syntax

C#

namespace SciComm Types

All Types

Classes

Structures


Interfaces

Enumerations

Icon

Type

Description

	SC	SC Class - SciComm Serial Communications Interface Methods
---	-----------	--

Send comments on this topic to support@microglyph.com

(c) 2007 MicroGlyph Systems. All rights reserved.

SC Class

Namespaces > **SciComm** > **SC**

SC Class - SciComm Serial Communications Interface Methods

Syntax

C#

`public class SC`

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
	SC()	SC Class - constructor
	COMCLS(Int32, Int32)	Routine to close serial port.
	COMCTL(Int32, Int32)	Routine to set control lines.
	COMDTM(Int32, Int32, Int32, Int32, Int32, Int32)	Routine to get date and time.
	COMENC(Int32, Byte[], Int32[], String)	Routine to encode bytes, integers, or strings
	COMERR(String, String, Int32)	Routine to get error messages and error code.
	COMFLS(Int32, Int32)	Routine to flush serial port buffers.
	COMOPN(Int32, Int32, Int32, Int32, Int32, Int32, Int32)	Routine to open serial port.
	COMREC(Int32, Int32, Int32, Int32, Byte[], Byte[], Int32)	Routine to read bytes from serial port buffer.
	COMSND(Int32, Int32, Int32, Int32, Byte[])	Routine to write bytes to serial port buffer.
	COMSTS(Int32, Int32, Int32, Int32, Int32, Int32, Int32, Int32, Int32, Int32)	Routine to get serial port status.
	COMSUS(Int32)	Routine to sleep the executing thread.
	COMTMR()	Routine to get current time in milliseconds.

Inheritance

Hierarchy

Object

└─ SC

Assembly: SCICOMM (Module: SCICOMM)

Send comments on this topic to support@microglyph.com

(c) 2007 MicroGlyph Systems. All rights reserved.

SC Constructor

[Namespaces](#) > [SciComm](#) > [SC](#) > [SC\(\)](#)



SC Class - constructor

Syntax

C#

```
public SC () Assembly: SCICOMM (Module: SCICOMM)
```

Send comments on this topic to support@microglyph.com

(c) 2007 MicroGlyph Systems. All rights reserved.

COMCLS Method (code, port)

Namespaces > [SciComm](#) > [SC](#) > [COMCLS\(Int32, Int32\)](#)

Routine to close serial port.

Syntax

C#

```
public static int COMCLS (  
    int code,  
    int port  
)
```

Parameters

code (**Int32**)

Function code

- 1 - Purge receive/send buffers
- 2 - Purge receive,wait on send buffers

port (**Int32**)

Serial port ID

- 1 - COM1
- 2 - COM2
- 3 - COM3
- 4 - COM4
- 5 - COM5
- 6 - COM6
- 7 - COM7
- 8 - COM8
- 9 - COM9
- 10 - COM10
- 11 - COM11
- 12 - COM12

Return Value

Status is returned as an int value.

- 0 - Normal return
- 1 - Invalid argument
- 2 - System exception
- 3 - SC Class not instantiated
- 4 - Port is closed
- 5 - Send thread failed to stop

Assembly: SCICOMM (Module: SCICOMM)

Send comments on this topic to support@microglyph.com

(c) 2007 MicroGlyph Systems. All rights reserved.

COMCTL Method (code, port)

Namespaces > [SciComm](#) > [SC](#) > [COMCTL\(Int32, Int32\)](#)

Routine to set control lines.

Syntax

C#

```
public static int COMCTL (  
    int code,  
    int port  
)
```

Parameters

code (**Int32**)

Function code

- 1 - Raise data-terminal-ready signal
- 2 - Raise request-to-send signal
- 3 - Raise line-break signal
- 4 - Drop data-terminal-ready signal
- 5 - Drop request-to-send signal
- 6 - Drop line-break signal

port (**Int32**)

Serial port ID

- 1 - COM1
- 2 - COM2
- 3 - COM3
- 4 - COM4
- 5 - COM5
- 6 - COM6
- 7 - COM7
- 8 - COM8
- 9 - COM9
- 10 - COM10
- 11 - COM11
- 12 - COM12

Return Value

Status is returned as an int value.

- 0 - Normal return
- 1 - Invalid argument
- 2 - System exception
- 3 - SC Class not instantiated

- 4 - Port is closed
- 5 - Handshake active error

Assembly: SCICOMM (Module: SCICOMM)

Send comments on this topic to support@microglyph.com

(c) 2007 MicroGlyph Systems. All rights reserved.

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

Namespaces > SciComm > SC > COMDTM(Int32, Int32, Int32, Int32, Int32, Int32,

Int32)

Routine to get date and time.

Syntax

C#

```
public static int COMDTM (  
    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 - SC Class not instantiated

Assembly: SCICOMM (Module: SCICOMM)

Send comments on this topic to support@microglyph.com

(c) 2007 MicroGlyph Systems. All rights reserved.

COMENC Method (code, bytbuf, intbuf, svalue)

Namespaces > [SciComm](#) > [SC](#) > **COMENC(Int32, Byte[], Int32[], String)**

Routine to encode bytes, integers, or strings

Syntax

C#

```
public static int COMENC (  
    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 - SC Class not instantiated

Assembly: SCICOMM (Module: SCICOMM)

Send comments on this topic to support@microglyph.com

(c) 2007 MicroGlyph Systems. All rights reserved.

COMERR Method (Istexc, Istrtn, Isterr)

[Namespaces](#) > [SciComm](#) > [SC](#) > **COMERR(String, String, Int32)**

Routine to get error messages and error code.

[-] Syntax

C#

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

[-] Parameters

Istexc (**String**)

Last SciComm Exception

Istrtn (**String**)

Last SciComm routine in error

Isterr (**Int32**)

Last SciComm routine error code

[-] Return Value

Status is returned as an int value.

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

[-] Remarks

Method clears current error messages and code.

Assembly: SCICOMM (Module: SCICOMM)

Send comments on this topic to support@microglyph.com

(c) 2007 MicroGlyph Systems. All rights reserved.

COMFLS Method (code, port)

Namespaces > [SciComm](#) > [SC](#) > [COMFLS\(Int32, Int32\)](#)

Routine to flush serial port buffers.

Syntax

C#

```
public static int COMFLS (  
    int code,  
    int port  
)
```

Parameters

code (**Int32**)

Function code

- 1 - Flush receive buffer
- 2 - Flush send buffers
- 3 - Flush receive and send buffers

port (**Int32**)

Serial port ID

- 1 - COM1
- 2 - COM2
- 3 - COM3
- 4 - COM4
- 5 - COM5
- 6 - COM6
- 7 - COM7
- 8 - COM8
- 9 - COM9
- 10 - COM10
- 11 - COM11
- 12 - COM12

Return Value

Status is returned as an int value.

- 0 - Normal return
- 1 - Invalid argument
- 2 - System exception
- 3 - SC Class not instantiated
- 4 - Port is closed

Assembly: SCICOMM (Module: SCICOMM)

Send comments on this topic to support@microglyph.com

(c) 2007 MicroGlyph Systems. All rights reserved.

COMOPN Method (port, baud, parity, stpbts, datbts, flow, reclen, sndlen)

Namespaces > **SciComm** > **SC** > COMOPN(Int32, Int32, Int32, Int32, Int32, Int32, Int32, Int32)

Routine to open serial port.

Syntax

C#

```
public static int COMOPN (  
    int port,  
    int baud,  
    int parity,  
    int stpbts,  
    int datbts,  
    int flow,  
    int reclen,  
    int sndlen
```

Parameters

port (**Int32**)

Serial port ID

- 1 - COM1
- 2 - COM2
- 3 - COM3
- 4 - COM4
- 5 - COM5
- 6 - COM6
- 7 - COM7
- 8 - COM8
- 9 - COM9
- 10 - COM10
- 11 - COM11
- 12 - COM12

baud (**Int32**)

Baud rate

- 1 - 110
- 2 - 300
- 3 - 600
- 4 - 1200
- 5 - 2400
- 6 - 4800
- 7 - 9600

- 8 - 14400
- 9 - 19200
- 10 - 38400
- 11 - 56000
- 12 - 57600
- 13 - 115200

parity (**Int32**)

Parity marking

- 1 - Even
- 2 - Mark
- 3 - None
- 4 - Odd
- 5 - Space

stpbits (**Int32**)

Stop bits per byte

- 1 - 0
- 2 - 1
- 3 - 1.5
- 4 - 2

datbts (**Int32**)

Data bits per byte
4 to 8

flow (**Int32**)

Flow control

- 1 - None
- 2 - RTS/CTS
- 3 - RTS/CTS + XON/XOFF
- 4 - XON/XOFF

reclen (**Int32**)

Receive buffer size
4096 (default and minimum value)

sndlen (**Int32**)

Send buffer size
4096 (default and minimum value)

▣ Return Value

Status is returned as an int value.

- 0 - Normal return
- 1 - Invalid argument

- 2 - System exception
- 3 - SC Class not instantiated
- 4 - Port is open
- 5 - Send thread failed to start

Assembly: SCICOMM (Module: SCICOMM)

Send comments on this topic to support@microglyph.com

(c) 2007 MicroGlyph Systems. All rights reserved.

COMREC Method (code, port, nbytes, tmeout, delbuf, inpbuf, nread)

Namespaces > SciComm > SC > COMREC(Int32, Int32, Int32, Int32, Byte[], Byte[], Int32)

Routine to read bytes from serial port buffer.

Syntax

C#

```
public static int COMREC (  
    int code,  
    int port,  
    int nbytes,  
    int tmeout,  
    byte[] delbuf,  
    byte[] inpbuf,  
    ref int nread
```

Parameters

code (**Int32**)

Function code

- 1 - Read
- 2 - Read + timeout
- 3 - Read + timeout + delimiter

port (**Int32**)

Serial port ID

- 1 - COM1
- 2 - COM2
- 3 - COM3
- 4 - COM4
- 5 - COM5
- 6 - COM6
- 7 - COM7
- 8 - COM8
- 9 - COM9
- 10 - COM10
- 11 - COM11
- 12 - COM12

nbytes (**Int32**)

Number bytes to read (maximum)

tmeout (**Int32**)

Time out (ms)

delbuf (**Byte[]**)

Delimiter array

inpbuf (**Byte[]**)

Array to receive read bytes

nread (**Int32**)

Number of bytes read

Return Value

Status is returned as an int value.

- 0 - Normal return
- 1 - Invalid argument
- 2 - System exception
- 3 - SC Class not instantiated
- 4 - Port is closed
- 5 - Serial port errors (see COMSTS)
- 6 - No delimiter match
- 7 - Read timed out

Assembly: SCICOMM (Module: SCICOMM)

Send comments on this topic to support@microglyph.com

(c) 2007 MicroGlyph Systems. All rights reserved.

COMSND Method (code, port, nbytes, tmeout, outbuf)

Namespaces > [SciComm](#) > [SC](#) > **COMSND(Int32, Int32, Int32, Int32, Byte[])**

Routine to write bytes to serial port buffer.

Syntax

C#

```
public static int COMSND (  
    int code,  
    int port,  
    int nbytes,  
    int tmeout,  
    byte[] outbuf  
)
```

Parameters

code (**Int32**)

Function code

- 1 - Write
- 2 - Write + time stamp
- 3 - Write + timeout
- 4 - Write + time stamp + timeout

port (**Int32**)

Serial port ID

- 1 - COM1
- 2 - COM2
- 3 - COM3
- 4 - COM4
- 5 - COM5
- 6 - COM6
- 7 - COM7
- 8 - COM8
- 9 - COM9
- 10 - COM10
- 11 - COM11
- 12 - COM12

nbytes (**Int32**)

Number bytes to write

tmeout (**Int32**)

Time out (ms)

outbuf (**Byte[]**)

Output bytes

Return Value

Status is returned as an int value.

- 0 - Normal return

- 1 - Invalid argument
- 2 - System exception
- 3 - SC Class not instantiated
- 4 - Port is closed
- 5 - Write timed out
- 6 - Memory allocation error
- 7 - Send thread failure

Assembly: SCICOMM (Module: SCICOMM)

Send comments on this topic to support@microglyph.com

(c) 2007 MicroGlyph Systems. All rights reserved.

COMSTS Method (port, lsreg, msreg, baudrate, parity, stpbts, datbts, flow, reccnt, reclen, sndcnt, sndlen)

Namespaces > SciComm > SC > COMSTS(Int32, Int32, Int32, Int32, Int32, Int32,

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

Routine to get serial port status.

Syntax

C#

```
public static int COMSTS (  
    int port,  
    ref int lsreg,  
    ref int msreg,  
    ref int baudrate,  
    ref int parity,  
    ref int stpbts,  
    ref int datbts,  
    ref int flow,  
    ref int reccnt,  
    ref int reclen,  
    ref int sndcnt,  
    ref int sndlen  
)
```

Parameters

port (**Int32**)

Serial port ID

- 1 - COM1
- 2 - COM2
- 3 - COM3
- 4 - COM4
- 5 - COM5
- 6 - COM6
- 7 - COM7
- 8 - COM8
- 9 - COM9
- 10 - COM10
- 11 - COM11
- 12 - COM12

lsreg (**Int32**)

Line status register

- 7 - Not used
- 6 - Xmit shift register empty
- 5 - Xmit hold register empty
- 4 - Line break detected

- 3 - Framing error
- 2 - Parity error
- 1 - Overrun error
- 0 - Data ready

msreg (**Int32**)

Modem status register

- 7 - Carrier detect signal
- 6 - Ring indicate
- 5 - Data set ready
- 4 - Clear to send
- 3 - Not used
- 2 - Not used
- 1 - Not used
- 0 - Not used

baudrate (**Int32**)

Baud rate

parity (**Int32**)

Parity marking

- 1 - Even
- 2 - Mark
- 3 - None
- 4 - Odd
- 5 - Space

stpbits (**Int32**)

Stop bits per byte

- 1 - 0
- 2 - 1
- 3 - 1.5
- 4 - 2

datbts (**Int32**)

Data bits per byte
4 to 8

flow (**Int32**)

Flow control

- 1 - None
- 2 - RTS/CTS
- 3 - RTS/CTS + XON/XOFF

•4 - XON/XOFF

recCnt (**Int32**)
Number bytes in receive buffer
recLen (**Int32**)
Length of receive buffer
sndCnt (**Int32**)
Number bytes in send buffer
sndLen (**Int32**)
Length of send buffer

▣ **Return Value**

Status is returned as an int value.

- 0 - Normal return
- 1 - Invalid argument
- 2 - System exception
- 3 - SC Class not instantiated
- 4 - Port is closed

Assembly: SCICOMM (Module: SCICOMM)

Send comments on this topic to support@microglyph.com

(c) 2007 MicroGlyph Systems. All rights reserved.

COMSUS Method (tmeout)

Namespaces > [SciComm](#) > [SC](#) > [COMSUS\(Int32\)](#)

Routine to sleep the executing thread.

Syntax

C#

```
public static int COMSUS (  
    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 - SC Class not instantiated

Assembly: SCICOMM (Module: SCICOMM)

Send comments on this topic to support@microglyph.com

(c) 2007 MicroGlyph Systems. All rights reserved.

COMTMR Method

[Namespaces](#) > [SciComm](#) > [SC](#) > **COMTMR()**

Routine to get current time in milliseconds.

Syntax

C#

```
public static int COMTMR ()  Return Value
```

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

Assembly: SCICOMM (Module: SCICOMM)

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 SciComm Library methods.

The following links provide access to the example programs:

[TCOM](#)

Use of SciComm Library Methods.

```

*****
#*
#*   TCOM - Program to test .NET 2005 Serial Port support
#*
*****

#   System definistions and imports

import clr
import System
import IronPython

clr.AddReferenceByPartialName("System")
clr.AddReferenceByPartialName("IronPython")
clr.AddReferenceToFile("SciComm.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 SciComm         import *

#   Main program

def Main():

    try:

#       Local variables definitions

        baud      = clr.Reference[Int32](0)
        parity    = clr.Reference[Int32](0)
        stpbts    = clr.Reference[Int32](0)
        datbts    = clr.Reference[Int32](0)
        flow      = clr.Reference[Int32](0)
        reclen    = clr.Reference[Int32](0)
        sndlen    = clr.Reference[Int32](0)
        lsreg     = clr.Reference[Int32](0)
        msreg     = clr.Reference[Int32](0)
        reccnt    = clr.Reference[Int32](0)
        sndcnt    = clr.Reference[Int32](0)
        lsterr    = clr.Reference[Int32](0)
        Timeout   = clr.Reference[Int32](0)
        NRead     = clr.Reference[Int32](0)
        baudrate  = clr.Reference[Int32](0)
        ival      = clr.Reference[Int32](0)
        dialog    = System.Boolean
        PrtNames  = System.Array[String](
            ["COM1","COM2",
             "COM3","COM4","COM5","COM6","COM7","COM8","COM9","COM10",
             "COM11","COM12"])
        ParNames  = System.Array[String](
            ["Even","Mark","None","Odd","Space"])
        StpNames  = System.Array[String](
            ["None","One","OnePointFive","Two"])
        FloNames  = System.Array[String](
            ["None","Rts/Cts","Rts/Cts + Xon/Xoff","Xon/Xoff"])
        lstexc    = clr.Reference[String]("")

```

```

lstrtn = clr.Reference[String]("")
SValue = clr.Reference[String]("")
TmpBuf = CreateInstance(Byte,4096)
BytBuf = CreateInstance(Byte,4096)
DelBuf = CreateInstance(Byte,2)
IntBuf = CreateInstance(Int32,16)

# SciComm Class must be instantiated

SC()

# Set initial values

bc = " "
oc = "O"
cc = "C"
qc = "Q"
sc = "S"
rc = "R"
wc = "W"
fc = "F"
xc = "X"
mc = "M"
reply = ""
line = ""
pline = ""
port = 0
NBytes = 0
j1 = 0
j2 = 0
n = 0
i = 0
j = 0
code = 0
func = 0
status = 0

baud.Value = 0
parity.Value = 0
stpbits.Value = 0
datbits.Value = 0
flow.Value = 0
reclen.Value = 0
sndlen.Value = 0
lsreg.Value = 0
msreg.Value = 0
reccnt.Value = 0
sndcnt.Value = 0
lsterr.Value = 0
Timeout.Value = 0
NRead.Value = 0
baudrate.Value = 0
ival.Value = 0

# Start Main

# Dialog loop

dialog = True
while dialog:

```

```

#      Function Request Message

Console.WriteLine("\n TCOM Functions:\n")
Console.WriteLine("    O = Open      (COMOPN)")
Console.WriteLine("    C = Close     (COMCLS)")
Console.WriteLine("    S = Status    (COMSTS)")
Console.WriteLine("    R = Read      (COMREC)")
Console.WriteLine("    W = Write     (COMSND)")
Console.WriteLine("    F = Flush     (COMFLS)")
Console.WriteLine("    X = Control   (COMCTL)")
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 == oc :

#      Open serial port

#      Get Serial Port ID
Console.WriteLine("\n      Port:\n")
Console.WriteLine("    1 = COM1")
Console.WriteLine("    2 = COM2")
Console.WriteLine("    3 = COM3")
Console.WriteLine("    4 = COM4")
Console.WriteLine("    5 = COM5")
Console.WriteLine("    6 = COM6")
Console.WriteLine("    7 = COM7")
Console.WriteLine("    8 = COM8")
Console.WriteLine("    9 = COM9")
Console.WriteLine("   10 = COM10")
Console.WriteLine("   11 = COM11")
Console.WriteLine("   12 = COM12")
Console.Write("\n      Enter Port[1]: ")
port = 1
line = Console.ReadLine()
if line.Length > 0:
    Int32.TryParse(line,ival)
#      Set select port value
    port = ival.Value

#      Set baud
Console.WriteLine("\n      Baud Rate:\n")
Console.WriteLine("    1 =    110")
Console.WriteLine("    2 =    300")
Console.WriteLine("    3 =    600")

```

```

Console.WriteLine("      4 = 1200")
Console.WriteLine("      5 = 2400")
Console.WriteLine("      6 = 4800")
Console.WriteLine("      7 = 9600")
Console.WriteLine("      8 = 14400")
Console.WriteLine("      9 = 19200")
Console.WriteLine("     10 = 38400")
Console.WriteLine("     11 = 56000")
Console.WriteLine("     12 = 57600")
Console.WriteLine("     13 = 115200")
Console.Write("\n      Enter Baud[7]: ")
baud.Value = 7
line = Console.ReadLine()
if line.Length > 0:
    Int32.TryParse(line,ival)
#       Set select baud value
#       baud.Value = ival.Value
Set parity
Console.WriteLine("\n      Parity:\n")
Console.WriteLine("      1 = Even")
Console.WriteLine("      2 = Mark")
Console.WriteLine("      3 = None")
Console.WriteLine("      4 = Odd")
Console.WriteLine("      5 = Space")
Console.Write("\n      Enter Parity[3]: ")
parity.Value = 3
line = Console.ReadLine()
if line.Length > 0:
    Int32.TryParse(line,ival)
#       Set select parity value
#       parity.Value = ival.Value
Set stop bits
Console.WriteLine("\n      Stop Bits:\n")
Console.WriteLine("      1 = 0")
Console.WriteLine("      2 = 1")
Console.WriteLine("      3 = 1.5")
Console.WriteLine("      4 = 2")
Console.Write("\n      Enter Stop Bits[2]: ")
stpbts.Value = 2
line = Console.ReadLine()
if line.Length > 0:
    Int32.TryParse(line,ival)
#       Set select stop bits value
#       stpbts.Value = ival.Value
Set data bits
Console.WriteLine("\n      Data Bits:\n")
Console.WriteLine("      4 to 8")
Console.Write("\n      Enter Data Bits[8]: ")
datbts.Value = 8
line = Console.ReadLine()
if line.Length > 0:
    Int32.TryParse(line,ival)
#       Set select data bits value
#       datbts.Value = ival.Value
Set flow
Console.WriteLine("\n      Flow:\n")
Console.WriteLine("      1 = None")
Console.WriteLine("      2 = Rts/Cts")
Console.WriteLine("      3 = Rts/Cts + Xon/Xoff")
Console.WriteLine("      4 = Xon/Xoff")

```

```

Console.WriteLine("\n    Enter Flow[1]: ")
flow.Value = 1
line = Console.ReadLine()
if line.Length > 0:
    Int32.TryParse(line,ival)
#       Set select flow bits value
#       flow.Value = ival.Value
#
Set reflen
Console.WriteLine("    Enter Reclen[4096]:")
reflen.Value = 4096
line = Console.ReadLine()
if line.Length > 0:
    Int32.TryParse(line,ival)
#       Set select reflen value
#       reflen.Value = ival.Value
#
Set sndlen
Console.WriteLine("    Enter Sndlen[4096]:")
sndlen.Value = 4096
line = Console.ReadLine()
if line.Length > 0:
    Int32.TryParse(line,ival)
#       Set select sndlen value
#       sndlen.Value = ival.Value
#
Open port
status = SC.COMOPN(port,baud.Value,parity.Value,
                  stpbts.Value,databts.Value,flow.Value,
                  reflen.Value,sndlen.Value)

if status != 0:
    Console.WriteLine("\n        COMOPN Errors: \n")
    Console.WriteLine("        Status = {0}",status)
    Console.WriteLine("        Port    = {0}",port)
    Console.WriteLine("        Baud    = {0}",baud.Value)
    Console.WriteLine("        Parity  = {0}",parity.Value)
    Console.WriteLine("        Stpbts = {0}",stpbts.Value)
    Console.WriteLine("        Databts = {0}",databts.Value)
    Console.WriteLine("        Flow    = {0}",flow.Value)
    Console.WriteLine("        Reclen = {0}",reflen.Value)
    Console.WriteLine("        Sndlen = {0}",sndlen.Value)
    continue

if reply == cc :

#
    Close serial port

    Console.WriteLine("\n        Close Functions:\n")
    Console.WriteLine("        1 = Purge Receive/Send Buffers")
    Console.WriteLine("        2 = Purge Receive/Wait on" +
                    " Send Buffers")

    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
    Console.WriteLine("        Enter Port    [1]: ")
    port = 1
    line = Console.ReadLine()
    if line.Length > 0:
        Int32.TryParse(line,ival)

```

```

#         Set select port value
         port = ival.Value
status = SC.COMCLS(code,port)
if status != 0:
    Console.WriteLine("\n          COMCLS Errors: \n")
    Console.WriteLine("          Status = {0}",status)
    Console.WriteLine("          Code   = {0}",code)
    Console.WriteLine("          Port   = {0}",port)
    continue

if reply == mc :

#         Menu command

        continue

if reply == sc :

#         Serial port status

        Console.Write("\n      Enter Port      [1]: ")
        port = 1
        line = Console.ReadLine()
        if line.Length > 0:
            Int32.TryParse(line,ival)
#         Set select port value
            port = ival.Value
status = SC.COMSTS(port,lsreg,msreg,baudrate,parity,
                  stpbts,databts,flow,reccnt,reclen,
                  sndcnt,sndlen)

if status == 0:
    Console.WriteLine("\n          COMSTS status:\n")
    Console.WriteLine("          Port      = {0}",
                    PrtNames[port-1])
    Console.WriteLine("          Lsreg     = {0,2:X2}",
                    lsreg.Value)
    Console.WriteLine("          Msreg     = {0,2:X2}",
                    msreg.Value)
    Console.WriteLine("          Baud      = {0}",
                    baudrate.Value)
    Console.WriteLine("          Parity    = {0}",
                    ParNames[parity.Value-1])
    Console.WriteLine("          Stpbts    = {0}",
                    StpNames[stpbts.Value-1])
    Console.WriteLine("          Databts   = {0}",databts.Value)
    Console.WriteLine("          Flow      = {0}",
                    FloNames[flow.Value-1])
    Console.WriteLine("          Reccnt    = {0}",reccnt.Value)
    Console.WriteLine("          Reclen    = {0}",reclen.Value)
    Console.WriteLine("          Sndcnt    = {0}",sndcnt.Value)
    Console.WriteLine("          Sndlen    = {0}",sndlen.Value)
else:
    Console.WriteLine("\n          COMSTS Errors: \n")
    Console.WriteLine("          Status = {0}",status)
    Console.WriteLine("          Port   = {0}",port)

        continue

if reply == rc :

```

```

#         Read bytes from receive buffer

Console.WriteLine("\n    Receive Functions:\n")
Console.WriteLine("        1 = Read")
Console.WriteLine("        2 = Read with Timeout")
Console.WriteLine("        3 = Read with Timeout/Delimiter")
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
Console.Write("    Enter Port    [1]: ")
port = 1
line = Console.ReadLine()
if line.Length > 0:
    Int32.TryParse(line,ival)
#         Set select port value
    port = ival.Value
Timeout    = 3000
DelBuf[0]  = 13
DelBuf[1]  = 10
NBytes     = 4096
status     = SC.COMREC (code,port,NBytes,Timeout,DelBuf,TmpBuf,NRead)
if status == 0:
#         Display bytes received
    if NRead.Value == 0:
        Console.WriteLine("\n            No Data Available")
        Console.WriteLine("            Code = {0}",code)
        Console.WriteLine("            Port = {0}",port)
    if NRead.Value > 0:
        NBytes = NRead.Value
        BytBuf = CreateInstance(Byte,NBytes)
#         Move number of bytes read into Buffer
        for i in range(NBytes):
            BytBuf[i] = TmpBuf[i]
        Console.WriteLine("\n            Count: {0}",NBytes)
#         Convert bytes read to ASCII string
        code    = 5
        status  = SC.COMENC (code,BytBuf,IntBuf,SValue)
#         Display ASCII string
        for i in range(0,NBytes,32):
            n      = Math.Min(32,NBytes-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,11):
            j1 = i
            j2 = i+Math.Min(11,NBytes-i)
            if i == 0:
                Console.Write("            Hex : ")
                for j in range(j2):
                    Console.Write("{0,2:X2} ",BytBuf[j])
                Console.WriteLine(" ")
            else:
                Console.Write("            : ")

```

```

        for j in range(j2) :
            Console.Write("{0,2:X2} ",BytBuf[j])
        Console.WriteLine(" ")
    else:
        Console.WriteLine("\n        COMREC Errors: \n")
        Console.WriteLine("        Status = {0}",status)
        Console.WriteLine("        Code   = {0}",code)
        Console.WriteLine("        Port   = {0}",port)
    continue

if reply == wc :

#       Write out bytes to send buffer

    Console.WriteLine("\n        Write Functions:\n")
    Console.WriteLine("        1 = Write")
    Console.WriteLine("        2 = Write with Timestamp")
    Console.WriteLine("        3 = Write with Timeout")
    Console.WriteLine("        4 = Write with Timeout/Timestamp")
    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
    Console.Write("        Enter Port      [1]: ")
    port = 1
    line = Console.ReadLine()
    if line.Length > 0:
        Int32.TryParse(line,ival)
#         Set select port value
        port = ival.Value
    Console.Write("        Enter String      : ")
    line = Console.ReadLine()
#     Add carriage return to end of line for delimiter reads
#     line = line + "\r\n"
    NBytes = line.Length
    Timeout = 3000
#     Convert string to ASCII bytes
    func = 3
    BytBuf = CreateInstance(Byte,NBytes)
    SValue.Value = line
    status = SC.COMENC(func,BytBuf,IntBuf,SValue)
#     Send the ASCII bytes
    status = SC.COMSND(code,port,NBytes,Timeout,BytBuf)
    if status != 0:
        Console.WriteLine("\n        COMSND Errors: \n")
        Console.WriteLine("        Status = {0}",status)
        Console.WriteLine("        Code   = {0}",code)
        Console.WriteLine("        Port   = {0}",port)

if reply == fc :

#       Flush receive and send buffers

    Console.WriteLine("\n        Flush Functions:\n")
    Console.WriteLine("        1 = Flush Receive Buffers")
    Console.WriteLine("        2 = Flush Send Buffers")
    Console.WriteLine("        3 = Flush Receive/Send Buffers")

```

```

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
Console.Write("    Enter Port    [1]: ")
port = 1
line = Console.ReadLine()
if line.Length > 0:
    Int32.TryParse(line,ival)
#    Set select port value
    port = ival.Value
status = SC.COMFLS(code,port)
if status <> 0:
    Console.WriteLine("\n        COMFLS Errors: \n")
    Console.WriteLine("        Status = {0}",status)
    Console.WriteLine("        Code   = {0}",code)
    Console.WriteLine("        Port   = {0}",port)

    continue

if reply == xc :

#    Set control lines

    Console.WriteLine("\n    Control Functions:\n")
    Console.WriteLine(
        "        1 = Raise Data-Terminal-Ready Signal")
    Console.WriteLine(
        "        2 = Raise Request-to-Send Signal")
    Console.WriteLine(
        "        3 = Raise Line-Break Signal")
    Console.WriteLine(
        "        4 = Drop Data-Terminal-Ready Signal")
    Console.WriteLine(
        "        5 = Drop Request-to-Send Signal")
    Console.WriteLine(
        "        6 = Drop Line-Break Signal")
    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
    Console.Write("    Enter Port    [1]: ")
    port = 1
    line = Console.ReadLine()
    if line.Length > 0:
        Int32.TryParse(line,ival)
#        Set select port value
        port = ival.Value
    status = SC.COMCTL(code,port)
    if status != 0:
        Console.WriteLine("\n        COMCTL Errors: \n")
        Console.WriteLine("        Status = {0}",status)
        Console.WriteLine("        Code   = {0}",code)
        Console.WriteLine("        Port   = {0}",port)

```

```

        continue

    if reply == qc :
#       Exit processing

        dialog = False

        continue

#   Dialog process loop

#   Report any errors

    status = SC.COMERR(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()

```