

# **MicroGlyph/SciComm™**

A Computer Communication Library

## **Programming Reference Manual**

Release 7.0  
Copyright © 1985-2006  
by 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](mailto:support@microglyph.com)  
Web: <http://www.microglyph.com>

## **PREFACE**

This programming reference manual provides technical information for Fortran programming development using the SciComm Communication library. SciComm supports PC compatible computers with RS232C serial controllers under the Microsoft Windows operating systems. SciComm supports up to twelve serial ports. SciComm's use of input and output buffering allows simultaneous two-way communication for asynchronous real time application development. A variety of baud rates are supported from 110 to 115,200 with flow control provided by software XON/XOFF, or by DTR/DSR, RTS/CTS hardware signaling. SciComm applications can be created and execute under the Microsoft Windows operating system as stand-alone applications. The SciComm library is available in versions that are compatible with a wide variety of Fortran compilers. This distribution includes the SciComm Communication library with a reference manual, an example source program, and help files.

## **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

<b>Tables</b> .....	iii
<b>Examples</b> .....	iii
<b>CHAPTER 1 INTRODUCTION</b> .....	1
1.1 GENERAL INFORMATION.....	1
1.2 SYSTEM REQUIREMENTS .....	1
<b>CHAPTER 2 COMMUNICATION ENVIRONMENT</b> .....	2
<b>CHAPTER 3 PROGRAMMING REFERENCE</b> .....	3
3.1 COMMUNICATION LIBRARY .....	3
3.2 COMMUNICATION FUNCTIONS .....	3
3.2.1 COMOPN.....	3
3.2.2 COMCLS.....	6
3.2.3 COMSTS .....	8
3.2.4 COMCTL .....	11
3.2.5 COMFLS.....	14
3.2.6 COMREC.....	15
3.2.7 COMSND.....	18
3.2.8 COMDTM.....	20
3.2.9 COMSUS .....	22
3.3 COMMUNICATION TEST PROGRAM .....	24
3.4 COMMUNICATION LIBRARY .....	24
<b>CHAPTER 4 PERFORMANCE CONSIDERATIONS</b> .....	25
4.1 RS232C CONTROLLER MEASURED DATA .....	25
<b>CHAPTER 5 HARDWARE INTERFACES</b> .....	26
<b>INDEX</b> .....	28
<b>Software License Agreement</b> .....	29
<b>Software License Registration</b> .....	32
<b>Software Report</b> .....	33

## Tables

Table 4.1 – Serial Controller Measurements .....	25
Table 5.1 – D9/D9 Connector Cable .....	26
Table 5.2 – D25/D25 Connector Cable .....	26
Table 5.3 – D9/D25 Connector Cable .....	27

## Examples

Example 3.1 – COMOPN Program Example .....	6
Example 3.2 – COMCLS Program Example.....	8
Example 3.3 – COMSTS Program Example.....	11
Example 3.4 – COMCTL Program Example.....	13
Example 3.5 – COMFLS Program Example .....	15
Example 3.6 – COMREC Program Example .....	18
Example 3.7 – COMSND Program Example .....	20
Example 3.8 – COMDTM Program Example .....	22
Example 3.9 – COMSUS Program Example.....	23

# Chapter 1

## Introduction

SciComm provides a Fortran communications library. The functions provide two way communications over standard RS-232C serial ports with any asynchronous communications device. SciComm utilizes native Windows API drivers that maintain separate input and output buffers and allow parallel asynchronous two-way communications with data integrity. A range of baud rates is provided from 110 to 115,200. Flow control is provided through software XON/XOFF characters, or by hardware DTR/DSR or RTS/CTS control line signaling.

### 1.1 General Information

SciComm was designed with the philosophy that high speed data input can be obtained with system level drivers which buffer each character as it arrives and saves it for later retrieval by the application program. Normally, no data will be lost while an application is processing previous read character strings from the system buffers. The exception occurs when the CPU is heavily taxed and not able to keep up with the input data rate. Under such circumstance data overrun errors can occur. The user must trade off transfer rate for lost data errors.

### 1.2 System Requirements

SciComm was designed for PC compatible computers under the Microsoft Windows operating systems and serial or asynchronous communications adapters. Communication ports com1 through com12 are supported by the hardware manufacturer's native Windows communication drivers. The full capabilities of RS232C serial controllers are supported in SciComm.

## Chapter 2

### Communication Environment

Before communication can take place on any serial port, a driver must be initialized for the specified device. The Windows operating system performs this task at the time the system comes up for all serial ports attached to the computer. The operating system determines if the device is available and ready. Devices that compete for usage on these ports are typically the mouse and the modem. SciComm provides the capability to connect to these serial ports through the use of the COMOPN routine. Ports can be disconnected by using the COMCLS routine. At the time the serial port is opened by COMOPN, the communication baud rate 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 serial communication functions provided by the SciComm library. The various subsections included provide all the communication functions representative of a full communication system. The system communication driver has been described in the communication environment section (previous section.) Routines to open and close communication paths, flush buffers, report on hardware status, and do the serial i/o are included in the SciComm communication library.

#### 3.1 Communication Library

SciComm routines are release in two forms. There is the SCCLIB.LIB static library that can be linked with the application to produce an executable file. There is also the SCCIMP.LIB import library which is used with the SCCDLL.DLL dynamic link library to dynamically link an application to produce an executable file. The executable produced by linking to the DLL is usually smaller in the amount of disk space used. 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 information.

#### 3.2 Communication Functions

SciComm communication routines are contained in the SCCLIB.LIB and SCCDLL.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. There is usually an integrated development environment (IDE) also provided with the compiler. The following sections describe the serial communication functions and their Fortran calling sequences.

##### 3.2.1 COMOPN

Purpose: - This function is used to perform the initial open of the communication port. A search for the communication driver on the specified port is made. The serial hardware is initialized to the mode selected and flow control requested.

Fortran Calling Sequence:

```
CALL COMOPN(PORT,BAUD,PARITY,STPBTS,CHRLen,FLOW,  
            RLEN,SLEN,STATUS)
```

Where:

PORT	- Communication port ID	(INPUT INTEGER)
	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	- Baud rate	(INPUT INTEGER)
	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	
	14 = 128000	
	15 = 256000	
PARITY	- Parity marking	(INPUT INTEGER)
	1 = None	
	2 = Odd	
	3 = Even	
	4 = Mark	
	5 = Space	
STPBTS	- Stop bits per character	(INPUT INTEGER)
	1 = 1	
	2 = 1.5	
	3 = 2	
CHRLLEN	- Number bits per character	(INPUT INTEGER)

- 1 = 4 bits/character
- 2 = 5 bits/character
- 3 = 6 bits/character
- 4 = 7 bits/character
- 5 = 8 bits/character

FLOW - Flow control code (INPUT INTEGER)

- 1 = None
- 2 = XON/XOFF (software)
- 3 = DTR/DSR (hardware)
- 4 = RTS/CTS (hardware)

RLEN - System input buffer size (INPUT INTEGER)

Default 8192 bytes

SLEN - System output buffer size (INPUT INTEGER)

Default 8192 bytes

STATUS - Returned status (OUTPUT INTEGER)

- 0 = Normal return
- 1 = Port already open
- 2 = Port not supported
- 3 = Invalid arguments
- 4 = Port not found
- 5 = Port access denied
- 6 = Port open error
- 7 = Thread startup error

Notes:

1. COMOPN establishes the logical connection between a communication port and the serial hardware associated with the port. Communication protocols are established.

2. To establish a new set of communication protocols it is necessary to call COMCLS first (if port is open), before calling COMOPN again.

3. XON/XOFF flow control uses the XON(11H) and XOFF(13H) characters to control input and output. Whenever XOFF is received at the PC, no more output will be send out from the PC until an XON is received to reestablish data flow. Whenever the input buffer is within overflowing, an XOFF character will be sent out to stop additional data from coming until the user has retrieved the input data. An XON character will be sent when the input buffer is close to being empty after a previous full condition. No character checking is performed on output data streams.

## Error Conditions:

1. See status parameter definition for all error codes.

## Program Example:

```
PROGRAM TCOMOPN
!
!   SciComm interface definition
!
USE SCCINF
IMPLICIT NONE
!
!   Variables definitions
!
INTEGER(4) :: PORT, BAUD, PARITY, STPBTS, CHRLen, FLOW, STATUS,
             RLEN, SLEN, CODE
!
!   Initialize COM2 port: 9600 baud, no parity, 1 stop bit,
!   8 bits, XON/XOFF disabled
!
PORT   = 2
BAUD   = 7
PARITY = 1
STPBTS = 1
CHRLen = 5
FLOW   = 1
RLEN   = 8192
SLEN   = 8192
!
!   Open COM2 port
!
CALL COMOPN(PORT,BAUD,PARITY,STPBTS,CHRLen,FLOW,RLEN,SLEN,STATUS)
IF (STATUS.NE.0) THEN
    WRITE(6,100)STATUS
END IF
!
!   Close COM2 port
!
CODE = 1
CALL COMCLS(PORT,CODE,STATUS)
IF (STATUS.NE.0) THEN
    WRITE(6,101)STATUS
END IF
!
!   Exit processing
!
STOP
!
!   FORMAT Statements
!
100  FORMAT(' *** Errors (COMOPN):',I4,' ***')
101  FORMAT(' *** Errors (COMCLS):',I4,' ***')
!
END PROGRAM TCOMOPN
```

### Example 3.1 – COMOPN Program Example

## 3.2.2 COMCLS

Purpose: - This function is used to close the communication port. The serial

hardware is reset to a disabled state and the driver control structure is set to an initial state. The Windows serial driver remains attached to the communication port but the port is marked closed (not open). No other functions will operate with the port unless COMOPN is again called for the port.

Fortran Calling Sequence:

```
CALL COMCLS(PORT, CODE, STATUS)
```

Where:

PORT - Communication port ID (INPUT INTEGER)

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

CODE - Function code (INPUT INTEGER)

1 = Purge all messages  
2 = Wait for outgoing messages

STATUS - Returned status (OUTPUT INTEGER)

0 = Normal return  
1 = Port not open  
2 = Port not supported  
3 = Invalid arguments  
4 = Thread shutdown error

Notes:

1. COMCLS breaks the logical connection between a communication port and the serial hardware associated with the port through the Windows communication driver. No further communication can take place after COMCLS has been called, although the Windows communication driver still remains present in a disabled state. The Windows communication driver can still be outputting bytes, even though the number of bytes in the output buffer is reported as zero. It is a good practice to delay a good amount of time to allow all the bytes to be received on the other end of the serial line, before calling COMCLS.

Error Conditions:

1. See status parameter definition for all error codes.

#### Program Example:

```
PROGRAM TCOMCLS
!
!   SciComm interface definition
!
USE SCCINF
IMPLICIT NONE
!
!   Variables definitions
!
INTEGER(4) :: PORT, BAUD, PARITY, STPBTS, CHRLEN, FLOW, STATUS,
            RLEN, SLEN, CODE
!
!   Initialize COM2 port: 9600 baud, no parity, 1 stop bit,
!   8 bits, XON/XOFF disabled
!
PORT = 2
BAUD = 7
PARITY = 1
STPBTS = 1
CHRLEN = 5
FLOW = 1
RLEN = 8192
SLEN = 8192
!
!   Open COM2 port
!
CALL COMOPN(PORT, BAUD, PARITY, STPBTS, CHRLEN, FLOW, RLEN, SLEN, STATUS)
IF (STATUS.NE.0) THEN
    WRITE(6,100)STATUS
END IF
!
!   Close COM2 port
!
CODE = 1
CALL COMCLS(PORT, CODE, STATUS)
IF (STATUS.NE.0) THEN
    WRITE(6,101)STATUS
END IF
!
!   Exit processing
!
STOP
!
!   FORMAT Statements
!
100 FORMAT(' *** Errors (COMOPN):', I4, ' ***')
101 FORMAT(' *** Errors (COMCLS):', I4, ' ***')
!
END PROGRAM TCOMCLS
```

### Example 3.2 – COMCLS Program Example

#### 3.2.3 COMSTS

Purpose: - This function is used to read or report on the status registers from the

serial communication port hardware and to collect the various control counters from the Windows communication driver data structures.

Fortran Calling Sequence:

```
CALL COMSTS(PORT,LSREG,MSREG,ERRSTS,RECCNT,SNDCNT,  
            RECLEN,SNDDLEN,STATUS)
```

Where:

PORT - Communication port ID (INPUT INTEGER)

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 - Line Status register bits (OUTPUT INTEGER)

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 - Modem Status register bits (OUTPUT INTEGER)

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

ERRSTS - Error status (OUTPUT INTEGER)

Accumulated error LSREG bits since last call

RECCNT	- No. bytes in receive buffer	(OUTPUT INTEGER)
SNDCNT	- No. bytes in send buffer	(OUTPUT INTEGER)
RECLEN	- Length of receive buffer	(OUTPUT INTEGER)
SNDLEN	- Length of send buffer	(OUTPUT INTEGER)
STATUS	- Returned status	(OUTPUT INTEGER)

0 = Normal return  
 1 = Port not open  
 2 = Port not supported

Notes:

1. The receive buffer count can indicate when the application's data stream has arrived and can be read using COMREC.

Error Conditions:

1. See status parameter definition for all error codes.

Program Example:

```

PROGRAM TCOMSTS
!
!   SciComm interface definition
!
USE SCCINF
IMPLICIT NONE
!
!   Variables definitions
!
INTEGER(4) :: PORT, BAUD, PARITY, STPBTS, CHRLEN, FLOW, STATUS,
&  LSREG, MSREG, ERRSTS, RECCNT, SNDCNT, RECLEN, SNDLEN, RLEN,
&  SLEN, CODE
!
!   Initialize COM2 port: 9600 baud, no parity, 1 stop bit,
!   8 bits, XON/XOFF disabled
!
PORT   = 2
BAUD   = 7
PARITY = 1
STPBTS = 1
CHRLEN = 5
FLOW   = 1
RLEN   = 8192
SLEN   = 8192
!
!   Open COM2 port
!
CALL COMOPN(PORT, BAUD, PARITY, STPBTS, CHRLEN, FLOW, RLEN, SLEN, STATUS)
IF (STATUS.NE.0) THEN
  WRITE(6,100)STATUS
END IF
!
!   Check status of hardware
!
CALL COMSTS(PORT, LSREG, MSREG, ERRSTS, RECCNT, SNDCNT,
&           RECLEN, SNDLEN, STATUS)

```

```

        IF (STATUS.NE.0) THEN
            WRITE(6,101)STATUS
        END IF

!       Close COM2 port

        CODE = 1
        CALL COMCLS(PORT,CODE,STATUS)
        IF (STATUS.NE.0) THEN
            WRITE(6,102)STATUS
        END IF

!
!       Exit processing
!
        STOP

!
!       FORMAT Statements
!
100    FORMAT(' *** Errors (COMOPN):',I4,' ***')
101    FORMAT(' *** Errors (COMSTS):',I4,' ***')
102    FORMAT(' *** Errors (COMCLS):',I4,' ***')
!
        END PROGRAM TCOMSTS

```

**Example 3.3 – COMSTS Program Example**

**3.2.4 COMCTL**

Purpose - This function is used to set communication control parameters.

Fortran Calling Sequence:

```
CALL COMCTL(PORT,CODE,VALUE,BUFFER,NBYTES,STATUS)
```

Where:

- |      |                         |                 |
|------|-------------------------|-----------------|
| PORT | - Communication port ID | (INPUT INTEGER) |
|      | 1 = 'COM1'              |                 |
|      | 2 = 'COM2'              |                 |
|      | 3 = 'COM3'              |                 |
|      | 4 = 'COM4'              |                 |
|      | 5 = 'COM5'              |                 |
|      | 6 = 'COM6'              |                 |
|      | 7 = 'COM7'              |                 |
|      | 8 = 'COM8'              |                 |
|      | 9 = 'COM9'              |                 |
|      | 10 = 'COM10'            |                 |
|      | 11 = 'COM11'            |                 |
|      | 12 = 'COM12'            |                 |
| CODE | - Control function code | (INPUT INTEGER) |

- 1 = Data-terminal-ready line
- 2 = Request-to-send line
- 3 = Line-break signal
- 4 = Set write timestamp option
- 5 = Set write no-wait queued option
- 6 = Set read with timeout value
- 7 = Set read with timeout value and delimiter string

VALUE	- Binary value	(INPUT/OUTPUT INTEGER)
	0 = Turn off	[code=1-5]
	1 = Turn on	[code=1-5]
	x = Timeout value (ms)	[code=6-7]
BUFFER	- Read delimiter string	(INPUT CHARACTER)
NBYTES	- Number bytes in string	(INPUT INTEGER)
	Must be > 0 and <= 32	
STATUS	- Returned status	(OUTPUT INTEGER)
	0 = Normal return	
	1 = Port not open	
	2 = Port not supported	
	3 = Invalid argument	

Notes:

1. The data-terminal-ready, request-to-send, and the line break optional settings allow customized user defined protocols to be implemented by control of these hardware lines.
2. The write with timestamp option causes the system local date and time to be formatted into a string [ <D:MMDDYYHHMMSSMM:> ] which is output at the beginning of each COMSND output stream.
3. The write no-wait queued option causes each COMSND output buffer to be moved to system global storage and to be queued for output by a separate processing thread. The user application should be sure to process the returned status variable from COMSND. This will indicate any out-of-memory conditions which can result from use of this option.
4. The read with timeout, and delimiter option allows the user application to do controlled waits for incoming data streams.

Error Conditions:

1. See status parameter definition for all error codes.

Program Example:

```

PROGRAM TCOMCTL
!
!   SciComm interface definition
!
USE SCCINF
IMPLICIT NONE
!
!   Variables definitions
!
INTEGER(4) :: PORT, BAUD, PARITY, STPBTS, CHRLEN, FLOW, STATUS,
& CODE, VALUE, NBYTES, RLEN, SLEN
CHARACTER(1),DIMENSION(80) :: BUFFER
!
!   Initialize COM2 port: 9600 baud, no parity, 1 stop bit,
!   8 bits, XON/XOFF disabled
!
PORT   = 2
BAUD   = 7
PARITY = 1
STPBTS = 1
CHRLEN = 5
FLOW   = 1
RLEN   = 8192
SLEN   = 8192
!
!   Open COM2 port
!
CALL COMOPN(PORT,BAUD,PARITY,STPBTS,CHRLEN,FLOW,RLEN,SLEN,STATUS)
IF (STATUS.NE.0) THEN
    WRITE(6,100)STATUS
END IF
!
!   Turn on write with timestamp option
!
code = 4
value = 1
CALL COMCTL(PORT,CODE,VALUE,BUFFER,NBYTES,STATUS)
IF (STATUS.NE.0) THEN
    WRITE(6,101)STATUS
END IF
!
!   Close COM2 port
!
CODE = 1
CALL COMCLS(PORT,CODE,STATUS)
IF (STATUS.NE.0) THEN
    WRITE(6,102)STATUS
END IF
!
!   Exit processing
!
STOP
!
!   FORMAT Statements
!
100 FORMAT(' *** Errors (COMOPN):',I4,' ***')
101 FORMAT(' *** Errors (COMCTL):',I4,' ***')
102 FORMAT(' *** Errors (COMCLS):',I4,' ***')
!
END PROGRAM TCOMCTL

```

### Example 3.4 – COMCTL Program Example

### 3.2.5 COMFLS

Purpose: - This function is used to flush the send and receive buffers in the Windows communication drivers.

Fortran Calling Sequence:

```
CALL COMFLS(PORT, CODE, STATUS)
```

Where:

PORT	- Communication port ID	(INPUT INTEGER)
	1 = 'COM1'	
	2 = 'COM2'	
	3 = 'COM3'	
	4 = 'COM4'	
	5 = 'COM5'	
	6 = 'COM6'	
	7 = 'COM7'	
	8 = 'COM8'	
	9 = 'COM9'	
	10 = 'COM10'	
	11 = 'COM11'	
	12 = 'COM12'	
CODE	- Buffer flush code	(INPUT INTEGER)
	1 = Flush receive buffer	
	2 = Flush send buffer	
	3 = Flush receive and send buffer	
STATUS	- Returned status	(OUTPUT INTEGER)
	0 = Normal return	
	1 = Port not open	
	2 = Port not supported	
	3 = Invalid arguments	

Notes:

1. None.

Error Conditions:

1. See status parameter definition for all error codes.

Program Example:

```
PROGRAM TCOMFLS
!
```

```

!      SciComm interface definition
!
      USE SCCINF
      IMPLICIT NONE
!
!      Variables definitions
!
      INTEGER(4) :: PORT, BAUD, PARITY, STPBTS, CHRLEN, FLOW, STATUS,
&      CODE, RLEN, SLEN
!
!      Initialize COM2 port: 9600 baud, no parity, 1 stop bit,
!      8 bits, XON/XOFF disabled
!
      PORT   = 2
      BAUD   = 7
      PARITY = 1
      STPBTS = 1
      CHRLEN = 5
      FLOW   = 1
      RLEN   = 8192
      SLEN   = 8192
!
!      Open COM2 port
!
      CALL COMOPN(PORT,BAUD,PARITY,STPBTS,CHRLEN,FLOW,RLEN,SLEN,STATUS)
      IF (STATUS.NE.0) THEN
        WRITE(6,100)STATUS
      END IF
!
!      Flush output buffer
!
      CODE = 2
      CALL COMFLS(PORT,CODE,STATUS)
      IF (STATUS.NE.0) THEN
        WRITE(6,101)STATUS
      END IF
!
!      Close COM2 port
!
      CODE = 1
      CALL COMCLS(PORT,CODE,STATUS)
      IF (STATUS.NE.0) THEN
        WRITE(6,102)STATUS
      END IF
!
!      Exit processing
!
      STOP
!
!      FORMAT Statements
!
100  FORMAT(' *** Errors (COMOPN):',I4,' ***')
101  FORMAT(' *** Errors (COMFLS):',I4,' ***')
102  FORMAT(' *** Errors (COMCLS):',I4,' ***')
!
      END PROGRAM TCOMFLS

```

### Example 3.5 – COMFLS Program Example

#### 3.2.6 COMREC

Purpose: - This function is used to transfer data from the receive buffer of the

Windows communication driver to the user's buffer. Data will be transferred up to limits of user's buffer area.

Fortran Calling Sequence:

```
CALL COMREC(PORT,BUFFER,LENGTH,LSREG,MSREG,NBYTES,  
            STATUS)
```

Where:

PORT - Communication port ID (INPUT INTEGER)

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

BUFFER - Receive data buffer (OUTPUT CHARACTER)

LENGTH - Length of receive buffer (INPUT INTEGER)

LSREG - Line Status register bits (OUTPUT INTEGER)

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 - Modem Status register bits (OUTPUT INTEGER)

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

NBYTES	- Number bytes transferred	(OUTPUT INTEGER)
STATUS	- Returned status	(OUTPUT INTEGER)
	0 = Normal return	
	1 = Port not open	
	2 = Port not supported	
	3 = Invalid arguments	
	4 = Transmission errors since last call, see LSREG bits for errors	
	5 = Delimiter not detected	
	6 = Read timed out	

Notes:

1. Receive data will be transferred up to the limits of the user's buffer length. No errors will be indicated if all of the data will not fit. On the next call to COMREC, the user can get the remaining data or any portion thereof. The length parameter provides a formatting capability by allowing the user to specify the read window.
2. The user may interrogate the status of the receive buffer before calling COMREC by calling COMSTS. Be aware that more data may come in between these two events.

Error Conditions:

1. See status parameter definition for all error codes.

Program Example:

```

PROGRAM TCOMREC
!
!   SciComm interface definition
!
USE SCCINF
IMPLICIT NONE
!
!   Variables definitions
!
INTEGER(4) :: PORT, BAUD, PARITY, STPBTS, CHRLEN, FLOW, STATUS,
&  LSREG, MSREG, ERRSTS, RECCNT, SNDCNT, RECLEN, SNDLEN, RLEN,
&  SLEN, CODE, NBYTES, LENGTH = 80
CHARACTER(1),DIMENSION(80) :: BUFFER
!
!   Initialize COM2 port: 9600 baud, no parity, 1 stop bit,
!   8 bits, XON/XOFF disabled
!
PORT   = 2
BAUD   = 7
PARITY = 1
STPBTS = 1
CHRLEN = 5
FLOW   = 1
RLEN   = 8192
SLEN   = 8192
!
!   Open COM2 port
!
CALL COMOPN(PORT, BAUD, PARITY, STPBTS, CHRLEN, FLOW, RLEN, SLEN, STATUS)

```

```

        IF (STATUS.NE.0) THEN
            WRITE(6,100)STATUS
        END IF
!
!   Read a character string
!
        CALL COMREC(PORT,BUFFER,LENGTH,LSREG,MSREG,NBYTES,STATUS)
        IF (STATUS.NE.0) THEN
            WRITE(6,101)STATUS
        END IF
!
!   Close COM2 port
!
        CODE = 1
        CALL COMCLS(PORT,CODE,STATUS)
        IF (STATUS.NE.0) THEN
            WRITE(6,102)STATUS
        END IF
!
!   Exit processing
!
        STOP
!
!   FORMAT Statements
!
100    FORMAT(' *** Errors (COMOPN):',I4,' ***')
101    FORMAT(' *** Errors (COMREC):',I4,' ***')
102    FORMAT(' *** Errors (COMCLS):',I4,' ***')
!
        END PROGRAM TCOMREC

```

**Example 3.6 – COMREC Program Example**

**3.2.7 COMSND**

Purpose: - This function is used to transfer the contents of the user's buffer to the communication output buffer. Data will begin to be transmitted immediately, a character at a time, until all data has been sent out the serial port. This happens complete asynchronously to the user's program execution. Computer processing can proceed during the time the data is going out.

Fortran Calling Sequence:

```
CALL COMSND(PORT,BUFFER,NBYTES,STATUS)
```

Where:

- |      |                         |                 |
|------|-------------------------|-----------------|
| PORT | - Communication port ID | (INPUT INTEGER) |
|      | 1 = 'COM1'              |                 |
|      | 2 = 'COM2'              |                 |
|      | 3 = 'COM3'              |                 |
|      | 4 = 'COM4'              |                 |
|      | 5 = 'COM5'              |                 |
|      | 6 = 'COM6'              |                 |
|      | 7 = 'COM7'              |                 |
|      | 8 = 'COM8'              |                 |

9 = 'COM9'  
10 = 'COM10'  
11 = 'COM11'  
12 = 'COM12'

**BUFFER** - Send data buffer (INPUT CHARACTER)

**NBYTES** - Number bytes to send (INPUT INTEGER)

**STATUS** - Returned status (OUTPUT INTEGER)

0 = Normal return  
1 = Port not open  
2 = Port not supported  
3 = Invalid arguments  
4 = Queued memory exhausted  
5 = Transmission errors detected

**Notes:**

1. The user's data will be transferred immediately to serial output buffer by COMSND and return will be made to the user's program. Serial data transmission will proceed at driver interrupt level. Two way communication on a single serial port is possible. Caution should be exercised when using very high baud rates. Some processors are not able to keep up with a double load. The flow control available in COMOPN should be considered when doing two way communication. Data overruns may occur when using high baud rates in this interrupt intensive mode of operation.

2. The user may interrogate the status of the output buffer before calling COMSND by calling COMSTS.

3. The COMCTL routine provides the capability to have a timestamp put at the beginning of all consnd output streams. COMCTL also has an option to do all the COMSND outputs without waiting. The output buffers will be queued into system global storage. A separate processing thread will be created to output the buffers asynchronously when the primary thread is not active. See COMCTL for details.

**Error Conditions:**

1. See status parameter definition for all error codes.

**Program Example:**

```
PROGRAM TCOMSND
!
!   SciComm interface definition
!
USE SCCINF
IMPLICIT NONE
!
!   Variables definitions
!
```

```

INTEGER(4) :: PORT, BAUD, PARITY, STPBTS, CHRLEN, FLOW, STATUS,
& CODE, VALUE, NBYTES, RLEN, SLEN
CHARACTER(1),DIMENSION(80) :: BUFFER = (/80*'X'/)
!
! Initialize COM2 port: 9600 baud, no parity, 1 stop bit,
! 8 bits, XON/XOFF disabled
!
PORT = 2
BAUD = 7
PARITY = 1
STPBTS = 1
CHRLEN = 5
FLOW = 1
RLEN = 8192
SLEN = 8192
!
! Open COM2 port
!
CALL COMOPN(PORT,BAUD,PARITY,STPBTS,CHRLEN,FLOW,RLEN,SLEN,STATUS)
IF (STATUS.NE.0) THEN
WRITE(6,100)STATUS
END IF
!
! Write a string of characters
!
NBYTES = 80
CALL COMSND(PORT,BUFFER,NBYTES,STATUS)
IF (STATUS.NE.0) THEN
WRITE(6,101)STATUS
END IF
!
! Close COM2 port
!
CODE = 1
CALL COMCLS(PORT,CODE,STATUS)
IF (STATUS.NE.0) THEN
WRITE(6,102)STATUS
END IF
!
! Exit processing
!
STOP
!
! FORMAT Statements
!
100 FORMAT(' *** Errors (COMOPN):',I4,' ***')
101 FORMAT(' *** Errors (COMSND):',I4,' ***')
102 FORMAT(' *** Errors (COMCLS):',I4,' ***')
!
END PROGRAM TCOMSND

```

### Example 3.7 – COMSND Program Example

#### 3.2.8 COMDTM

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

Fortran Calling Sequence:

```
CALL COMDTM(MONTH,DAY,YEAR,HOUR,MINS,SECS,MSECS)
```

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 COMDTM to measure the length of time between two events in the user application.

Error Conditions:

None.

Program Example:

```
PROGRAM TCOMDTM
!
! SciComm interface definition
!
USE SCCINF
IMPLICIT NONE
!
! Variables definitions
!
INTEGER(4) :: MONTH, DAY, YEAR, HOUR, MINS, SECS, MSECS
!
! Get date and time
!
CALL COMDTM(MONTH, DAY, YEAR, HOUR, MINS, SECS, MSECS)
WRITE(6,100)MONTH, DAY, YEAR, HOUR, MINS, SECS, MSECS
!
! Exit processing
!
STOP
!
! FORMAT Statements
!
100 FORMAT(' MONTH = ',I4,',', ' DAY   = ',I4,',', ' YEAR = ',I4,',',
&        ' HOUR  = ',I4,',', ' MINS  = ',I4,',', ' SECS = ',I4,',',
&        ' MSECS = ',I4)
!
END PROGRAM TCOMDTM
```

## Example 3.8 – COMDTM Program Example

### 3.2.9 COMSUS

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

Fortran Calling Sequence:

```
CALL COMSUS(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 COMSUS to delay execution, waiting for the arrival of enough data without using any cpu cycles.

Error Conditions:

1. See status parameter definition for all error codes.

Program Example:

```
PROGRAM TCOMSUS
!
!   SciComm interface definition
!
!   USE SCCINF
!   IMPLICIT NONE
!
!   Variables definitions
!
!   INTEGER(4) :: MSECS,STATUS
!
!   Suspend execution for 5 seconds
!
!   MSECS = 5000
!   CALL COMSUS(MSECS,STATUS)
!   IF (STATUS.NE.0) THEN
!       WRITE(6,100)STATUS
!   END IF
!
!   Exit processing
!
!   STOP
!
!   FORMAT Statements
!
```

```
100  FORMAT(' *** Errors (COMSUS):',I4,' ***')  
!  
      END PROGRAM TCOMSUS
```

### **Example 3.9 – COMSUS Program Example**

### **3.3 Communication Test Program**

The communication test program (**TCOM**) is released as Fortran source code with SciComm. This program can be linked with the SciComm library to produce an executable program. The program allows a user to test out the serial communication port through the use of an interactive menu driven program. A loop back test may be performed by electrically connecting the send pin to the receive pin on the serial communications port connector. TCOM allows the user to test out and examine the transfer of known, user generated, data streams.

All of the functions provided in the SciComm communication library can be tested through TCOM. A port may be opened using software, hardware, on no flow control. The hardware status registers may be interrogated. The number of characters received or left to send can be viewed by using the status function. Finally, the flush function will delete all characters in the input or output buffers.

### **3.4 Communication Library**

SciComm is released with two Fortran linkable libraries. There is the SCCLIB.LIB static library that can be linked with the application to produce an executable file. There is also the SCCDLL.DLL dynamic link library that can be dynamically linked during execution by including the SCCIMP.LIB during the LINK process. These SciComm libraries are used in conjunction with the standard runtime Fortran library. The user writes application software with calls to SciComm routines described in this manual. At LINK time, the user includes the SciComm library of his choice with the Fortran library.

## Chapter 4

### Performance Considerations

The Windows operating system performs certain operations in background at a interrupt level. When the timer interrupt occurs, the processor is used to perform a whole series of update operations. Changes in data being displayed on client windows cause display adapter updates. Such operations can inhibit processing input serial data. This can lead to data overruns on the input serial port. This is not normally a problem. At high baud rates this can become a problem. At 56,000 baud, a byte of data will come in every 179 microseconds. The serial controller can usually keep up using hardware fifo's, but the processor does have to be available to read the data from the serial controller when it arrives and to store the byte in the driver buffer. The user must insure correct operation of the serial i/o application by a judicious use of check summing, parity, and attention to the error status variables.

#### 4.1 RS232C Controller Measured Data

The SciComm product has been tested by connecting two processors via a serial port using a special cable. An application on the first processor was executed that sent a buffer of 2000 bytes to the second processor. The second processor read in all 2000 bytes and then sent the same 2000 bytes back to the first processor. The first processor verified the integrity of every byte and recorded the round trip time. This sequence was repeated 500 times to obtain good statistics. The mode used was no parity, 8 data bits, and 1 stop bit. The total number of bits for each byte transmitted was 10. This included 1 start bit, 8 data bits, and 1 stop bit. The results of this test were:

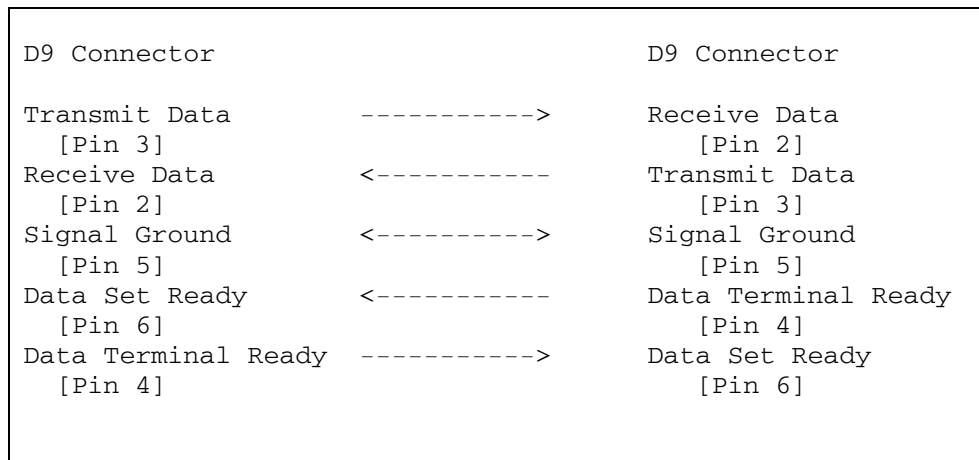
Baud Rate (Bits/second)	Measured Rate (Bytes/second)	Measured Rate (Bits/second)	Bandwidth Used (%)
300	29	299	99
1200	119	1196	99
9600	956	9569	99
14400	1440	14400	100
19200	1913	19138	99
38400	3827	38277	99
56000	5594	55944	99
115200	11126	111265	96

**Table 4.1 – Serial Controller Measurements**

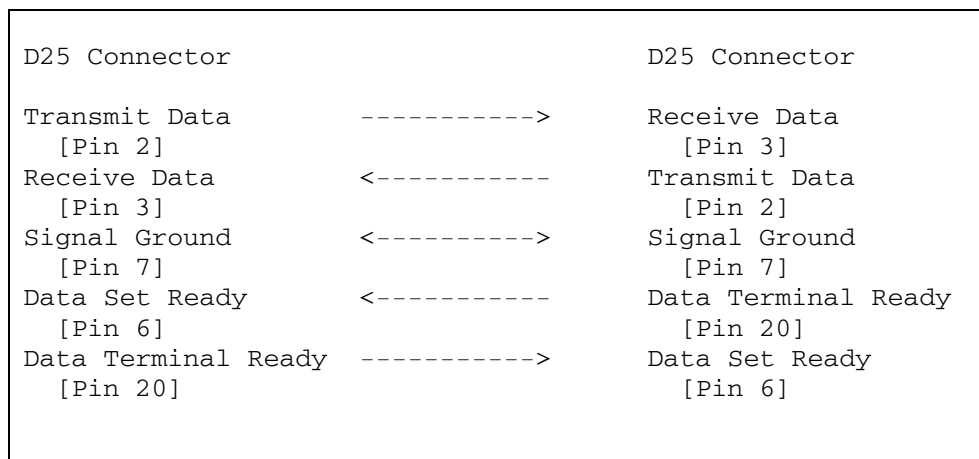
## Chapter 5

### Hardware Interfaces

Cable wiring between two processors is always a difficult proposition. Because there are so many choices in connecting cable wires to connectors, this task appears difficult to do correctly. Several wiring diagrams are provided which allow XON/XOFF or hardware flow control. The Windows drivers have to function with the cable attached between two computers. The following are three possible cabling wirings:



**Table 5.1 – D9/D9 Connector Cable**



**Table 5.2 – D25/D25 Connector Cable**

D9 Connector		D25 Connector	
Transmit Data [Pin 3]	----->	Receive Data [Pin 3]	
Receive Data [Pin 2]	<-----	Transmit Data [Pin 2]	
Signal Ground [Pin 5]	<----->	Signal Ground [Pin 7]	
Data Set Ready [Pin 6]	<-----	Data Terminal Ready [Pin 20]	
Data Terminal Ready [Pin 4]	----->	Data Set Ready [Pin 6]	

**Table 5.3 – D9/D25 Connector Cable**

# INDEX

## A

asynchronous, 18

## B

baud rate, 2  
baud rates, 19  
bits per character, 4  
Buffer flush, 14  
bytes in receive buffer, 10  
bytes in send buffer, 10

## C

C, 24  
Cable wiring, 26  
com1 through com12, 1  
COMCLS, 6  
COMCTL, 11  
COMDTM, 20  
COMFLS, 14  
COMOPN, 3  
COMREC, 15  
COMSND, 18  
COMSTS, 8  
COMSUS, 22

## D

**D25 Connector**, 26  
**D9 Connector**, 26

data overruns, 25

Data overruns, 19  
data-terminal-ready, 12  
disabled, 7  
Driver interrupt level, 19  
DTR/DSR, 1  
dynamic link library, 3, 24

## E

empty, 5  
Error status, 9

## F

flow control, 3, 19  
Flow control, 5  
flush, 14  
Fortran, 1, 3

## I

IDE, 3  
Interrupt level, 25

## L

Length of receive buffer, 10  
Length of send buffer, 10  
line break, 12

Line Status register, 9, 16

LINK, 24

local time and date, 20

## M

Measured Data, 25  
Modem Status register, 9, 16

## O

out-of-memory, 12  
outputs without waiting, 19  
overflowing, 5

## P

Parity, 4

## R

request-to-send, 12  
RS-232C, 1  
RTS/CTS, 1  
runtime, 24

## S

SCCIMP.LIB, 3, 24  
SciComm, 3  
SCPDLL.DLL, 3, 24

SCPLIB.LIB, 3, 24  
separate processing thread, 12  
static library, 3, 24  
status registers, 8  
Stop bits, 4  
suspend execution, 22  
system global storage, 12, 19

## T

**TCOM**, 24  
timestamp, 19  
timestamp option, 12  
two way communication, 19

## W

WinAPI interface error, 5  
Windows API, 1  
write no-wait queued option, 12

## X

XOFF, 5  
XON, 5  
XON/XOFF, 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

## SciComm 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

# Software Report

## SciComm Version 7.0

Name : \_\_\_\_\_

Company : \_\_\_\_\_

Address : \_\_\_\_\_

Address : \_\_\_\_\_

City : \_\_\_\_\_ State/Province : \_\_\_\_\_

Postal Code : \_\_\_\_\_ Country : \_\_\_\_\_

Voice : \_\_\_\_\_ Fax : \_\_\_\_\_

Email : \_\_\_\_\_

Purchase Date : \_\_\_\_\_

Problem or suggestion (include examples if possible):

---

---

---

---

---

---

---

---

---

---

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

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