

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

Visual Basic

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

Visual Basic

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

Visual Basic

Public Sub **New**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

Visual Basic

```
Public Shared Function COMCLS ( _  
    code As Integer, _  
    port As Integer _  
) As Integer
```

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

Visual Basic

```
Public Shared Function COMCTL ( _  
    code As Integer, _  
    port As Integer _  
) As Integer
```

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

Visual Basic

```
Public Shared Function COMDTM ( _  
    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 - 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

Visual Basic

```
Public Shared Function COMENC ( _  
    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 - 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

Visual Basic

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

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

Visual Basic

```
Public Shared Function COMFLS ( _  
    code As Integer, _  
    port As Integer _  
) As Integer
```

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

Visual Basic

```
Public Shared Function COMOPN ( _  
    port As Integer, _  
    baud As Integer, _  
    parity As Integer, _  
    stpbts As Integer, _  
    datbts As Integer, _  
    flow As Integer, _  
    reclen As Integer, _  
    sndlen As Integer _  
) As Integer
```

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

Visual Basic

```
Public Shared Function COMREC ( _  
    code As Integer, _  
    port As Integer, _  
    nbytes As Integer, _  
    tmeout As Integer, _  
    delbuf As Byte(), _  
    inpbuf As Byte(), _  
    ByRef nread As Integer _  
) As Integer
```

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

Visual Basic

```
Public Shared Function COMSND ( _  
    code As Integer, _  
    port As Integer, _  
    nbytes As Integer, _  
    tmeout As Integer, _  
    outbuf As Byte() _  
) As Integer
```

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

Visual Basic

```
Public Shared Function COMSTS ( _  
    port As Integer, _  
    ByRef lsreg As Integer, _  
    ByRef msreg As Integer, _  
    ByRef baudrate As Integer, _  
    ByRef parity As Integer, _  
    ByRef stpbts As Integer, _  
    ByRef datbts As Integer, _  
    ByRef flow As Integer, _  
    ByRef reccnt As Integer, _  
    ByRef reclen As Integer, _  
    ByRef sndcnt As Integer, _  
    ByRef sndlen As Integer _  
)
```

As Integer

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

Visual Basic

```
Public Shared Function COMSUS ( _  
    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 - 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

Visual Basic

Public Shared Function COMTMR **As Integer** **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
'*
'*****

Imports System
Imports SciComm

Public Class TCOM

'   Instantiate SciComm Class

    Public Shared Dim SObject As SciComm.SC = New SciComm.SC()

'   Main Program

    Public Shared Sub Main ()

'   Local variables definitions

        Dim port,baud,parity,stpbits,datbts,flow,reclen,sndlen,lsreg,
            msreg,recCnt,sndcnt,lsterr,NBytes,Timeout,NRead,rtnsts,func,
            baudrate,j1,j2,n,ival,i,j,status,code As Integer
        Dim dialog As Boolean
        Dim reply,lstexc,lstrtn,line,pline,bc,oc,cc,qc,sc,rc,wc,fc,xc,
            mc As String
        Dim PrtNames() As String = New String(11) {"COM1","COM2",
            "COM3","COM4","COM5","COM6","COM7","COM8","COM9","COM10",
            "COM11","COM12"}
        Dim ParNames() As String = New String(4) {"Even","Mark",
            "None","Odd","Space"}
        Dim StpNames() As String = New String(3) {"None","One",
            "OnePointFive","Two"}
        Dim FloNames() As String = New String(3) {"None",
            "Rts/Cts","Rts/Cts + Xon/Xoff","Xon/Xoff"}
        Dim TmpBuf(4096) As Byte
        Dim DelBuf(1) As Byte
        Dim IntBuf(15) As Integer

'   Set initial values

        port = 1
        bc = " "
        oc = "O"
        cc = "C"
        qc = "Q"
        sc = "S"
        rc = "R"
        wc = "W"
        fc = "F"
        xc = "X"
        mc = "M"
        lstexc = ""
        lstrtn = ""

'   Start Main

        Try

```

```

' Dialog loop

dialog = True
While (dialog)

' Function Request Message

Console.WriteLine(vbNewLine + " TCOM Functions:" + vbNewLine)
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(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 character

Case oc

' Open serial port

port = 0
baud = 0
parity = 0
stpbits = 0
datbits = 0
flow = 0
reclen = 0
sndlen = 0

' Get Serial Port ID
Console.WriteLine(vbNewLine + " Port:" + vbNewLine)
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(vbNewLine + "    Enter Port[1]: ")
port = 1
line = Console.ReadLine()
If (line.Length > 0) Then
    Int32.TryParse(line,ival)
    Set select port value
    port = ival
End If
Set baud
Console.WriteLine(vbNewLine + "    Baud Rate:" + vbNewLine)
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(vbNewLine + "    Enter Baud[7]: ")
baud = 7
line = Console.ReadLine()
If (line.Length > 0) Then
    Int32.TryParse(line,ival)
    Set select baud value
    baud = ival
End If
Set parity
Console.WriteLine(vbNewLine + "    Parity:" + vbNewLine)
Console.WriteLine("    1 = Even")
Console.WriteLine("    2 = Mark")
Console.WriteLine("    3 = None")
Console.WriteLine("    4 = Odd")
Console.WriteLine("    5 = Space")
Console.Write(vbNewLine + "    Enter Parity[3]: ")
parity = 3
line = Console.ReadLine()
If (line.Length > 0) Then
    Int32.TryParse(line,ival)
    Set select parity value
    parity = ival
End If
Set stop bits
Console.WriteLine(vbNewLine + "    Stop Bits:" + vbNewLine)
Console.WriteLine("    1 = 0")
Console.WriteLine("    2 = 1")
Console.WriteLine("    3 = 1.5")
Console.WriteLine("    4 = 2")
Console.Write(vbNewLine + "    Enter Stop Bits[2]: ")
stpbits = 2
line = Console.ReadLine()
If (line.Length > 0) Then
    Int32.TryParse(line,ival)

```

```

'         Set select stop bits value
          stpbts = ival
End If
'
Set data bits
Console.WriteLine(vbNewLine + "    Data Bits:" + vbNewLine)
Console.WriteLine("        4 to 8")
Console.Write(vbNewLine + "    Enter Data Bits[8]: ")
datbts = 8
line = Console.ReadLine()
If (line.Length > 0) Then
    Int32.TryParse(line,ival)
'        Set select data bits value
          datbts = ival
End If
'
Set flow
Console.WriteLine(vbNewLine + "    Flow:" + vbNewLine)
Console.WriteLine("        1 = None")
Console.WriteLine("        2 = Rts/Cts")
Console.WriteLine("        3 = Rts/Cts + Xon/Xoff")
Console.WriteLine("        4 = Xon/Xoff")
Console.Write(vbNewLine + "    Enter Flow[1]: ")
flow = 1
line = Console.ReadLine()
If (line.Length > 0) Then
    Int32.TryParse(line,ival)
'        Set select flow bits value
          flow = ival
End If
'
Set reclen
Console.Write("    Enter Reclen[4096]:")
reclen = 4096
line = Console.ReadLine()
If (line.Length > 0) Then
    Int32.TryParse(line,ival)
'        Set select reclen value
          reclen = ival
End If
'
Set sndlen
Console.Write("    Enter Sndlen[4096]:")
sndlen = 4096
line = Console.ReadLine()
If (line.Length > 0) Then
    Int32.TryParse(line,ival)
'        Set select sndlen value
          sndlen = ival
End If
'
Open port
status = SciComm.SC.COMOPN(port,baud,parity,stpbts,datbts,
                            flow,reclen,sndlen)
If (status <> 0) Then
    Console.WriteLine(vbNewLine +
        "        COMOPN Errors: " + vbNewLine)
        Console.WriteLine("        Status = {0}",status)
        Console.WriteLine("        Port   = {0}",port)
        Console.WriteLine("        Baud   = {0}",baud)
        Console.WriteLine("        Parity = {0}",parity)
        Console.WriteLine("        Stpbts = {0}",stpbts)
        Console.WriteLine("        Datbts = {0}",datbts)
        Console.WriteLine("        Flow   = {0}",flow)
        Console.WriteLine("        Reclen = {0}",reclen)

```

```

        Console.WriteLine("          Sndlen = {0}",sndlen)
    End If

Case cc

    '
    Close serial port

    Console.WriteLine(vbNewLine +
        "          Close Functions:" + vbNewLine)
    Console.WriteLine("          1 = Purge Receive/Send Buffers")
    Console.WriteLine("          2 = Purge Receive/Wait on" +
        "          Send Buffers")
    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
    Console.Write("          Enter Port      [1]: ")
    port = 1
    line = Console.ReadLine()
    If (line.Length > 0) Then
        Int32.TryParse(line,ival)
        '
        Set select port value
        port = ival
    End If
    status = SciComm.SC.COMCLS (code,port)
    If (status <> 0) Then
        Console.WriteLine(vbNewLine +
            "          COMCLS Errors: " + vbNewLine)
        Console.WriteLine("          Status = {0}",status)
        Console.WriteLine("          Code   = {0}",code)
        Console.WriteLine("          Port   = {0}",port)
    End If

Case mc

    '
    Menu command

Case sc

    '
    Serial port status

    Console.Write(vbNewLine + "          Enter Port      [1]: ")
    port = 1
    line = Console.ReadLine()
    If (line.Length > 0) Then
        Int32.TryParse(line,ival)
        '
        Set select port value
        port = ival
    End If
    status = SciComm.SC.COMSTS (port,lsreg,msreg,baudrate,parity,
        stpbts,datbts,flow,recnt,reclen,
        sndcnt,sndlen)

    If (status = 0)
        Then
            Console.WriteLine(vbNewLine +
                "          COMSTS status:" + vbNewLine)

```

```

        Console.WriteLine("          Port      = {0}",
                           PrtNames(port-1))
        Console.WriteLine("          Lsreg     = {0,2:X2}",lsreg)
        Console.WriteLine("          Msreg     = {0,2:X2}",msreg)
        Console.WriteLine("          Baud      = {0}",baudrate)
        Console.WriteLine("          Parity    = {0}",
                           ParNames(parity-1))
        Console.WriteLine("          Stpbts   = {0}",
                           StpNames(stpbts-1))
        Console.WriteLine("          Datbts   = {0}",datbts)
        Console.WriteLine("          Flow     = {0}",
                           FloNames(flow-1))
        Console.WriteLine("          Reccnt   = {0}",reccnt)
        Console.WriteLine("          Reclen   = {0}",reclen)
        Console.WriteLine("          Sndcnt   = {0}",sndcnt)
        Console.WriteLine("          Sndlen   = {0}",sndlen)
    Else
        Console.WriteLine(vbNewLine +
                           "          COMSTS Errors: " + vbNewLine)
        Console.WriteLine("          Status = {0}",status)
        Console.WriteLine("          Port   = {0}",port)
    End If

```

Case rc

Read bytes from receive buffer

```

        Console.WriteLine(vbNewLine +
                           "          Receive Functions:" + vbNewLine)
        Console.WriteLine("          1 = Read")
        Console.WriteLine("          2 = Read with Timeout")
        Console.WriteLine("          3 = Read with Timeout/Delimiter")
        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
        Console.WriteLine("          Enter Port      [1]: ")
        port = 1
        line = Console.ReadLine()
        If (line.Length > 0) Then
            Int32.TryParse(line,ival)
            Set select port value
            port = ival
        End If
        Timeout      = 3000
        DelBuf(0)    = 13
        DelBuf(1)    = 10
        NBytes       = 4096
        status       = SciComm.SC.COMREC (code,port,NBytes,Timeout,
                                         DelBuf,TmpBuf,NRead)
        If (status = 0)
            Then
                Display bytes received
                If (NRead = 0) Then
                    Console.WriteLine(vbNewLine +
                                       "          No Data Available")
                End If
            End If
        End If

```

```

        Console.WriteLine("          Code = {0}",code)
        Console.WriteLine("          Port = {0}",port)
    End If
    If (NRead > 0) Then
        NBytes = NRead
        Dim BytBuf(NBytes-1) As Byte
        Move number of bytes read into Buffer
        For i = 0 To (NRead-1)
            BytBuf(i) = TmpBuf(i)
        Next i
        Console.WriteLine(
            vbNewLine + "          Count: {0}",NRead)
        Convert bytes read to ASCII string
        code = 5
        line = ""
        rtnsts = SciComm.SC.COMENC(code,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("          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 +
                "          COMREC Errors: " + vbNewLine)
            Console.WriteLine("          Status = {0}",status)
            Console.WriteLine("          Code = {0}",code)
            Console.WriteLine("          Port = {0}",port)
        End If

    Case wc

        Write out bytes to send buffer

        Console.WriteLine(vbNewLine +

```

```

        "        Write Functions:" + vbNewLine)
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(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
Console.Write("        Enter Port        [1]: ")
port = 1
line = Console.ReadLine()
If (line.Length > 0) Then
    Int32.TryParse(line,ival)
'        Set select port value
        port = ival
End If
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
Dim BytBuf(NBytes-1) As Byte
rtnsts = SciComm.SC.COMENC(func,BytBuf,IntBuf,line)
'        Send the ASCII bytes
status = SciComm.SC.COMSND(code,port,NBytes,Timeout,BytBuf)
If (status <> 0) Then
    Console.WriteLine(vbNewLine +
        "                COMSND Errors: " + vbNewLine)
        Console.WriteLine("                Status = {0}",status)
        Console.WriteLine("                Code   = {0}",code)
        Console.WriteLine("                Port   = {0}",port)
End If

Case fc

'        Flush receive and send buffers

Console.WriteLine(vbNewLine +
    "        Flush Functions:" + vbNewLine)
    Console.WriteLine("        1 = Flush Receive Buffers")
    Console.WriteLine("        2 = Flush Send Buffers")
    Console.WriteLine("        3 = Flush Receive/Send Buffers")
    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
Console.Write("        Enter Port        [1]: ")
port = 1

```

```

line = Console.ReadLine()
If (line.Length > 0) Then
    Int32.TryParse(line,ival)
    Set select port value
    port = ival
End If
status = SciComm.SC.COMFLS (code,port)
If (status <> 0) Then
    Console.WriteLine (vbNewLine +
        "          COMFLS Errors: " + vbNewLine)
        Console.WriteLine ("          Status = {0}",status)
        Console.WriteLine ("          Code   = {0}",code)
        Console.WriteLine ("          Port   = {0}",port)
End If

Case xc

    Set control lines

    Console.WriteLine (vbNewLine +
        "          Control Functions:" + vbNewLine)
    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 (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
    Console.Write ("          Enter Port      [1]: ")
    port = 1
    line = Console.ReadLine()
    If (line.Length > 0) Then
        Int32.TryParse(line,ival)
        Set select port value
        port = ival
    End If
    status = SciComm.SC.COMCTL (code,port)
    If (status <> 0) Then
        Console.WriteLine (vbNewLine +
            "          COMCTL Errors: " + vbNewLine)
            Console.WriteLine ("          Status = {0}",status)
            Console.WriteLine ("          Code   = {0}",code)
            Console.WriteLine ("          Port   = {0}",port)
    End If

Case qc

```

```

'      Exit processing

      dialog = false

      Case Else
'      Unknown command

      End Select

'      Dialog process loop

      End While

'      Report any errors

      status = SciComm.SC.COMERR(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

```