

MicroGlyph/SciSnet™

A Computer Sockets Communication Library

Programming Reference Manual

Release 7.0
Copyright © 2007
MicroGlyph Systems
All rights reserved

MicroGlyph Systems
P.O. Box 474
Lexington, MA 02420-0005

Tel: (781) 861-0426
Fax: (781) 674-1179

Email: support@microglyph.com
Web: <http://www.microglyph.com>

PREFACE

This programming reference manual provides technical information for Fortran programming development using the SciSnet sockets communication library. SciSnet supports PC compatible computers with network controllers under Microsoft Windows operating systems. SciSnet supports up to 100 network connections with overlapped input/output messaging. Transfer rates are dependent on the network connectivity provided. SciSnet applications can be created and execute under Microsoft Windows operating systems as Windows based applications. The SciSnet library is available in versions that are compatible with a wide variety of Fortran compilers. This distribution includes the SciSnet sockets communication library with a reference manual, an example source program, and a help file.

NOTICE

MicroGlyph Systems reserves the right to make changes without notice to information contained in this manual. MicroGlyph Systems assumes no responsibility for any errors or consequential damages that may result from use or misinterpretation of any information contained herein. No part of this manual may be copied or reproduced without prior written permission from MicroGlyph Systems.

Trademark Acknowledgements

All brand or product names used in this manual are the trademarks or registered trademarks of their respective holders.

Contents

Tables	iv
Examples	iv
CHAPTER 1 INTRODUCTION.....	1
1.1 GENERAL INFORMATION	1
1.2 SYSTEM REQUIREMENTS	1
CHAPTER 2 COMMUNICATION ENVIRONMENT.....	2
CHAPTER 3 PROGRAMMING REFERENCE	3
3.1 COMMUNICATION LIBRARY	3
3.2 COMMUNICATION FUNCTIONS	3
3.2.1 NETOPN	3
3.2.2 NETCLS	6
3.2.3 NETSTS	8
3.2.4 NETSRV	11
3.2.5 NETFLS	14
3.2.6 NETREC	17
3.2.7 NETSND	19
3.2.8 NETDTM	22
3.2.9 NETSUS.....	24
3.3 COMMUNICATION TEST PROGRAM	26
3.4 COMMUNICATION LIBRARY	26
CHAPTER 4 PERFORMANCE CONSIDERATIONS	27
4.1 ETHERNET CONTROLLER MEASURED DATA.....	27
INDEX	28
Software License Agreement	29
Software License Registration.....	32

Software Report 33

Tables

Table 4.1 – Ethernet Controller Measurements	27
--	----

Examples

Example 3.1 – NETOPN Program Example	6
Example 3.2 – NETCLS Program Example	8
Example 3.3 – NETSTS Program Example	11
Example 3.4 – NETSRV Program Example.....	14
Example 3.5 – NETFLS Program Example	16
Example 3.6 – NETREC Program Example	19
Example 3.7 – NETSND Program Example	22
Example 3.8 – NETDTM Program Example	23
Example 3.9 – NETSUS Program Example.....	24

Chapter 1

Introduction

SciSnet provides a Fortran TCP/IP based sockets communications library. SciSnet functions provide two way machine-to-machine communications over network interfaces. SciSnet utilizes the Windows sockets API that maintains separate input and output streams, and allows parallel asynchronous two-way machine-to-machine communications with TCP/IP data integrity. This guarantees the order of the messages and the integrity of every bit.

1.1 General Information

SciSnet was designed for high speed machine-to-machine data transfers. Two data transfer threads are used to receive and send data messages between two communicating machines. The NETSND SciSnet function queues a data message to be transmitted. The data transfer thread handles immediate transfer of the queued message. The input data thread queues input messages to be retrieved later by the NETREC SciSnet function. Basically, an application can set up a send and receive node on one machine, and a send and receive node on another. One machine can acquire hardware sensor or otherwise gathered information, and send this information to the other SciSnet connected machine. This allows transfer of data at very high rates in an automated way.

1.2 System Requirements

SciSnet was designed for PC compatible computers using Microsoft Windows operating systems with the Windows sockets API and network support for TCP/IP. TCP/IP must be installed and configured with local area networks, and/or the Internet. Name lookup must be enabled and be available.

Chapter 2

Communication Environment

Before communication can take place between two machines, a network server (NETSRV) must be initialized on each machine. The Windows operating system will then perform the communication task between the two systems. The operating system determines if the socket is available and connected. SciSnet provides the capability to connect to these two machines through the use of the NETOPN routine. Sockets can be disconnected by using the NETCLS routine. At the time the socket is opened by NETOPN, the communication parameters and buffering protocol are set for the connection period.

Chapter 3

Programming Reference

This program reference section provides the basic information necessary for a Fortran user to write application programs that will use sockets communication functions provided by the SciSnet library. The various subsections provide all the communication functions representative of a complete communication system. Routines to open and close communication sockets, flush buffers, report on sockets status, and do the network communication are included in the SciSnet communication library.

3.1 Communication Library

SciSnet routines are released as a DLL library. The SCNIMP.LIB import library is used with the SCNDLL.DLL dynamic link library to dynamically link an application to produce an executable file. The user should consult the Fortran user's manual released by the compiler manufacturer for the exact format of the options that will compile and link the application. The RUN.BAT and README.TXT files in this release contain such rudimentary information.

3.2 Communication Functions

SciSnet communication routines are contained in the SCNDLL.DLL libraries created by the library manager facility. This assures compatibility with the Fortran compiler object files and the linking loader that should be used to create and link the application Fortran programs to be executed. The user should consult the Fortran user's manual for the exact format of the Fortran language command that will compile the application programs. The following sections describe the network communication functions and their Fortran calling sequences.

3.2.1 NETOPN

Purpose: - This function is used to open of a network connection between a local machine and another machine on the network. A search for the remote host name is made. Upon successful lookup, the network connection is made and the specified options are set in the SciSnet control structure.

Fortran Calling Sequence:

```
CALL NETOPN(NETID,LPRNAME,RPORT,CODE,RTIME,CTIME,  
            STATUS)
```

Where:

NETID	- Network connection ID	(input integer)
	1 thru 100	

LPRNAME	- Remote host name address	(input integer)
RPORT	- Remote host receive port	(input integer)
	Valid range 0 to 65535	
CODE	- Option code	(input integer)
	0 = None	
	1 = Use timestamp with output messages	
	2 = Use timeout value receiving input messages	
RTIME	- Receive wait timeout (ms)	(input integer)
CTIME	- Connect wait timeout (ms)	(input integer)
STATUS	- Returned status	(output integer)
	0 = Normal return	
	1 = Connection already open	
	2 = Invalid arguments	
	3 = Local server not running	
	4 = Remote host not found	

Notes:

1. NETOPN establishes a network connection between two network connected machines. Up to 100 connections can be established at any one time.
2. The remote host receive port is an agreed upon number that is used by NETSRV on each machine in a connection pair as the local host receive port.
3. The NETSRV function must be called before attempting to call the NETOPN function.
4. To change the options established by NETOPN, it is necessary to call NETCLS first (if a network connection is open), before calling NETOPN again.

Error Conditions:

1. See STATUS parameter definition for all error codes.

Program Example:

```

PROGRAM TNETOPN

!   SciSnet interface definition

USE SCNINF
IMPLICIT NONE

!   Variables Definitions

```

```

    INTEGER(4) :: NETID, LPORT, RPORT, RLEN, SLEN, CODE, RTIME,
+   CTIME, STATUS, I, LPRNAME
    CHARACTER(1),DIMENSION(9) :: RNAME = ('w','i','n','2','0','0',
+   '0','a',' '/')

!*****!
!
!   Initialize Network servers and connections
!
!*****!

!   Start up the network server threads

    CODE   = 1
    LPORT  = 8001
    RLEN   = 1440
    SLEN   = 1440
    CALL NETSRV(CODE,LPORT,RLEN,SLEN,STATUS)
    IF (STATUS.NE.0) THEN
        WRITE(6,100)CODE,LPORT,RLEN,SLEN,STATUS
    END IF

!   Open a network socket

    NETID  = 1
    RNAME(9:9) = CHAR(0)
    RPORT  = 8001
    CODE   = 0
    RTIME  = 0
    CTIME  = 0
    LPRNAME = LOC(RNAME)
    CALL NETOPN(NETID,LPRNAME,RPORT,CODE,RTIME,CTIME,STATUS)
    IF (STATUS.NE.0) THEN
        WRITE(6,101)NETID,(RNAME(I),I=1,9),RPORT,CODE,RTIME,CTIME,
+   STATUS
    END IF

!*****!
!
!   Close down Network servers and connections
!
!*****!

!   Close down the network connection

    code = 2
    CALL NETCLS(NETID,CODE,STATUS)
    IF (STATUS.NE.0) THEN
        WRITE(6,102)NETID,CODE,STATUS
    END IF

!   Stop the network server threads

    CODE   = 2
    CALL NETSRV(CODE,LPORT,RLEN,SLEN,STATUS)
    IF (STATUS.NE.0) THEN
        WRITE(6,100)CODE,LPORT,RLEN,SLEN,STATUS
    END IF

!   Stop execution

    STOP

!   Format Statements

100  FORMAT(' *** Errors: (NETSRV) *** \/,
+   \ CODE   = \,I10,/,
+   \ LPORT  = \,I10,/,
+   \ RLEN   = \,I10,/,
+   \ SLEN   = \,I10,/,

```

```

+      \ STATUS = \,I10)
101  FORMAT( \ *** Errors: (NETOPN) *** \,/,
+      \ NETID = \,I10,/,
+      \ RNAME = \,9A1,/,
+      \ RPORT = \,I10,/,
+      \ CODE = \,I10,/,
+      \ RTIME = \,I10,/,
+      \ CTIME = \,I10,/,
+      \ STATUS = \,I10)
102  FORMAT( \ *** Errors: (NETCLS) *** \,/,
+      \ NETID = \,I10,/,
+      \ CODE = \,I10,/,
+      \ STATUS = \,I10)

      END PROGRAM TNETOPN

```

Example 3.1 – NETOPN Program Example

3.2.2 NETCLS

Purpose: - This function is used to close a network connection between a local machine and another machine on the network. Upon successful completion, the network connection is closed and the SciSnet control structure is reset to initial conditions. No other functions will operate with the network connection unless NETOPN is again called to re-establish a new network connection.

Fortran Calling Sequence:

```
CALL NETCLS(NETID, CODE, STATUS)
```

Where:

NETID	- Network connection ID 1 thru 100	(input integer)
CODE	- Function code 1 = Purge all messages 2 = Wait for outgoing messages	(input integer)
STATUS	- Returned status 0 = Normal return 1 = Connection not open 2 = Invalid arguments	(output integer)

Notes:

1. NETCLS closes the connection between the local machine and another machine on the network. No further communication is possible for the specified network connection after NETCLS has been called.

Error Conditions:

1. See STATUS parameter definition for all error codes.

Program Example:

```

PROGRAM TNETCLS

!   SciSnet interface definition

USE SCNINF
IMPLICIT NONE

!   Variables Definitions

INTEGER(4) :: NETID, LPORT, RPORT, RLEN, SLEN, CODE, RTIME,
+  CTIME, STATUS, I, LPRNAME
CHARACTER(1),DIMENSION(9) :: RNAME = ('w','i','n','2','0','0',
+  '0','a',' '/')

!*****!
!
!   Initialize Network servers and connections
!
!*****!

!   Start up the network server threads

CODE   = 1
LPORT  = 8001
RLEN   = 1440
SLEN   = 1440
CALL NETSRV(CODE,LPORT,RLEN,SLEN,STATUS)
IF (STATUS.NE.0) THEN
  WRITE(6,100)CODE,LPORT,RLEN,SLEN,STATUS
END IF

!   Open a network socket

NETID  = 1
RNAME(9:9) = CHAR(0)
RPORT  = 8001
CODE   = 0
RTIME  = 0
CTIME  = 0
LPRNAME = LOC(RNAME)
CALL NETOPN(NETID,LPRNAME,RPORT,CODE,RTIME,CTIME,STATUS)
IF (STATUS.NE.0) THEN
  WRITE(6,101)NETID,(RNAME(I),I=1,9),RPORT,CODE,RTIME,CTIME,
+  STATUS
END IF

!*****!
!
!   Close down Network servers and connections
!
!*****!

!   Close down the network connection

code = 2
CALL NETCLS(NETID,CODE,STATUS)

```


1 thru 100

LPLNAME	- Local host name address	(input integer)
LPLADDR	- Local host ip address	(input integer)
LPRNAME	- Remote host name addresss	(input integer)
LPRADDR	- Remote host ip address	(input integer)
NRMSG	- Number of receive messages	(output integer)
NSMSG	- Number of send messages	(output integer)
NRCNT	- Number bytes receive messages	(output integer)
NSCNT	- Number bytes send messages	(output integer)
NRERR	- Number receive message errors	(output integer)
NSERR	- Number send message errors	(output integer)
RERR	- Receive message error	(output integer)
SERR	- Send message error	(output integer)
STATUS	- Returned status	(output integer)

0 = Normal return
1 = Connection not open
2 = Invalid arguments
3 = Transmission errors detected

Notes:

1. The number of receive messages or the receive messages total byte count indicate when the message stream has arrived and can be read using the NETREC function.

Error Conditions:

1. See STATUS parameter definition for all error codes.

Program Example:

```
PROGRAM TNETSTS
!   SciSnet interface definition
USE SCNINF
IMPLICIT NONE
```

```

!      Variables Definitions

      INTEGER(4) :: NETID, LPORT, RPORT, RLEN, SLEN, CODE, RTIME,
+      CTIME, STATUS, I, NRMSG, NSMSG, NRCNT, NSCNT, NRERR, NSERR,
+      RERR, SERR, LPLNAME, LPLADDR, LPRNAME, LPRADDR, LPNAME
      CHARACTER(1),DIMENSION(9) :: NAME = ('w','i','n','2','0','0',
+      '0','a',' ' /)
      CHARACTER(1),DIMENSION(128) :: RNAME,RADDR,LNAME,LADDR

!*****!
!
!      Initialize Network servers and connections
!
!*****!

!      Start up the network server threads

      CODE = 1
      LPORT = 8001
      RLEN = 1440
      SLEN = 1440
      CALL NETSRV(CODE,LPORT,RLEN,SLEN,STATUS)
      IF (STATUS.NE.0) THEN
         WRITE(6,100)CODE,LPORT,RLEN,SLEN,STATUS
      END IF

!      Open a network socket

      NETID = 1
      NAME(9:9) = CHAR(0)
      RPORT = 8001
      CODE = 0
      RTIME = 0
      CTIME = 0
      LPNAME = LOC(NAME)
      CALL NETOPN(NETID,LPNAME,RPORT,CODE,RTIME,CTIME,STATUS)
      IF (STATUS.NE.0) THEN
         WRITE(6,101)NETID,(NAME(I),I=1,9),RPORT,CODE,RTIME,CTIME,
+      STATUS
      END IF

!*****!
!
!      Get Network connection status
!
!*****!

      LPLNAME = LOC(LNAME)
      LPLADDR = LOC(LADDR)
      LPRNAME = LOC(RNAME)
      LPRADDR = LOC(RADDR)
      CALL NETSTS(NETID,LPLNAME,LPLADDR,LPRNAME,LPRADDR,NRMSG,NSMSG,
+      NRCNT,NSCNT,NRERR,NSERR,RERR,SERR,STATUS)
      WRITE(6,102)NETID,(LNAME(I),I=1,40),(LADDR(I),I=1,40),
+      (RNAME(I),I=1,40),(RADDR(I),I=1,40),
+      NRMSG,NSMSG,NRCNT,NSCNT,NRERR,NSERR,
+      RERR,SERR,STATUS

!*****!
!
!      Close down Network servers and connections
!
!*****!

!      Close down the network connection

      code = 2
      CALL NETCLS(NETID,CODE,STATUS)
      IF (STATUS.NE.0) THEN
         WRITE(6,102)NETID,CODE,STATUS

```

```

        END IF

!       Stop the network server threads

        CODE   = 2
        CALL NETSRV(CODE,LPORT,RLEN,SLEN,STATUS)
        IF (STATUS.NE.0) THEN
            WRITE(6,100)CODE,LPORT,RLEN,SLEN,STATUS
        END IF

!       Stop execution

        STOP

!       Format Statements

100    FORMAT(' *** Errors: (NETSRV) *** ',/,
+          '\ CODE   = ',I10,/,
+          '\ LPORT  = ',I10,/,
+          '\ RLEN   = ',I10,/,
+          '\ SLEN   = ',I10,/,
+          '\ STATUS = ',I10)
101    FORMAT(' *** Errors: (NETOPN) *** ',/,
+          '\ NETID  = ',I10,/,
+          '\ RNAME  = ',9A1,/,
+          '\ RPORT  = ',I10,/,
+          '\ CODE   = ',I10,/,
+          '\ RTIME  = ',I10,/,
+          '\ CTIME  = ',I10,/,
+          '\ STATUS = ',I10)
102    FORMAT(' *** Errors: (NETOPN) *** ',/,
+          '\ NETID  = ',I10,/,
+          '\ LNAME  = ',40A1,/,
+          '\ LADDR  = ',40A1,/,
+          '\ RNAME  = ',40A1,/,
+          '\ RADDR  = ',40A1,/,
+          '\ RPORT  = ',I10,/,
+          '\ NRMSG  = ',I10,/,
+          '\ NSMSG  = ',I10,/,
+          '\ NRCNT  = ',I10,/,
+          '\ NSCNT  = ',I10,/,
+          '\ NRERR  = ',I10,/,
+          '\ NSERR  = ',I10,/,
+          '\ RERR  = ',I10,/,
+          '\ SERR  = ',I10,/,
+          '\ STATUS = ',I10)
103    FORMAT(' *** Errors: (NETCLS) *** ',/,
+          '\ NETID  = ',I10,/,
+          '\ CODE   = ',I10,/,
+          '\ STATUS = ',I10)

        END PROGRAM TNETSTS

```

Example 3.3 – NETSTS Program Example

3.2.4 NETSRV

Purpose - This function is used to bring up the SciSnet servers on a local machine. It is the first function to call when initiating network socket communication, and it is the last function to call when terminating network socket communication.

Fortran Calling Sequence:

```
CALL NETSRV(CODE,LPORT,RLEN,SLEN,STATUS)
```

Where:

CODE	- Function code	(input integer)
	1 = Start server 2 = Stop server	
LPORT	- Local server message port	(input integer)
	Valid range 0 to 65535	
RLEN	- Receive socket buffer space	(input integer)
	Use 1440 as a default value	
SLEN	- Send socket buffer space	(input integer)
	Use 1440 as a default value	
STATUS	- Returned status	(output integer)
	0 = Normal return 1 = Server already started 2 = Server already stopped 3 = Invalid arguments 4 = Server error 5 = Sockets open error 6 = Socket error	

Notes:

1. The LPORT variable specified with NETSRV must agree with the RPORT variable specified by the NETOPN on the remote machine. Certain port numbers are public and registered numbers, and must be avoided for use with SciSnet.
2. The RLEN variable sets the receive buffer size used by the socket WinAPI interface. This value can effect the performance of socket transmissions. The default value recommended is 1440.
3. The SLEN variable sets the send buffer size used by the socket WinAPI interface. This value can effect the performance for socket transmissions. The default value recommended is 1440.

Error Conditions:

1. See STATUS parameter definition for all error codes.

Program Example:

```
PROGRAM TNETSRV

!   SciSnet interface definition

USE SCNINF
IMPLICIT NONE

!   Variables Definitions

INTEGER(4) :: NETID, LPORT, RPORT, RLEN, SLEN, CODE, RTIME,
+   CTIME, STATUS, I, LPRNAME
CHARACTER(1), DIMENSION(9) :: RNAME = ('w', 'i', 'n', '2', '0', '0',
+   '0', 'a', ' ')

!*****!
!
!   Initialize Network servers and connections
!
!*****!

!   Start up the network server threads

CODE   = 1
LPORT  = 8001
RLEN   = 1440
SLEN   = 1440
CALL NETSRV(CODE, LPORT, RLEN, SLEN, STATUS)
IF (STATUS.NE.0) THEN
    WRITE(6,100) CODE, LPORT, RLEN, SLEN, STATUS
END IF

!   Open a network socket

NETID  = 1
RNAME(9:9) = CHAR(0)
RPORT  = 8001
CODE   = 0
RTIME  = 0
CTIME  = 0
LPRNAME = LOC(RNAME)
CALL NETOPN(NETID, RNAME, RPORT, CODE, RTIME, CTIME, STATUS)
IF (STATUS.NE.0) THEN
    WRITE(6,101) NETID, (RNAME(I), I=1, 9), RPORT, CODE, RTIME,
+   CTIME, STATUS
END IF

!*****!
!
!   Close down Network servers and connections
!
!*****!

!   Close down the network connection

code = 2
CALL NETCLS(NETID, CODE, STATUS)
IF (STATUS.NE.0) THEN
    WRITE(6,102) NETID, CODE, STATUS
END IF

!   Stop the network server threads

CODE   = 2
CALL NETSRV(CODE, LPORT, RLEN, SLEN, STATUS)
IF (STATUS.NE.0) THEN
```

```

WRITE(6,100)CODE,LPORT,RLEN,SLEN,STATUS
END IF

! Stop execution

STOP

! Format Statements

100 FORMAT(' *** Errors: (NETSRV) *** ',/,
+        '\ CODE = ',I10,/,
+        '\ LPORT = ',I10,/,
+        '\ RLEN = ',I10,/,
+        '\ SLEN = ',I10,/,
+        '\ STATUS = ',I10)
101 FORMAT(' *** Errors: (NETOPN) *** ',/,
+        '\ NETID = ',I10,/,
+        '\ RNAME = ',9A1,/,
+        '\ RPORT = ',I10,/,
+        '\ CODE = ',I10,/,
+        '\ RTIME = ',I10,/,
+        '\ CTIME = ',I10,/,
+        '\ STATUS = ',I10)
102 FORMAT(' *** Errors: (NETCLS) *** ',/,
+        '\ NETID = ',I10,/,
+        '\ CODE = ',I10,/,
+        '\ STATUS = ',I10)

END PROGRAM TNETSRV

```

Example 3.4 – NETSRV Program Example

3.2.5 NETFLS

Purpose: - This function is used to flush the send and receive messages maintained in the SciSnet control structure.

Fortran Calling Sequence:

```
CALL NETFLS(NETID,CODE,STATUS)
```

Where:

- NETID - Network connection ID (input integer)
1 thru 100
- CODE - Function code (input integer)
1 = Flush receive messages
2 = Flush send messages
3 = Flush receive and send messages
- STATUS - Returned status (output integer)

0 = Normal return
 1 = Connection not open
 2 = Invalid arguments

Notes:

1. None.

Error Conditions:

1. See STATUS parameter definition for all error codes.

Program Example:

```

PROGRAM TNETFLS

!   SciSnet interface definition

USE SCNINF
IMPLICIT NONE

!   Variables Definitions

INTEGER(4) :: NETID, LPORT, RPORT, RLEN, SLEN, CODE, RTIME,
+   CTIME, STATUS, I, LPRNAME
CHARACTER(1),DIMENSION(9) :: RNAME = ('w','i','n','2','0','0',
+   '0','a',' ')

!*****!
!
!   Initialize Network servers and connections
!
!*****!

!   Start up the network server threads

CODE   = 1
LPORT  = 8001
RLEN   = 1440
SLEN   = 1440
CALL NETSRV(CODE,LPORT,RLEN,SLEN,STATUS)
IF (STATUS.NE.0) THEN
  WRITE(6,100)CODE,LPORT,RLEN,SLEN,STATUS
END IF

!   Open a network socket

NETID  = 1
RNAME(9:9) = CHAR(0)
RPORT  = 8001
CODE   = 0
RTIME  = 0
CTIME  = 0
LPRNAME = LOC(RNAME)
CALL NETOPN(NETID,RNAME,RPORT,CODE,RTIME,CTIME,STATUS)
IF (STATUS.NE.0) THEN
  WRITE(6,101)NETID,(RNAME(I),I=1,9),RPORT,CODE,RTIME,
+   CTIME,STATUS
END IF

!*****!
!
!   Flush both receive and send messages
!
!*****!

```

```

!
!*****!
CODE = 3
CALL NETFLS (NETID, CODE, STATUS)
IF (STATUS.NE.0) THEN
    WRITE (6,102) NETID, CODE, STATUS
END IF

!*****!
!
!       Close down Network servers and connections
!
!*****!

!       Close down the network connection

code = 2
CALL NETCLS (NETID, CODE, STATUS)
IF (STATUS.NE.0) THEN
    WRITE (6,103) NETID, CODE, STATUS
END IF

!       Stop the network server threads

CODE = 2
CALL NETSRV (CODE, LPORT, RLEN, SLEN, STATUS)
IF (STATUS.NE.0) THEN
    WRITE (6,100) CODE, LPORT, RLEN, SLEN, STATUS
END IF

!       Stop execution

STOP

!       Format Statements

100  FORMAT (' *** Errors: (NETSRV) *** ',/,
+         ' CODE = ',I10,/,
+         ' LPORT = ',I10,/,
+         ' RLEN = ',I10,/,
+         ' SLEN = ',I10,/,
+         ' STATUS = ',I10)
101  FORMAT (' *** Errors: (NETOPN) *** ',/,
+         ' NETID = ',I10,/,
+         ' RNAME = ',9A1,/,
+         ' RPORT = ',I10,/,
+         ' CODE = ',I10,/,
+         ' RTIME = ',I10,/,
+         ' CTIME = ',I10,/,
+         ' STATUS = ',I10)
102  FORMAT (' *** Errors: (NETFLS) *** ',/,
+         ' NETID = ',I10,/,
+         ' CODE = ',I10,/,
+         ' STATUS = ',I10)
103  FORMAT (' *** Errors: (NETCLS) *** ',/,
+         ' NETID = ',I10,/,
+         ' CODE = ',I10,/,
+         ' STATUS = ',I10)

END PROGRAM TNETFLS

```

Example 3.5 – NETFLS Program Example

3.2.6 NETREC

Purpose: - This function is used to transfer a receive message from the SciSnet control structure. A message will be transferred up to limit of buffer size. NETREC will wait the timeout value (if provided with NETOPN) before completion.

Fortran Calling Sequence:

```
CALL NETREC(NETID,LPBUFFER,LENGTH,NBYTES,STATUS)
```

Where:

NETID - Network connection ID (input integer)

1 thru 100

LPBUFFER - Message data buffer (input integer)

LENGTH - Length of message buffer (input integer)

NBYTES - Number bytes received (output integer)

STATUS - Returned status (output integer)

0 = Normal return

1 = Connection not open

2 = Invalid arguments

3 = Message data buffer truncated

4 = Transmission errors detected

5 = Receive timed out

Notes:

1. A receive message will be transferred up to the limits of the buffer length. An error will be indicated if the message will not fit in the user buffer. The SciSnet input thread handles queuing the input receive messages as they arrive.

2. The use of the receive wait timeout option/value in NETOPN allows a user to delay a fixed amount of time, waiting for the arrival of a message from a remote machine.

3. Using an alternate technique, the user may interrogate the status of receive messages before calling NETREC by calling the NETSTS function. With the NETSUS function, a way to delay a fixed amount of time, allows an application to wait for arrival of a message.

Error Conditions:

1. See STATUS parameter definition for all error codes.

Program Example:

```
PROGRAM TNETREC

!   SciSnet interface definition

USE SCNINF
IMPLICIT NONE

!   Variables Definitions

INTEGER(4) :: NETID, LPORT, RPORT, RLEN, SLEN, CODE, RTIME,
+   CTIME, STATUS, I, LENGTH, NBYTES, LPRNAME, LPBUFFER
CHARACTER(1), DIMENSION(9) :: RNAME = ('w', 'i', 'n', '2', '0', '0',
+   '0', 'a', ' ')
CHARACTER(1), DIMENSION(128) :: BUFFER

!*****!
!
!   Initialize Network servers and connections
!
!*****!

!   Start up the network server threads

CODE   = 1
LPORT  = 8001
RLEN   = 1440
SLEN   = 1440
CALL NETSRV(CODE, LPORT, RLEN, SLEN, STATUS)
IF (STATUS.NE.0) THEN
    WRITE(6,100) CODE, LPORT, RLEN, SLEN, STATUS
END IF

!   Open a network socket

NETID  = 1
RNAME(9:9) = CHAR(0)
RPORT  = 8001
CODE   = 0
RTIME  = 0
CTIME  = 0
LPRNAME = LOC(RNAME)
CALL NETOPN(NETID, LPRNAME, RPORT, CODE, RTIME, CTIME, STATUS)
IF (STATUS.NE.0) THEN
    WRITE(6,101) NETID, (RNAME(I), I=1, 9), RPORT, CODE, RTIME,
+   CTIME, STATUS
END IF

!*****!
!
!   Get one receive message
!
!*****!

LENGTH = 128
LPBUFFER = LOC(BUFFER)
CALL NETREC(NETID, LPBUFFER, LENGTH, NBYTES, STATUS)
IF (STATUS.NE.0) THEN
    WRITE(6,102) NETID, LENGTH, NBYTES, STATUS
END IF

!*****!
!
!   Close down Network servers and connections
!
!*****!

!   Close down the network connection
```

```

code = 2
CALL NETCLS (NETID, CODE, STATUS)
IF (STATUS.NE.0) THEN
  WRITE (6, 103) NETID, CODE, STATUS
END IF

!   Stop the network server threads

CODE = 2
CALL NETSRV (CODE, LPORT, RLEN, SLEN, STATUS)
IF (STATUS.NE.0) THEN
  WRITE (6, 100) CODE, LPORT, RLEN, SLEN, STATUS
END IF

!   Stop execution

STOP

!   Format Statements

100  FORMAT (' *** Errors: (NETSRV) *** ',/,
+         ' CODE   = ', I10, /,
+         ' LPORT  = ', I10, /,
+         ' RLEN   = ', I10, /,
+         ' SLEN   = ', I10, /,
+         ' STATUS = ', I10)
101  FORMAT (' *** Errors: (NETOPN) *** ',/,
+         ' NETID  = ', I10, /,
+         ' RNAME  = ', 9A1, /,
+         ' RPORT  = ', I10, /,
+         ' CODE   = ', I10, /,
+         ' RTIME  = ', I10, /,
+         ' CTIME  = ', I10, /,
+         ' STATUS = ', I10)
102  FORMAT (' *** Errors: (NETREC) *** ',/,
+         ' NETID  = ', I10, /,
+         ' LENGTH = ', I10, /,
+         ' NBYTES = ', I10, /,
+         ' STATUS = ', I10)
103  FORMAT (' *** Errors: (NETCLS) *** ',/,
+         ' NETID  = ', I10, /,
+         ' CODE   = ', I10, /,
+         ' STATUS = ', I10)

END PROGRAM TNETREC

```

Example 3.6 – NETREC Program Example

3.2.7 NETSND

Purpose: - This function is used to send a message to a remote machine. NETSND does not wait for the message to be sent, but queues the message to the SciSnet control structure. The output processing thread handles the transmission and verification of complete transfer to the remote machine.

Fortran Calling Sequence:

```
CALL NETSND(NETID, LPBUFFER, NBYTES, STATUS)
```

Where:

NETID	- Network connection ID	(input integer)
	1 thru 100	
LPBUFFER	- Message data buffer address	(input integer)
NBYTES	- Number bytes to send	(input integer)
STATUS	- Returned status	(output integer)
	0 = Normal return	
	1 = Connection not open	
	2 = Invalid arguments	
	3 = Queued memory exhausted	
	4 = Transmission errors detected	

Notes:

1. NETSND does not wait for a message to be sent, but queues the message to the SciSnet control structure. The output processing thread handles the transmission and verification of message transfer to remote machines.
2. The user may interrogate the status of sent messages by using the NETSTS function.
3. The NETSND routine provides the capability to have a timestamp inserted at the beginning of all output messages. NETOPN provides a way to set this timestamp option at the time the network connection is made.

Error Conditions:

1. See STATUS parameter definition for all error codes.

Program Example:

```
PROGRAM TNETSND
!   SciSnet interface definition

USE SCNINF
IMPLICIT NONE

!   Variables Definitions

INTEGER(4) :: NETID, LPORT, RPORT, RLEN, SLEN, CODE, RTIME,
+   CTIME, STATUS, I, NBYTES, LPRNAME, LPBUFFER
CHARACTER(1),DIMENSION(9) :: RNAME = ('w','i','n','2','0','0',
+   '0','a',' ' /)
CHARACTER(28) :: TMSG = ('Test message to a remote host')
CHARACTER(1),DIMENSION(128) :: BUFFER

!*****!
```

```

!
!       Initialize Network servers and connections
!
!*****!
!
!       Start up the network server threads

CODE   = 1
LPORT  = 8001
RLEN   = 1440
SLEN   = 1440
CALL NETSRV (CODE, LPORT, RLEN, SLEN, STATUS)
IF (STATUS.NE.0) THEN
    WRITE (6,100) CODE, LPORT, RLEN, SLEN, STATUS
END IF

!       Open a network socket

NETID  = 1
RNAME (9:9) = CHAR(0)
RPORT  = 8001
CODE   = 0
RTIME  = 0
CTIME  = 0
LPRNAME = LOC (RNAME)
CALL NETOPN (NETID, LPRNAME, RPORT, CODE, RTIME, CTIME, STATUS)
IF (STATUS.NE.0) THEN
    WRITE (6,101) NETID, (RNAME (I), I=1, 9), RPORT, CODE, RTIME,
+   CTIME, STATUS
END IF

!*****!
!
!       Send one message
!
!*****!

NBYTES = 28
DO I = 1, NBYTES
    BUFFER(I) = TMSG(I:I)
END DO
LPBUFFER = LOC (BUFFER)
CALL NETSND (NETID, LPBUFFER, NBYTES, STATUS)
IF (STATUS.NE.0) THEN
    WRITE (6,102) NETID, NBYTES, STATUS
END IF

!*****!
!
!       Close down Network servers and connections
!
!*****!

!       Close down the network connection

code = 2
CALL NETCLS (NETID, CODE, STATUS)
IF (STATUS.NE.0) THEN
    WRITE (6,103) NETID, CODE, STATUS
END IF

!       Stop the network server threads

CODE   = 2
CALL NETSRV (CODE, LPORT, RLEN, SLEN, STATUS)
IF (STATUS.NE.0) THEN
    WRITE (6,100) CODE, LPORT, RLEN, SLEN, STATUS
END IF

!       Stop execution

```

```

        STOP

!      Format Statements

100  FORMAT( ' *** Errors: (NETSRV) *** ',/,
+        ' CODE   = ',I10,/,
+        ' LPORT  = ',I10,/,
+        ' RLEN   = ',I10,/,
+        ' SLEN   = ',I10,/,
+        ' STATUS = ',I10)
101  FORMAT( ' *** Errors: (NETOPN) *** ',/,
+        ' NETID  = ',I10,/,
+        ' RNAME  = ',9A1,/,
+        ' RPORT  = ',I10,/,
+        ' CODE   = ',I10,/,
+        ' RTIME  = ',I10,/,
+        ' CTIME  = ',I10,/,
+        ' STATUS = ',I10)
102  FORMAT( ' *** Errors: (NETSND) *** ',/,
+        ' NETID  = ',I10,/,
+        ' NBYTES = ',I10,/,
+        ' STATUS = ',I10)
103  FORMAT( ' *** Errors: (NETCLS) *** ',/,
+        ' NETID  = ',I10,/,
+        ' CODE   = ',I10,/,
+        ' STATUS = ',I10)

        END PROGRAM TNETSND

```

Example 3.7 – NETSND Program Example

3.2.8 NETDTM

Purpose: - This function is used to get the local time and date from Windows.

Fortran Calling Sequence:

NETDTM(MONTH, DAY, YEAR, HOUR, MINS, SECS, MSECS, STATUS)

Where:

MONTH	- Month (1-12)	(output integer)
DAY	- Day (1-31)	(output integer)
YEAR	- Year	(output integer)
HOUR	- Hour	(output integer)
MINS	- Minutes	(output integer)
SECS	- Seconds	(output integer)
MSECS	- Milliseconds	(output integer)

Notes:

1. The user may use NETDTM to measure the length of time between two events in a user application. Be aware that not all systems provide accurate time.

Error Conditions:

None.

Program Example:

```
PROGRAM TNETDTM

!   SciSnet interface definition

USE SCNINF
IMPLICIT NONE

!   Variables Definitions

INTEGER(4) :: MONTH, DAY, YEAR, HOUR, MINS, SECS, MSECS

!*****!
!
!   Get current date and time
!
!*****!

CALL NETDTM(MONTH, DAY, YEAR, HOUR, MINS, SECS, MSECS)
WRITE(6,100) MONTH, DAY, YEAR, HOUR, MINS, SECS, MSECS

!   Stop execution

STOP

!   Format Statements

100  FORMAT(' Month = ',I10,',',
+        ' Day   = ',I10,',',
+        ' Year  = ',I10,',',
+        ' Hour  = ',I10,',',
+        ' Mins  = ',I10,',',
+        ' Secs  = ',I10,',',
+        ' Msecs = ',I10)

END PROGRAM TNETDTM
```

Example 3.8 – NETDTM Program Example

3.2.9 NETSUS

Purpose: - This function is used to suspend execution of the application for a fixed period of time.

Fortran Calling Sequence:

```
CALL NETSUS(MSECS,STATUS)
```

Where:

MSECS - Milliseconds to suspend for (input integer)

STATUS - Returned status (output integer)

0 = Normal return
1 = Function failed

Notes:

1. An application can use NETSUS to delay execution, waiting for the arrival of input messages without using any cpu cycles.

Error Conditions:

1. See STATUS parameter definition for all error codes.

Program Example:

```
PROGRAM TNETSUS
!   SciSnet interface definition
USE SCINIF
IMPLICIT NONE
!   Variables Definitions
INTEGER(4) :: WAITMS
!*****!
!   Delay execution for 2 seconds
!*****!
WAITMS = 2000
CALL NETSUS(WAITMS,STATUS)
!   Stop execution
STOP
END PROGRAM TNETSUS
```

Example 3.9 – NETSUS Program Example

3.3 Communication Test Program

The communication test program TNET is released as Fortran source code with SciSnet. This program can be compiled and linked with the SciSnet library to produce an executable program. The program allows a user to test network communication through the use of an interactive menu driven program. TNET allows the user to test out and examine the transfer of known user generated data messages.

The functions provided in the SciSnet sockets communication library can be tested through TNET. A socket to another machine may be opened. The network status may be interrogated. The number of messages received or left to send can be viewed by using the status function. Finally, the flush function will delete all messages in the input or output buffers.

3.4 Communication Library

SciSnet is released a Fortran linkable libraries. The SCNDLL.DLL dynamic link library can be dynamically linked by including the SCNIMP.LIB during the LINK process. This SciSnet library is used in conjunction with the standard runtime Fortran library. The user application software makes calls to SciSnet routines described in this manual. At LINK time, the user should include the SciSnet library as one of the libraries along with the Fortran run time library.

Chapter 4

Performance Considerations

SciSnet utilizes TCP/IP under the Microsoft Windows operating system. The sockets WinAPI is the foundation for SciSnet. Data bytes are 8 bits. The underlying protocol is stream based. All data bytes are ordered in the stream. SciSnet has implemented a message based technology upon the sockets WinAPI. Headers are prepended to each message, describing message type and extent. A client/server is implemented on each SciSnet node via NETSRV. Once NETOPN is active, an application can send or receive messages from one node to any other SciSnet node by use of the sockets WinAPI. The only parameters that influence performance are the receive/send socket buffer space variables provided in NETSRV. These values can be adjusted to obtain optimum performance. Usually the values are operating system and medium sensitive.

4.1 Ethernet Controller Measured Data

The SciSnet product was tested by connecting two processors via ethernet controllers. An application on the first processor was executed that sent messages to the second processor. The second processor read in the messages and then sent these same messages back to the first processor. The first processor verified the integrity of every byte and recorded the round trip time. This sequence was repeated 10 times to obtain reasonable statistics. The results of this test were:

Buffer Size (Bytes)	Measured Rate (Bytes/second)	Measured Rate (Bits/second)	Bandwidth Used (%)
200000	9090909	72727272	73
500000	9310987	74487896	75
800000	9523809	76555024	77
1000000	9950249	79601992	80
2000000	10000000	80000000	80
5000000	9988015	79904120	80
8000000	9796718	78373744	78
10000000	9749440	77995520	78

Table 4.1 – Ethernet Controller Measurements

INDEX

I

100 connections, 4

B

buffer length, 17
buffer space, 26

C

client/server, 26
close network
 connection, 6
Connect wait
 timeout, 4
control structure, 3
cpu cycles, 24
CTIME, 4

D

data transfer
 threads, 1
delay execution, 24
dynamic link library,
 3, 25

E

ethernet controllers,
 26

F

flush, 14
Fortran, 3, 25

H

headers, 26

I

IDE, 3
input thread, 17

L

LINK, 25
local area networks,
 1
local time and
 date, 22

M

machine-to-
 machine, 1

Measured Data, 26
message based,
 26
multithreaded, 25

N

name lookup, 1
NETCLS, 6
NETDTM, 22
NETFLS, 14
NETOPN, 3
NETREC, 16
NETSND, 19
NETSRV, 11
NETSTS, 8
NETSUS, 23

O

object files, 3

R

receive buffer size,
 12
receive message, 17
receive messages, 9
receive port, 4
Receive wait
 timeout, 4
registered numbers,
 12

remote host name, 3
RTIME, 4
RUN.BAT, 3
runtime, 25

S

SciSnet, 3
SCNDLL.DLL, 3, 25
SCNIMP.LIB, 3, 25
SCNLIB.LIB, 3, 25
send buffer size, 12
sent messages, 20
static library, 3, 25
suspend execution,
 23

T

TCP/IP, 1
timeout, 17
timestamp, 20
TNET, 25

W

Windows sockets
 API, 1

Software License Agreement

PLEASE READ ALL TERMS AND CONDITIONS OF THIS AGREEMENT PRIOR TO USING THE SOFTWARE RELEASED WITH THIS MANUAL. INSTALLING AND USING THE SOFTWARE INDICATES YOUR ACCEPTANCE OF THESE TERMS AND CONDITIONS. If you do not agree to the terms of this license, return the package to the place of purchase for refund.

LICENSE: - MicroGlyph Systems grants you a nonexclusive, single-user license to use the software on a single computer. The software may be physically transferred to another computer provided the software is used on only one computer at a time. If you wish to use the software on a network server or other multi-user system, you must purchase the same number of copies of the software as users attached to the network or terminals attached to the multi-user system.

COPYRIGHT: - The software and enclosed documentation are copyrighted. You may make one copy of the software for back-up purposes only, provided the MicroGlyph Systems copyright notice is included on the back-up copy., and the back-up copy is not installed on any other computer. You agree to use reasonable efforts to prevent unauthorized copying of the software or documentation.

THE SOFTWARE AND DOCUMENTATION MAY NOT BE RENTED, LEASED, COPIED, MODIFIED, OR TRANSFERRED EXCEPT AS EXPRESSLY PROVIDED IN THIS AGREEMENT.

LIMITED WARRANTY: - MicroGlyph Systems warrants that the media on which the software is supplied shall be free from defects in materials and workmanship under normal use for a period of 90 days from the date of purchase. MicroGlyph Systems' entire liability and your exclusive remedy under this warranty shall be replacement of the defective media when returned to MicroGlyph Systems accompanied by a copy of the original invoice. MicroGlyph Systems shall have no obligation to replace the media if MicroGlyph Systems determines failure has resulted from accident, abuse, or neglect.

EXCLUSION OF WARRANTIES AND LIMITATION OF DAMAGES: - THE SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS", AND MICROGLYPH SYSTEMS DISCLAIMS ALL EXPRESS AND IMPLIED WARRANTIES (EXCEPT THE LIMITED WARRANTY ON MEDIA STATED ABOVE), INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR PARTICULAR PURPOSE. MICROGLYPH SYSTEMS DOES NOT WARRANT THAT THE SOFTWARE LIBRARY WILL MEET YOUR REQUIREMENTS OR THAT ITS OPERATION WILL BE UNINTERRUPTED OR ERROR FREE. YOU ASSUME RESPONSIBILITY FOR THE SELECTION OF THE SOFTWARE AND FOR INSTALLATION, USE, AND RESULTS OBTAINED FROM THE SOFTWARE. MICROGLYPH SYSTEMS SHALL NOT BE LIABLE FOR ANY LOST PROFITS OR ANY SPECIAL CONSEQUENTIAL, OR INDIRECT DAMAGES EVEN IF MICROGLYPH SYSTEMS HAS BEEN ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH DAMAGES.

Some states do not allow the exclusion of implied warranties, so the above exclusion PLEASE READ ALL TERMS AND CONDITIONS OF THIS AGREEMENT PRIOR TO

USING THE SOFTWARE RELEASED WITH THIS MANUAL. INSTALLING AND USING THE SOFTWARE (INCLUDING FUTURE SOFTWARE UPDATES) INDICATES YOUR ACCEPTANCE OF THESE TERMS AND CONDITIONS.

If you do not agree to the terms of this license, return the package to the place of purchase for refund.

LICENSE: - MicroGlyph Systems grants you a nonexclusive, single-user license to use the software on a single computer. The software may be physically transferred to another computer provided the software is used on only one computer at a time. If you wish to use the software on a network server or other multi-user system, you must purchase the same number of copies of the software as users attached to the network or terminals attached to the multi-user system. You may elect to purchase from MicroGlyph Systems additional, unassigned licenses (copies) as the need may arise.

COPYRIGHT: - The software and enclosed documentation are copyrighted. You may make one copy of the software for back-up purposes only, provided the MicroGlyph Systems copyright notice is included on the back-up copy, and the back-up copy is not installed on any other computer. You agree to use reasonable efforts to prevent unauthorized copying of the software or documentation.

THE SOFTWARE AND DOCUMENTATION MAY NOT BE RENTED, LEASED, COPIED, MODIFIED, OR TRANSFERRED EXCEPT AS EXPRESSLY PROVIDED IN THIS AGREEMENT OR BY AGREEMENT OF MICROGLYPH SYSTEMS.

LIMITED WARRANTY: - MicroGlyph Systems warrants that the media on which the software is supplied shall be free from defects in materials and workmanship under normal use for a period of 90 days from the date of purchase. MicroGlyph Systems' entire liability and your exclusive remedy under this warranty shall be replacement of the defective media when returned to MicroGlyph Systems accompanied by a copy of the original invoice. MicroGlyph Systems shall have no obligation to replace the media if MicroGlyph Systems determines failure has resulted from accident, abuse, or neglect.

EXCLUSION OF WARRANTIES AND LIMITATION OF DAMAGES: - THE SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS", AND MICROGLYPH SYSTEMS DISCLAIMS ALL EXPRESS AND IMPLIED WARRANTIES (EXCEPT THE LIMITED WARRANTY ON MEDIA STATED ABOVE), INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR PARTICULAR PURPOSE. MICROGLYPH SYSTEMS DOES NOT WARRANT THAT THE SOFTWARE LIBRARY WILL MEET YOUR REQUIREMENTS OR THAT ITS OPERATION WILL BE UNINTERRUPTED OR ERROR FREE. YOU ASSUME RESPONSIBILITY FOR THE SELECTION OF THE SOFTWARE AND FOR INSTALLATION, USE, AND RESULTS OBTAINED FROM THE SOFTWARE. MICROGLYPH SYSTEMS SHALL NOT BE LIABLE FOR ANY LOST PROFITS OR ANY SPECIAL CONSEQUENTIAL, OR INDIRECT DAMAGES EVEN IF MICROGLYPH SYSTEMS HAS BEEN ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH DAMAGES.

Some states do not allow the exclusion of implied warranties, so the above exclusion may not apply to you. Some states do not allow the limitation or exclusion of liability for incidental or consequential damages, so the limitation or exclusion may not apply to you.

The limited warranty stated above gives you specific legal rights, and you may also have other rights which vary from state to state.

TERMINATION: - This license agreement is effective until terminated. MicroGlyph Systems reserves the right to terminate any software license at any time for cause provided adequate notice is given. You may terminate the agreement by destroying all copies of the software and documentation. The license will also terminate if you fail to comply with the terms and conditions of this agreement. You agree upon termination to destroy all copies of the software and documentation. Failure to do so will in no way extend any warranties, expressed or implied, beyond the license termination. By failing to destroy any copies of the software upon any termination of this license, the user-purchaser agrees to waive any remedies that may exist under any applicable state or federal laws, and include any adaptation of the UCC (uniform commercial code).

UPDATES: - From time to time MicroGlyph Systems may offer, at extra charge, updates to the software or documentation. In order for you to obtain or be advised of such updates, the enclosed Software License Registration form must be completed and returned to MicroGlyph Systems within 10 days of purchase.

EXPORT RESTRICTIONS: - You agree not to export the software and documentation from, or re-import them to the country in which you purchased them without first obtaining (1) all licenses and permits required by the appropriate regulatory authorities, including those of the United States, if applicable, and (2) MicroGlyph Systems' written permission to do so. MicroGlyph Systems assumes no liability for purchaser's failure to comply with any state or federal laws pertaining to usage of software products.

GENERAL: - By installing and using the software, you acknowledge that you have read this agreement, understand it, and agree to be bound by it. You further agree that it is the entire agreement between MicroGlyph Systems and you, that it supersedes any oral or written proposal or prior agreement, and that it may not be modified except in writing signed by both MicroGlyph Systems and you.

This agreement shall be governed by the laws of the State of Massachusetts. For the purposes of this license, the term "software" includes, but is not limited to, any product purchased from MicroGlyph Systems for use in a computer of any type that is designed to accept such products. The term "software" includes any products referred to as "updates" or "future updates." This license is applicable to all MicroGlyph Systems software products and extends to any documentation of the product.

Every individual copy of "software" is covered in whole by this software license, to include any copies made for back-up purposes. Failure to pay the full purchase price of the software does not in any way release the user of the product from the terms of this license, i.e., "pirated" versions shall automatically be subject to the terms and liabilities of this license.

Any documentation products attached to any software are proprietary products of MicroGlyph Systems information within these products is subject to change without notice, and MicroGlyph Systems assumes no liability for any errors that may appear in these documents.

Software License Registration

SciSnet Version 7.0

Name : _____

Company : _____

Address : _____

Address : _____

City : _____ State/Province : _____

Postal Code : _____ Country : _____

Voice : _____ Fax : _____

Email : _____

Purchase Date : _____

Mail To : MicroGlyph Systems
P.O. Box 474
Lexington, MA 02420-0005

Voice : (781) 861-0426
Fax : (781) 674-1179
Email : support@microglyph.com
Web : http://www.microglyph.com

