BSCLIB

Multi-Platform Bisync Developer’s Tool Kit

Programmer's Guide

Serengeti Systems Incorporated

Printed in the USA

August 19, 2002

DISCLAIMER

BSCLIB is sold as is. Serengeti Systems Incorporated (Serengeti) makes no representations or warranties with respect to the contents hereof and specifically disclaims any implied warranties for a particular purpose.

Serengeti shall have no liability for loss or damage caused or alleged to be caused directly or indirectly by this computer program, including but not limited to interruption of service, loss of business or anticipatory profits or consequential damages resulting from the use of this program.

Further, Serengeti reserves the right to revise this publication and program from time to time without notice.

ATTRIBUTION CLAUSES

Table of Contents

Table of Figures

SECTION **PAGE**

1 INTRODUCTION

BSCLIB is a software tool kit for emulating the point-to-point and multi-point Binary Synchronous Communications (BSC) protocol. BSCLIB provides an Applications Program Interface (API) that supports a variety of programming languages, operating systems, and communications hardware.

The API interface provides a straightforward means of controlling information flow between your program and a BSC communications link. It allows a user to create a custom application that seamlessly integrates BSC communications.

Typical BSCLIB applications include Electronic Data Interchange (EDI), host-to-terminal links, point-of-sale (POS) systems, automatic funds transfer, insurance claims processing, remote data collection, and many other batch file-transfer applications. Any application that interfaces with a BSC network and requires tight integration of local and remote systems is a candidate for BSCLIB.

The BSCLIB API allows an application program to transfer information as it is needed or in batches. A properly designed BSCLIB application seamlessly integrates data communications with its other functions. The entire communications session can occur transparently to the application's end-user.

BSCLIB gives the developer a consistent BSC interface for a variety of operating systems and communications adapters. BSCLIB applications can connect to remote systems that support Binary Synchronous Communications as defined in the IBM publication, General Information – Binary Synchronous Communications (IBM Order No. GA27-3004).

1.1 Background

The Binary Synchronous Communications protocol has been in use since the 1960s. Although the IBM publication noted in the preceding section is the definitive authority on BSC, even it leaves some issues unaddressed. Over the years, many developers have used BSC in varying ways to achieve a wide variety of goals.

For many developers, BSCLIB can be used with less than a full understanding of the binary synchronous protocol. However, the BSC protocol does involve many concepts, and it would be difficult to say that anybody can develop a BSC application with absolutely NO protocol

knowledge. How much any developer may need to learn will be a function of complexity of their application.

This manual may refer to unfamiliar BSC concepts. Do not think you need to understand all of them. BSCLIB provides far more options and far more capability than the average application will ever require. For most developers, the best way to use this manual will be as an adjunct to the code samples provided with BSCLIB, rather than as a tutorial for cover-to-cover reading. This guide attempts to explain a few concepts, but is more focused on the development of BSCLIB applications than the in-depth details of the BSC protocol.

From our experience, we suggest you read the first two sections carefully, to gain a good overview of the product. Next, skim Sections 3, 4 and especially 5, configuring BSCLIB. Armed with that understanding, you can return to Sections 3 and 4 for a more in-depth examination to determine whether you can make use of the high-level BSCLIB Wrapper Library (BSCAWL) or whether you need to get "down and dirty" with BSCLIB's Low Level Interface Functions.

Serengeti's technical support is available to assist you with detailed questions on how to make the best use of BSCLIB. They are also available to assist in the actual design and development of your application for a fee. They are NOT available to teach the bisync protocol. If you do require more understanding of BSC than you receive from this manual and this product, you should consult the IBM publication and other resources.

1.2 BSCLIB Features

- **Applications Interface Features**
  - support for C, C++, Visual Basic, and Java applications
  - auto-dial and auto-answer support for a variety of modems
  - integrated line trace for line monitoring
  - configurable ASCII↔EBCDIC translation tables
  - blocking and non-blocking I/O
  - real-time session statistics
  - support for receiving "uni-directional" or "simplex" data feeds

- Standard BSC Protocol Features
  - 2780/3780 point-to-point emulation
  - multi-point link emulation (see below)
  - Vertical Forms Control (VFC) recognition

- device select recognition
- space compression/expansion (3780 only)
- space truncation (2780 only)
- WACK, RVI and TTD support
- EBCDIC New Line (NL) character recognition
- transparent text mode
- terminal identification
- transmit and receive double buffering
- CRC-16 block checking
- ASCII data link control characters
- LRC block checking for ASCII data link control
- SOH headed blocks (inbound and outbound)
- sending and receiving limited conversational replies
- auto-detection of incoming 2780 or 3780 data streams

· **NON-Standard BSC Protocol Features**
- binary file mode
- optional stripping of VFC and device select sequences
- variable communications buffer sizes up to 4192 bytes
- optional suppression of inbound and outbound inter-record separators

· **Multi-Point BSC Protocol Features**
- support for both control and tributary stations
- recognition of multiple select addresses
- automatic poll or select recognition

1.3 BSCLIB Hardware Options

BSCLIB supports connections through a single-port synchronous adapter, an external AutoSync modem or an intelligent multi-port synchronous adapter. Only one type of device can be used on a system at a time, however, up to four multiple single-port or six multi-port adapters can be installed on a single system.

1.3.1 Single-Port SyncPCI

The single-port SyncPCI version of BSCLIB supports one BSC connection. The connection is achieved with the installation of a Serengeti Systems

SyncPCI synchronous communications adapter. The SyncPCI adapter supports a wide variety of dial-up and leased-line telephone connections, as well as connections to FRADs, CSU/DSUs, and modem eliminators that use and RS-232 interface.

1.3.2 Multi-Port SyncPCI

Multi-port capability may be achieved with the installation of up to four SyncPCI single-port adapters. Using multiple SyncPCI adapters is a low cost way to provide up to four connections on a single system.

1.3.3 Single-Port AutoSync

The single-port AutoSync version of BSCLIB supports one BSC connection. The connection is achieved via a standard asynchronous serial port and an AutoSync capable modem. The AutoSync functionality is supported by the Hayes Optima Business Modems manufactured by Zoom Telephonics. An AutoSync connection is limited to dial-up telephone lines only.

AutoSync is a feature of the Optima modems that permits synchronous communications to occur between the modem and the remote system while the local connection between the PC and modem is asynchronous. As such, the AutoSync version of BSCLIB uses a COM port on Windows systems or a TTY port on Unix systems.

1.3.4 Multi-Port SmartSync/DCP

Multi-port capability may also be achieved with the 8-port SmartSync/DCP communications co-processor adapter. The SmartSync/DCP version of BSCLIB supports up six adapters, providing up to 48 simultaneous BSC connections from a single system. The SmartSync/DCP adapter support a wide variety of dial-up and leased-line telephone connections, as well as connections to FRADs, CSU/DSUs, and modem eliminators.

The SmartSync/DCP adapter offers the following advantages over multiple SyncPCI adapters:

- Only requires a single PCI slot to provide eight BSC connections
- Supports up to six adapters totaling 48 simultaneous BSC links
- Offloads BSC emulation processing from main CPU

To support multiple ports, you write a BSCLIB application that controls a single-port and execute a separate instance of the application for each port

required (the AWLTEST and CTEST example programs work this way.) Another method for supporting multiple ports is to write a single program that services each port in sequence or by a priority scheme appropriate for your application.

1.3.5 3780 vs. 2780 Point-to-Point Emulation

BSCLIB supports the 3780 and 2780 variants of the BSC protocol. BSCLIB can auto-detect if incoming data is in 3780 or 2780 format during the first re receive operation. It can than configure itself to operate in that mode for the duration of the session. BSCLIB can also be to only operate in 3780 or 2780 mode.

1.3.6 Multi-Point Emulation

BSCLIB supports both point-to-point BSC emulation, typically used by 3780 and 2780 RJE data terminals, as well the multi-point (not to be confused with BSCLIB multi-port), also known as multi-drop, variant of the BSC protocol. BSC 3270 terminals are connected in a multi-point environment.

Typically, point-to-point and multi-point BSC environments are mutually exclusive and you will not create an application that supports both. In fact, multi-point BSC environments are extremely rare.

2 APPLICATION STRUCTURE

BSCLIB is available for Microsoft Windows (98 / ME / NT / 2000 / XP) and several Unix operating systems including x86 Linux, Sparc Solaris, and AIX.

Not all versions of BSCLIB are supported on all operating systems. For an up-to-date list of supported platforms, contact Serengeti Systems.

The BSCLIB API is nearly the same across all supported operating systems. Applications can be ported from one supported system to another with relative ease. The following sections provide an overview of how BSCLIB operates in different environments.

2.1 Single-Port SyncPCI BSCLIB Applications

Single-port BSCLIB applications that use the SyncPCI synchronous adapter consist of two separate programs. The application program, which you will write, links with the library portion of BSCLIB, presents the user interface, reads and writes files, creates log files, etc. It communicates, via a shared memory segment, with the second program, a background process (often referred to as a *daemon process* in Unix) that implements the BSC protocol.

The background process, **emubsc**, is referred to as the **BSC Protocol Handler**. It is loaded automatically by BSCLIB and automatically terminated when the communication session is complete.

The SyncPCI device driver is also required by BSCLIB, but your program will not interface directly with it. The structure of a Windows and Unix single-port BSCLIB applications are shown in the following diagrams.

2.2 Multi-Port SyncPCI BSCLIB Applications

Multi-port BSCLIB applications that use two, three, or four SyncPCI adapters are simply replications of the single-port application described previously. For each SyncPCI adapter installed, the complete set of processes described for a single-port SyncPCI application, are replicated.

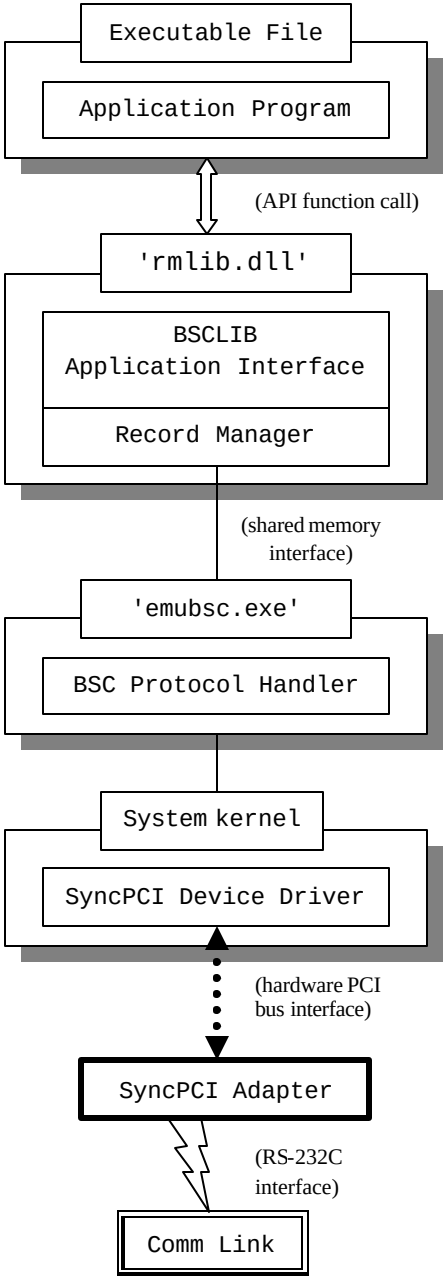

Figure 1. Windows SyncPCI BSCLIB Application

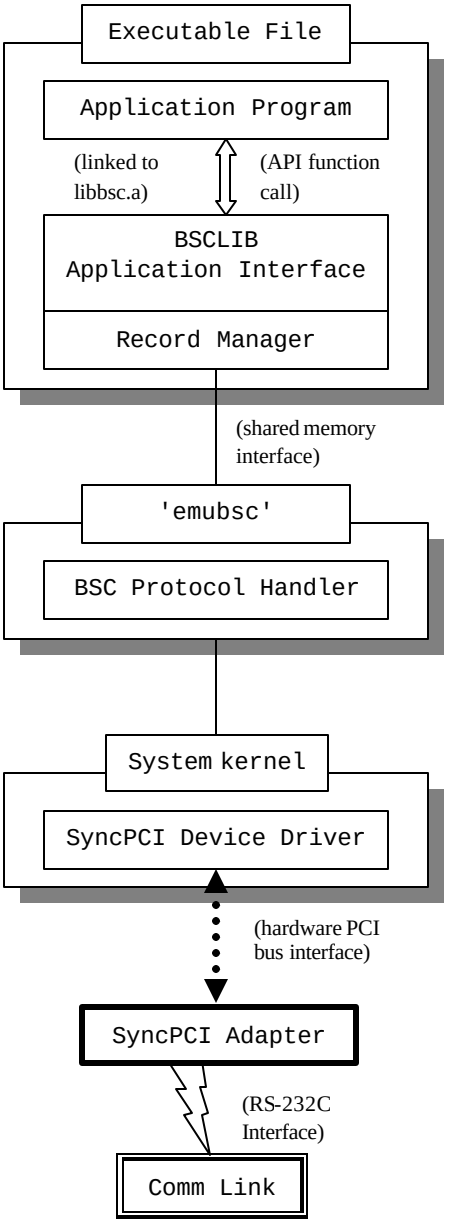

Figure 2. Unix SyncPCI BSCLIB Application

2.3 Single-Port AutoSync BSCLIB Applications

AutoSync BSCLIB applications consist of three separate programs. The first program is your application. This program, which is linked with the library portion of BSCLIB, presents the user interface, reads and writes files, creates log files, etc. The application program communicates, via a shared memory segment, with two background processes (often referred to as a *daemon process* in Unix) that implement the BSC protocol.

The first background process, **emubsc**, is referred to as the **BSC Protocol Handler**. It is started automatically by BSCLIB and automatically terminated when the communication session is complete.

The second background process, **abscdrvr**, handles AutoSync functions and is also automatically started by BSCLIB and then terminated when the communication session is complete.

No additional device drivers are required with the AutoSync version of BSCLIB.

The structure of a Windows and Unix AutoSync BSCLIB application are shown in the following diagrams.

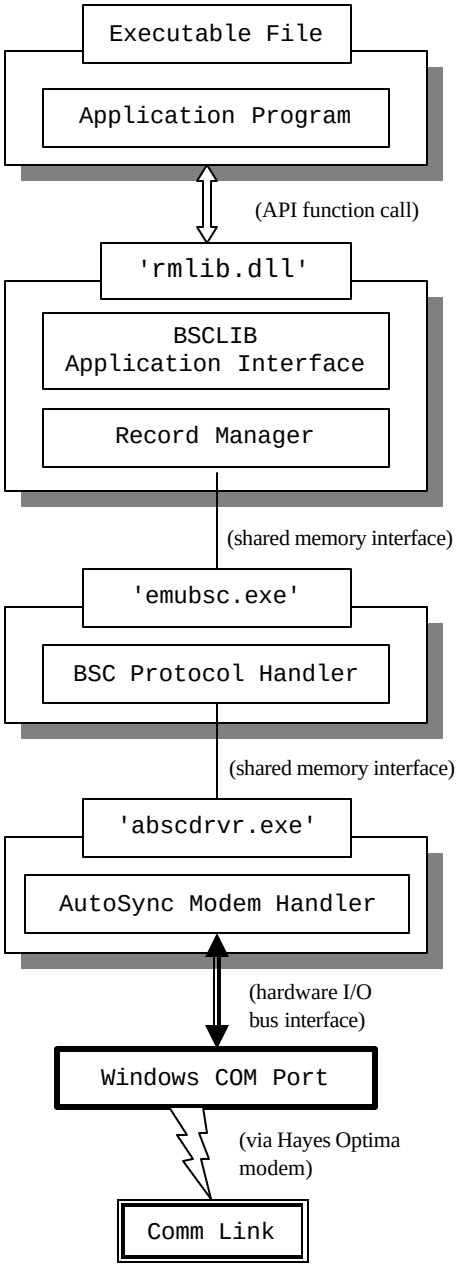

Figure 3. Windows AutoSync BSCLIB Application

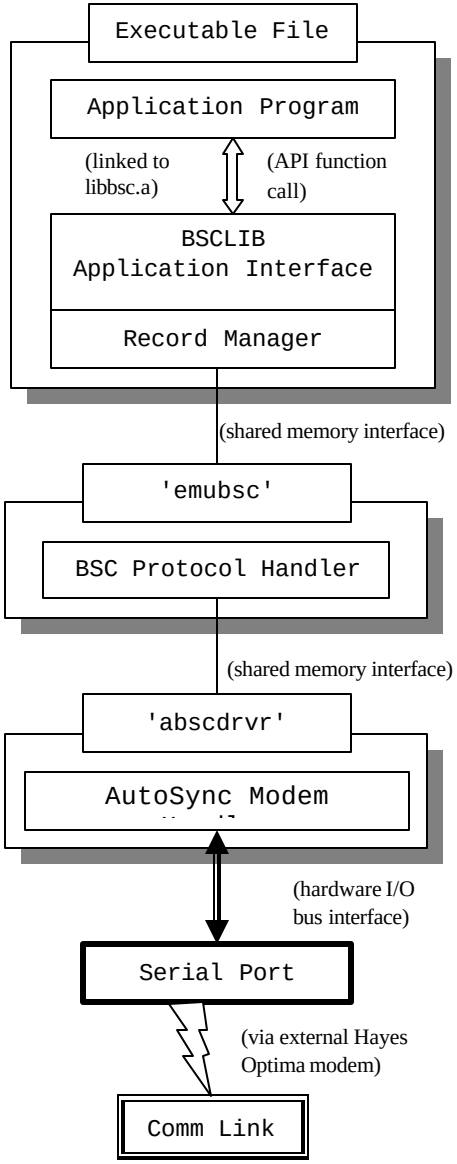

Figure 4. Unix AutoSync BSCLIB Application

2.4 Multi-Port SmartSync/DCP BSCLIB Applications

Multi-port applications that use the SmartSync/DCP adapter can have up to 48 ports. All BSC protocol tasks are downloaded to the adapter and run using the board's co-processor. The program that is downloaded to control all eight ports on the adapter is named **amxbsc.bin**. The **dcpload** program is used to initialize the board and must be run prior to starting your BSCLIB application. See Appendix for more information on **dcpload**.

The XDCP device driver must be installed in your system to enable access to the SmartSync/DCP adapter.

The structure of a Windows and Unix SmartSync/DCP BSCLIB applications are show in the following diagrams.

BSCLIB supports up to six SmartSync/DCP adapters in a single system. Your application can seamlessly address a given port on any board.

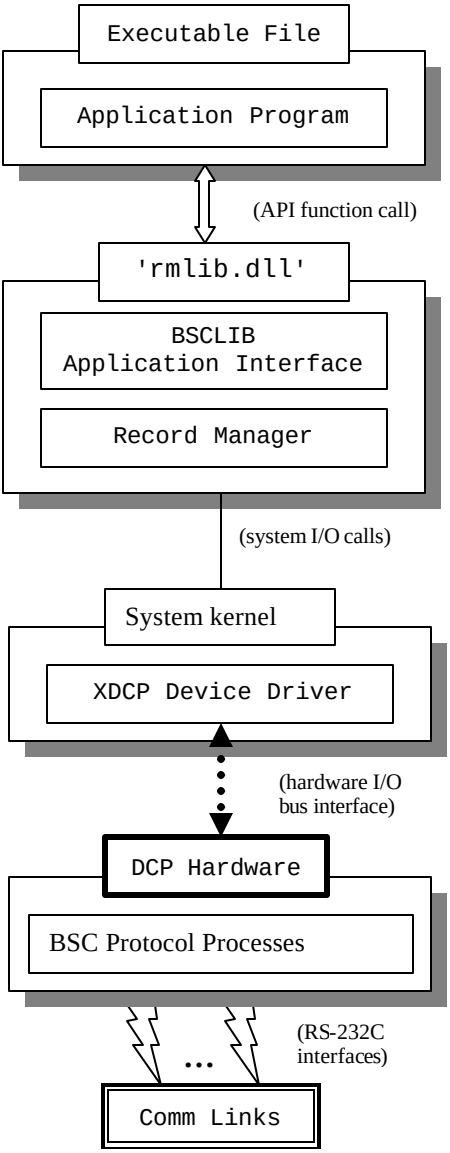

Figure 5. Windows DCP BSCLIB Application

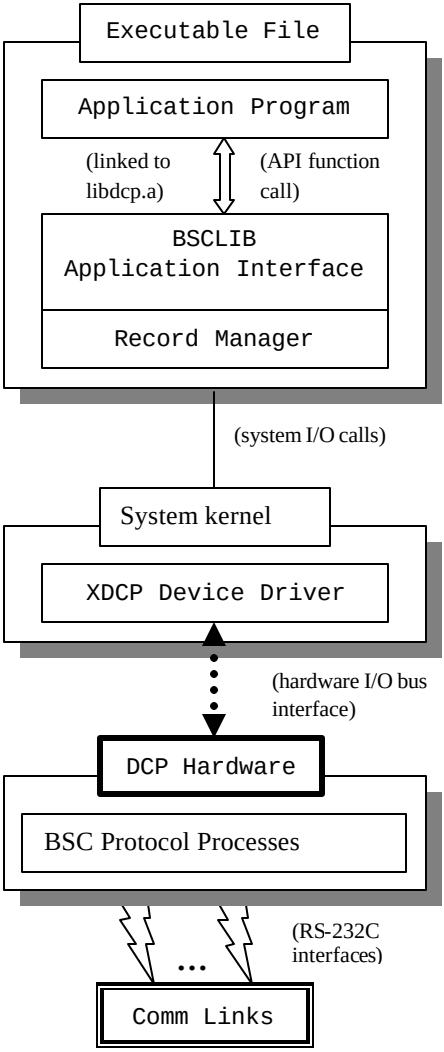

Figure 6. Unix DCP BSCLIB Application

2.5 The "Layers" of BSCLIB

We have just addressed the variations among BSCLIB linked modules across different operating systems and communications interfaces. From a conceptual perspective, however, every BSCLIB application can be viewed "top-down" as a number of layers:

1. Application program
2. BSCLIB API Wrapper Library (BSCAWL)
3. Low-Level BSCLIB API (BAPI)
4. Record Manager (RM) Layer
5. BSC Protocol Manager (BPM) Layer
6. BSC Device Drivers

2.5.1 Application Program and API

A BSCLIB application program may be written in any programming language capable of issuing an external call to a Windows DLL or linking to Unix library. Languages most commonly used are C, C++, Visual Basic, and Java. The application program is linked to one or both of BSCLIB's two APIs.

The BSCLIB API Wrapper Library (BSCAWL) is a high-level API that provides a variety of functions necessary to implement the BSC protocol. Chapter 3 provides a detailed description of BSCAWL. BSCAWL is built on top of the BSCLIB API (BAPI), which is a low-level API. Use of BAPI requires the application to do bit-level manipulation of data structures, however, BAPI does provides certain low-level functions that are not available in BSCAWL. Chapter 4 provides a detailed description of BAPI.

2.5.2 Record Manager (RM) Layer

The Record Manager (RM) layer oversees the transfer of logical records between your program's buffers and the buffers used to exchange data with the communications link. For BAPI applications the program's buffer is specified in the BSCLIB Control Block (BCB). When overseeing this process, the RM manages the various options requested by your program.

When receiving, the RM waits for a communications buffer to be received. When a block is available and a request from your program is received, logical records are transferred one at a time until the block is empty. Before a logical record is handed to the application interface, compressed spaces (if any) are expanded and an EBCDIC to ASCII translation is performed (if these options are enabled.)

Inbound logical records are either delimited by inter-record separators and end-of-block characters, as defined by the BSC protocol, or fixed length. Your program may disable BSCLIB's recognition of one or more of these delimiters resulting in *physical record I/O.*

Also, printer and punch device selection and Vertical Forms Control (VFC) sequences may be passed through to your program, translated to CR/LF or LF equivalents, or stripped from the data stream based on your program's requests. Your program is responsible for recognizing and acting upon these sequences if they are passed through.

When transmitting, the RM accepts logical records from the application interface, performs space compression and an ASCII to EBCDIC translation (if enabled.) The logical records are placed in a comm buffer for transmission. Your program has record-by-record control over outbound record blocking using the following options:

- Fill communications buffer and continue
- End communications buffer with ETB and continue
- End communications buffer with ETX and continue
- End communications buffer with ETX, then transmit an EOT, which terminates the transmission

On both transmit and receive, EBCDIC transparency enables BSC line control characters, such as ETX and EOT, to be transferred as data. Your program may enable and disable outbound transparency, but inbound transparency is detected and handled automatically.

The Binary I/O option is a special case of EBCDIC transparency that disables ASCII↔EBCDIC character translation. This mode is useful for transferring files containing characters that do not have EBCDIC equivalents, such as ".EXE" and ".COM" files (executable files.) This mode is not always supported by mainframe computers, but is a feature of most 3780/2780 terminal emulation products.

If your program needs to manipulate *non-transparent* inbound and outbound data streams directly using the EBCDIC character set, BSCLIB supports an option that disables ASCII↔EBCDIC translation entirely. BSCLIB assumes the data stream to be in EBCDIC.

BSCLIB may also be configured for ASCII data link control, in which all data link control characters and data are using the ASCII character set. BSCLIB supports odd parity or no parity in ASCII data link control mode.

2.5.3 BSC Protocol Manager (BPM) Layer

The BSC Protocol Manager (BPM) Layer controls the establishment and termination of a communications session, and the sending and receiving of communications buffers during the session.

In single-port and AutoSync versions of BSCLIB, the BPM layer is a separate process that is loaded by RM layer and communicates via shared memory. In multi-port SmartSync/DCP environments, the BPM layer is downloaded and run on the board. The different implementations of the BPM layer have no affect on your application program.

A BSCLIB application programmer is generally not concerned with the operations of the BPM layer. But, it may be helpful to understand that the BPM layer is driven by events on the communications link and by requests from the RM layer that are in turn generated by I/O requests from your application program.

The BPM layer handles BSC protocol functions such as bidding for the line to establish a session, Cyclical Redundancy Check (CRC-16) verification of comm buffers, Temporary Text Delay (TTD) and Wait Acknowledgment (WACK) flow control, even/odd comm buffer sequencing, etc.

2.5.4 BSCLIB Device Drivers

BSCLIB is shipped with a device driver that corresponds to the communications adapter you selected when ordering. Both the SyncPCI and SmartSync/DCP adapters are plug-and-play adapters that do not require any hardware configuration prior to installation. When BSCLIB is used with an AutoSync modem, BSCLIB does not install a separate driver, but it does utilize the default serial device driver provided with the operating system.

3 BSCLIB PROGRAMMING

The BSCLIB API Wrapper Library (BSCAWL) is a high-level interface built on top of the low-level BSCLIB API (BAPI). Under Windows, BSCAWL is a DLL with both a C language interface and C++ Class Library interface. Any language that can access standard Windows DLLs, such as Visual Basic and Java, can utilize BSCAWL. Under Unix, BSCAWL is static library file.

Not all of the capabilities of BAPI are available through BSCAWL functions, most notably some of the multi-point capabilities and unidirectional receive functionality. A special BSCAWL function, **BSCIssueAPICall**, allows the developer to access all BAPI capabilities. To learn more about BAPI, refer to Chapter 4.

BSCAWL is multi-threaded and is written in C. The Pthreads library is utilized in the Unix environment. BSCAWL provides callback functionality for languages that support callbacks. A callback function is used to notify an application program that a requested BSCLIB operation has completed. For languages that do not support callbacks, BSCAWL calls are blocking so a multi-threaded application design is recommended. All BSCAWL calls are thread-safe.

BSCAWL is provided "as is". Complete source code, including API header files, is provided in the **bscawl** subdirectory. Users are authorized to modify the source code for their needs; however, Serengeti Systems will not be responsible for BSCAWL functionality once any modifications have been made.

Unix BSCAWL applications need to link the **libbscawl.a** static library and either **libbsc.a** for AutoSync and SyncPCI applications or **libdcp.a** for SmartSync/DCP applications. To link these libraries to your application, add the **-lbscawl** option and either **-lbsc** or **-ldcp** to your link command. BSCLIB libraries are installed in the system library directory, usually **/usr/lib**.

Windows BSCAWL applications need to link to **bscawl.dll**. In addition, **bscawl.dll** will load **rmlib.dll** and either **xbsc.dll** or **xdcpdrvr.dll**, depending on the communication adapter being used.

3.1 AWLTEST -- BSCAWL Sample Application

An example application, **awltest**, and its complete C source code are provided with the BSCLIB Software Development Kit. The source code is located in the **samples/awltest** subdirectory under the installation directory. This program exercises most of the capabilities of BSCAWL.

For Unix development, a make file, **Makefile**, is provided to compile and link the program. This make file demonstrates the correct compiler and link options to use when developing a program that uses BSCLIB. To build the sample program, simply run **make** from the **samples/awltest** subdirectory.

For Windows development, project files compatible with Microsoft Visual Studio 6 are provided to compile and build various versions of the program. The project file is located in the **samples/awltest** subdirectory.

The resulting program is useful as both a demonstration and test program for BSCLIB in a point-to-point (contention mode) BSC environment. (Compiler switches allow you to build a version that is for multi-point BSC environments, as well.) You can exercise virtually all of BSCLIB's functionality from menus and prompts within the program. The BSCLIB result codes are displayed immediately after a function returns. This program is a valuable learning aid for new BSCLIB programmers.

3.2 BSCAWL Functions (in Alphabetical Order)

BSCAbort	Abort pending OPEN, READ, WRITE operation.
BSCAnswer	Put modem into auto-answer mode, and answer incoming call.
BSCClearReceivedTerminalID	Reset any saved terminal ID received.
BSCClearStatistics	Zero out contents of current SPB.
BSCClose	Close communications session. (i.e. drop the connection)
BSCConnect	Raise DTR in anticipation of establishing a connection.
BSCCreateHandle	Get BSCAWL handle to a new BSCAWL context (must be first call made to BSCAWL).
BSCDial	Dial modem to establish a connection.
BSCGetATAnswerString	Get the currently configured AT modem auto-answer initialization string.

BSCGetATInitString	Get current AT modem initialization command string.
BSCGetHardwareType	Get type of Serengeti hardware installed
BSCGetNumDCPBoard	In SmartSync/DCP environments, Get number of boards installed.
BSCGetParms	Get pointer to current PIB.
BSCGetStatistics	Get contents of current SPB.
BSCGetTranslationTable	Get the current ASCII – EBCDIC translation tables.
BSCHardwareCommand	Issue BSCLIB hardware command call.
BSCInitialize	Initialize BSCLIB configuration parameters.
BSCInstall	Install and start BSCLIB components.
BSCIsInstalled	Get BSCLIB installation status.
BSCIsOpen	Get BSCLIB line connection status.
BSCIssueAPICall	Issue a BSCLIB API (BAPI) function call.
BSCLoadSettings	Read configuration parameters from specified .ini file and save in PIB.
BSCRead	Issue BSCLIB READ call.
BSCReadAddSelectAddressToList	In multi-point mode, add an address to select address list.
BSCReadGetFlags	Get BCB flags from most recent READ operation.
BSCReadGetTimeout	Get READ BCB time-out set in the current BSCAWL context.
BSCReadGetReceivedTerminalID	Get a received terminal ID.
BSCReadGetSelectAddressList	In multi-point mode, get multiple select addresses currently defined.
BSCReadGetSelectedAddress	In multi-point mode, get selected station address.
BSCReadInitiateRVI	Issue BSCLIB READ, and then issue RVI, requesting a line turnaround.
BSCReadRemoveSelectAddress-FromList	In multi-point mode, remove an address from select address list.
BSCReadSetPollInteval	In multi-point mode, set poll interval when retry count is greater than one.
BSCReadStoreMultiPointAddress	In multi-point mode, store single address to be used with a READ command.
BSCReadStoreSelectAddressList	In multi-point mode, store multiple addresses to recognize when selected.

BSCReadToETX	Issue BSCLIB READs until ETX terminated block is received.
BSCReadUnidirectional	Issue BSCLIB READ in "simplex" or uni-direction mode.
BSCReceiveFile	Receive a file.
BSCCloseHandle	Stop/remove BSCLIB components.
BSCSaveSettings	Save current configuration parameters to specified .ini file.
BSCSendFile	Send a file.
BSCSetBlockRespTimeout	Set block response time-out.
BSCSetDTR	Set DTR modem signal on or off.
BSCSetMaxBaudRate	Set maximum DTE baud rate for asynchronous modem dialing.
BSCStatus	Issue BSCLIB STATUS call.
BSCStoreATAnswerString	Set new AT modem command string to enable auto answer.
BSCStoreATInitString	Set new AT modem initialization command string.
BSCStoreStatistics	Store new values in current SPB.
BSCSetTerminalID	Store optional terminal ID.
BSCTrace	Turn BSCLIB tracing on and off.
BSCStoreTranslationTable	Store new set of ASCII – EBCDIC translation tables.
BSCWrite	Issue BSCLIB WRITE call.
BSCWriteETB	Issue BSCLIB WRITE call and end block with ETB.
BSCWriteETX	Issue BSCLIB WRITE call and end block with ETX.
BSCWriteETXEOT	Issue BSCLIB WRITE call, end block with ETX, and send EOT.
BSCWriteGetFlags	Get BCB flags from most recent WRITE operation.
BSCWriteGetTimeout	Get WRITE BCB time-out set in the current BSCAWL context.
BSCWriteSendEOT	Send an EOT.
BSCWriteSendForwardAbort	Issue BSCLIB WRITE call and send a forward to abort to cancel transmission.
BSCWriteSendLineTickle	Transmit a line tickle (empty transmission to signal still online).

BSCWriteSetSelectInteval	In multi-point mode, set select interval when retry count is greater than one.
BSCWriteStoreMultiPointAddress	In multi-point mode, store single address to used with a WRITE command.
BSCUninstall	Stop all BSCLIB processes and release all BSCLIB resources.

3.3 BSCAWL Functions By BAPI Opcode

General Purpose

BSCIssueAPICall	Make a BAPI call directly.
BSCReceiveFile	Function to completely receive a file.
BSCSendFile	Function to completely send a file.

Opcode 0 - INITIALIZE

BSCCreateHandle	Get BSCAWL handle (must be first call made to BSCAWL).
BSCInitialize	Initialize BSCLIB configuration parameters.
BSCGetTranslationTable	Get pointer to current ASCII – EBCDIC translation tables.
BSCLoadSettings	Read configuration parameters from specified .ini file and save in PIB.
BSCSaveSettings	Write current configuration parameters to specified .ini file.
BSCSetTerminal ID	Store optional terminal ID.
BSCStoreTranslationTable	Store new set of ASCII – EBCDIC translation tables.
BSCClearReceivedTerminalID	Reset any saved terminal ID received.
BSCGetParms	Get pointer to current PIB.

Opcode 1 - INSTALL

BSCInstall	Install/start BSCLIB components.
BSCIsInstalled	Get BSCLIB installation status.

Opcode 2 - OPEN

BSCConnect	Raise DTR in anticipation of establishing a connection.
BSCAnswers	Put modem into auto-answer mode, and answer incoming call.

BSCDial	Place an outgoing telephone call in to establish a connection.
BSCIsOpen	Get BSCLIB line connected status.

Opcode 3 - READ

BSCRead	Issue BSCLIB READ call.
BSCReadInitiateRVI	Issue BSCLIB READ, then issue RVI requesting a line turnaround.
BSCReadToETX	Issue BSCLIB READs until ETX terminated block is received.
BSCReadUnidirectional	Issue BSCLIB READ in "simplex" or uni-direction mode.
BSCReadGetFlags	Get READ BCB flags currently set in the hBC context.
BSCReadGetTimeout	Get READ BCB time-out currently set in the hBC context.
BSCReadGetReceivedTerminalID	Get a received terminal ID.
BSCReadGetSelectedAddress	In multi-point mode, Get selected station address.
BSCReadAddSelectAddressToList	In multi-point mode, add an address to select address list.
BSCReadRemoveSelectAddress-FromList	In multi-point mode, remove an address from select address list.
BSCReadStoreMultiPointAddress	In multi-point mode, store single address to be used with a READ command.
BSCReadStoreSelectAddressList	In multi-point mode, store multiple addresses to recognize when selected.
BSCReadGetSelectAddressList	In multi-point mode, get multiple select addresses currently defined.
BSCReadSetPollInteval	In multi-point mode, set poll interval when retry count is greater than one.

Opcode 4 - WRITE

BSCWrite	Issue BSCLIB WRITE call.
BSCWriteETB	Issue BSCLIB WRITE call and end block with ETB.
BSCWriteETX	Issue BSCLIB WRITE call and end block with ETX.
BSCWriteETXEOT	Issue BSCLIB WRITE call, end block with ETX, and send EOT.
BSCWriteGetFlags	Get WRITE BCB flags currently set in the hBC context.

BSCWriteGetTimeout	Get WRITE BCB time-out currently set in the hBC context.
BSCWriteSendForwardAbort	Issue BSCLIB WRITE call and send a forward to abort to cancel transmission.
BSCWriteSendLineTickle	Transmit a line tickle (empty transmission to signal still online).
BSCWriteSendEOT	Send an EOT.
BSCWriteStoreMultiPointAddress	In multi-point mode, store single address to used with a WRITE command.
BSCWriteSetSelectInteval	In multi-point mode, set select interval when retry count is greater than one.

Opcode 5 - ABORT

BSCAbort	Abort pending OPEN, READ, WRITE operation.

Opcode 6 - STATUS

BSCStatus	Issue BSCLIB STATUS call.

Opcode 7 - STATISTICS

BSCGetStatistics	Get contents of current SPB.
BSCStoreStatistics	Store new values in current SPB.
BSCClearStatistics	Zero out contents of current SPB.

Opcode 8 - TRACE

BSCTrace	Turn BSCLIB tracing on and off.

Opcode 9 - CLOSE

BSCClose	Close communications session.

Opcode 10 – UNINSTALL

BSCUninstall	Stop all BSCLIB processes.
BSCCloseHandle	Stop/remove BSCLIB components.

Opcode 12 – HARDWARE COMMANDS

BSCHardwareCommand	Issue BSCLIB hardware command call.
BSCGetHardwareType	Get type of Serengeti hardware installed.
BSCSetDTR	Set DTR modem signal on or off.
BSCGetNumDCPBoard	In SmartSync/DCP environments, Get number of boards installed.
BSCSetBlockRespTimeout	Set block response time-out.
BSCSetMaxBaudRate	Set maximum DTE baud rate for asynchronous modem dialing.
BSCGetATInitString	Get current AT modem initialization command string.
BSCGetATAnswerString	Get the currently configured AT modem auto-answer initialization string.
BSCStoreATAnswerString	Set new AT modem command string to enable auto answer.
BSCStoreATInitString	Set new AT modem initialization command string.

3.4 BSCAWL Function Definitions

Refer to the **ssitypes.h** source module for definitions of the data types (e.g., USHORT) used in the following function definitions.

Note 1: The flags arguments in the following functions are defined as ULONG (a 32-bit value) and comprise both the BCB Flags and AuxFlags parameters. Refer to the sample programs for details on how to use BSCAWL flag arguments.

Note 2: While several of the following functions permit a callback function to be provided whereby BSCAWL notifies your application when the operation is complete, not all progra mming languages support the use of callback functions (e.g., Visual Basic, Java). In the absence of a callback, all calls into BSCAWL are blocking.

BSCAbort

Parameters:	HBSC	HBC	Pointer to BSC handle.
Returns:	USHORT	The return code from the BSCLIB call.	
BAPI Call:	Opcode 5		

Description: Aborts any pending OPEN, CLOSE, READ or WRITE command.

BSCAnswer

Parameters:	HBSC	HBC	Pointer to BSC handle
	ULONG	dwFlags	OPEN flags as defined in BCB
	USHORT	nTimeout	Number of seconds to wait for connection. If 0, wait indefinitely
	AWLPROC	CallbackProc	Callback function to notify when the connection is made. If NULL, then this will be a blocking call.
	PVOID	pCBParms	Optional pointer to be passed back to the callback, to give it context information to be used within the callback function. Set to NULL if not used.
Returns:	USHORT	The return code from the BSCLIB call.	

BAPI Call: Opcode 2, Subopcode 1

Description: Puts modem into Auto Answer mode, raise the DTR signal, and wait for the DSR signal from the modem, indicating that a connection with a remote station has been established. Calls **BSCInstall** automatically, if not already called.

BSCClearReceivedTerminalID

Parameters:	HBSC	hBC	Pointer to BSC handle.
Returns:	USHORT	The return code from the BSCLIB call.	

BAPI Call: Opcode 0, Subopcode 10

Description: Clears the memory used to store a terminal ID, if one was received.

BSCClearStatistics

Parameters:	HBSC	HBC	Pointer to BSC handle.
Returns:	USHORT	The return code from the BSCLIB call.	

BAPI Call: Opcode 7, Subopcode 0

Description: Zeros out BSCLIB's internal statistics stored in Statistics Parameter Block (SPB).

BSCClose

Parameters:	HBSC	hBC	Pointer to BSC handle.
	ULONG	dwFlags	CLOSE flags as defined in BCB.
Returns:	USHORT	The return code from the BSCLIB call.	
BAPI Call:	Opcode 9, Subopcode 0		
Description:	Closes the physical connection to the host, and ends the communications session		

BSCConnect

Parameters:	HBSC	hBC	Pointer to BSC handle.
	ULONG	dwFlags	Open flags as defined in BCB.
	USHORT	nTimeout	Number of seconds to wait for DSR. If 0, wait indefinitely.
	AWLPROC	CallbackProc	Callback function to notify when the connection is made. If NULL, then this will be a blocking call.
	PVOID	PCBParms	Optional pointer to be passed back to the callback function, to give it context information to be used within the callback.
Returns:	USHORT	The return code from the BSCLIB call.	
BAPI Call:	Opcode 2, Subopcode 1		
Description:	Raises the DTR signal and wait for the DSR signal from the modem, indicating that a connection with a remote station has been established. Calls **BSCInstall** automatically, if not already called.		

BSCCreateHandle

Parameters:	n/a	
Returns:	HBSC	Pointer to BSC handle.
BAPI Call:	n/a	
Description:	This is the first call that an application using the library must make to the library. The programmer then must call **BSCLoadSettings** to initialize PIB configuration parameters or call **BSCSaveSettings** to create a default PIB and .ini file.	

BSCDial

Parameters:	HBSC	HBC	Pointer to BSC handle.
	PCHAR	SzPhoneNumber	The phone number to dial; if not provided, the default number from the .ini file is used.
	ULONG	DwFlags	Open flags as defined in BCB.
	USHORT	NTimeout	Number of seconds to wait for connection. If 0, wait indefinitely.
	AWLPROC	CallbackProc	Callback function to notify when the connection is made. If NULL, then block until complete.
	PVOID	PCBParms	Optional pointer to be passed back to the callback, to give it context information to be used within the callback function. Set to NULL if not used.
Returns:	USHORT	The return code from the BSCLIB call.	

BAPI Call: Opcode 2, Subopcode 0

Description: Dials the modem and waits for a connection. Calls **BSCInstall** automatically, if not already called.

BSCGetATAnswerString

Parameters:	HBSC	HBC	Pointer to BSC handle.
	PCHAR	szAnswerString	Returned modem answer string.
Returns:	USHORT	The return code from the BSCLIB call.	

BAPI Call: Opcode 12, Subopcode 14

Description: Gets the currently configured AT modem auto-answer initialization string.

BSCGetATInitString

Parameters:	HBSC	HBC	Pointer to BSC handle.
	PCHAR	SzInitString	Returned modem initialization string.
Returns:	USHORT	The return code from the BSCLIB call.	

BAPI Call: Opcode 12, Subopcode 12

Description: Get the currently configured AT modem initialization string.

BSCGetHardwareType

Parameters:	HBSC	hBC	Pointer to BSC handle.
Returns:	USHORT	The return code from the BSCLIB call.	
BAPI Call:	Opcode 12, Subopcode 0		
Description:	Returns the type of hardware installed.		

BSCGetNumDCPBoards

Parameters:	HBSC	hBC	Pointer to BSC handle.
Returns:	Char	The number of DCP boards detected in the system.	
BAPI Call:	Opcode 12, Subopcode 8		
Description:	Returns the number of DCP boards detected/installed in the system.		

BSCGetParms

Parameters:	HBSC	hBC	Pointer to BSC handle.
Returns:	PPARMS	A pointer to the Parameter Initialization Block (PIB) referenced from the hBC context structure.	
BAPI Call:	n/a		
Description:	Gets a pointer to the Parameter Initialization Block (PIB) contained in the hBC context structure. Your application should initialize the PIB by way of this pointer prior to calling **BSCInstall**.		

BSCGetStatistics

Parameters:	HBSC	HBC	Pointer to BSC handle.
	PSTATS	PStats	Statistics structure to copy into.
Returns:	USHORT	The return code from the BSCLIB call.	
BAPI Call:	Opcode 7, Subopcode 1		
Description:	Gets BSCLIB's internal statistics and puts them in the passed in Statistics Parameter Block (SPB).		

BSCGetTranslationTable

Parameters:	HBSC	hBC	Pointer to BSC handle.
	ULONG	dwFlags	BCB flags to select table.

	PUCHAR	pTable	The buffer into which the requested table will be copied.
Returns:	USHORT	The return code from the BSCLIB call	

BAPI Call: Opcode 0, Subopcode 1

Description: Gets the ASCII to EBCDIC or EBCDIC to ASCII translation tables used by BSCLIB to convert incoming and outgoing data.

BSCHardwareCommand

Parameters:	HBSC	hBC	Pointer to BSC handle.
	UCHAR	nSubopcode	Hardware command subopcode.
	PUSHORT	nParm	Subopcode dependent argument.
	PCHAR	szParm	Subopcode dependent argument.
Returns:	USHORT	The return code from the BSCLIB call.	

BAPI Call: Opcode 12, user defined subopcode

Description: Issues a specified hardware command.

BSCInitialize

Parameters:	HBSC	hBC	Pointer to BSC handle.
Returns:	USHORT	The return code from the BSCLIB call.	

BAPI Call: Opcode 0, Subopcode 0

Description: Restores the BSCLIB configuration parameters to those obtained in the previous call to **BSCSetupFromINI**.

BSCInstall

Parameters:	HBSC	hBC	Pointer to BSC handle.
Returns:	USHORT	The return code from the BSCLIB call.	

BAPI Call: Opcode 1, Subopcode 0

Description: Loads the BSC protocol handler and initializes shared memory. Calls **BSCInitialize** automatically, if not already called.

BSCIsInstalled

Parameters:	HBSC	hBC	Pointer to BSC handle.
Returns:	BOOLEAN	True (non-zero) if BSCLIB is currently installed.	
BAPI Call:	n/a		
Description:	Gets the status of BSCLIB. Returns TRUE between calls to **BSCInstall** and **BSCUninstall**.		

BSCIsOpen

Parameters:	HBSC	hBC	Pointer to BSC handle.
Returns:	BOOLEAN	True (non-zero) if the line is connected (successful return from **BSCOpen**, **BSCAnswer**, or **BSCDial**.	
BAPI Call:	n/a		
Description:	Gets the connection status of BSCLIB.		

BSCIssueAPICall

Parameters:	HBSC	hBC	Pointer to BSC handle.
	PBCB	pBcb	Pointer to a BCB structure.
Returns:	USHORT	The return code from the BSCLIB call.	
BAPI Call:	Dependent on the BCB structure provided to the function.		
Description:	Issues a BAPI call.		

BSCLoadSettings

Parameters:	HBSC	hBC	Pointer to BSC handle.
	PCHAR	szIniFile	BSCAWL .ini file.
	PCHAR	szSection	Optional section name to reference in .ini file. If NULL, default section [BSCLIB] is used. May be useful to keep separate configurations for multiple ports.
Returns:	BOOLEAN	TRUE if successfully read .ini file; FALSE if failed.	
BAPI Call:	n/a		

Description: Reads the values from the given ini file and assign them to the context referenced by hBC. If the file does not exist, the function returns FALSE. Lookup the BSCAWL .ini file for more information.

BSCRead

Parameters:	HBSC	hBC	Pointer to BSC handle.
	PUCHAR	pBuf	Buffer to copy data into.
	USHORT	nReadBytes	Length of the provided buffer.
	PUSHORT	pnBytesRead	Pointer to where number of bytes received is returned on blocking calls (no callback function provided).
	PULONG	pdwFlags	Pointer to read flags as defined in BCB (both input and output).
	USHORT	nTimeout	Number of seconds to wait for data. If 0, wait indefinitely.
	AWLPROC	CallbackProc	Callback function to notify when the data is available. If NULL, then block until data is read.
	PVOID	pCBParms	Optional pointer to be passed back to the callback, to give it context information to be used within the callback function. Set to NULL if not used.
Returns:	USHORT	The return code from the BSCLIB call.	

BAPI Call: Opcode 3, Subopcode 0

Description: Reads incoming data.

BSCReadAddSelectAddressToList

Parameters:	HBSC	HBC	Pointer to BSC handle.
	PUCHAR	PBuf	Buffer containing address as a NULL terminated string.
Returns:	USHORT	The return code from the BSCLIB call.	

BAPI Call: Opcode 3, Subopcode 22

Description: In multi-point mode, stores single address to recognize when selected.

BSCReadGetFlags

Parameters:	HBSC	HBC	Pointer to BSC handle.

Returns:	ULONG	The current value for READ flags.	

BAPI Call: n/a

Description: Gets the READ flags currently set in the BSC context. This is useful for applications that cannot access the contents of the hBC context.

BSCReadGetTimeout

Parameters:	HBSC	hBC	Pointer to BSC handle.
Returns:	USHORT	The current value for READ time-out.	

BAPI Call: n/a

Description: Gets the READ time-out currently set in the BSC context. This is useful for applications that cannot access the contents of the hBC context.

BSCReadGetReceivedTerminalID

Parameters:	HBSC	HBC	Pointer to BSC handle.
	PCHAR	PBuf	Buffer to copy ID into.
	PUSHORT	pnBytesRead	Pointer to number of bytes stored in buffer.
Returns:	USHORT	The return code from the BSCLIB call.	

BAPI Call: Opcode 3, Subopcode 10

Description: Gets received Terminal ID. pBuf must point to a buffer sufficiently large to accept the maximum Terminal ID length of 20 bytes.

BSCReadGetSelectAddressList

Parameters:	HBSC	HBC	Pointer to BSC handle.
	PCHAR	Pbuf	Buffer to save a NULL terminated list of NULL terminated strings.
	USHORT	Nlen	Length of buffer provided.
Returns:	USHORT	The return code from the BSCLIB call.	

BAPI Call: n/a

Description: In multi-point mode, gets the current select address list.

BSCReadGetSelectedAddress

Parameters:	HBSC	HBC	Pointer to BSC handle.
	PCHAR	PBuf	Buffer to copy address into.
	PUSHORT	pnBytesRead	Pointer to number of bytes stored in buffer.
Returns:	USHORT	The return code from the BSCLIB call.	

BAPI Call: Opcode 3, Subopcode 11

Description: Gets Selected Station Address. pBuf must point to a buffer sufficiently large to accept the maximum Address length of 20 bytes.

BSCReadInitiateRVI

Parameters:	HBSC	HBC	Pointer to BSC handle.
	PUCHAR	Pbuf	Buffer to copy data into.
	USHORT	NReadBytes	Length of the provided buffer.
	PUSHORT	PnBytesRead	Pointer to number of bytes read into buffer.
	PULONG	PdwFlags	Pointer to READ flags as defined in BCB (both input and output).
	USHORT	NTimeout	Number of seconds to wait for data. If 0, wait indefinitely.
	AWLPROC	CallbackProc	Callback function to notify when the data is available. If NULL, then block until complete.
	PVOID	pCBParms	Optional pointer to be passed back to the callback, to give it context information to be used within the callback function. Set to NULL if not used.
Returns:	USHORT	The return code from the BSCLIB call.	

BAPI Call: Opcode 3, Subopcode 1

Description: Reads pending data, then issues an RVI requesting control of the line.

BSCReadRemoveSelectAddressFrom

Parameters:	HBSC	hBC	Pointer to BSC handle.
	PUCHAR	pBuf	Buffer containing address as a NULL terminated string.
Returns:	USHORT	The return code from the BSCLIB call.	

BAPI Call: Opcode 3, Subopcode 23

Description: In multi-point mode, removes an address from select address list.

BSCReadSetPollInterval

Parameters:	HBSC	HBC	Pointer to BSC handle.
	USHORT	NTimeout	Number of seconds to wait between sending polls in multi-point mode when configured as control station..

Returns: n/a

BAPI Call: n/a

Description: In multi-point mode, sets poll interval when retry count is greater than one. The interval is the period of time BSCLIB waits if a poll is refused or ignored before another is sent.

BSCReadStoreMultiPointAddress

Parameters:	HBSC	HBC	Pointer to BSC handle.
	PCHAR	pAddress	Buffer containing address as a NULL terminated string.
Returns:	USHORT	The return code from the BSCLIB call.	

BAPI Call: Opcode 3, Subopcode 20

Description: In multi-point mode, stores single address to recognize when selected (when a tributary station) or address to poll (when a control station). When a tributary station, calling this function overrides any address(es) set with the **BSCReadStoreSelectAddressList** function.

BSCReadStore SelectAddressList

Parameters:	HBSC	HBC	Pointer to BSC handle.
	PCHAR	PBuf	Buffer containing addresses as a NULL terminated list of NULL terminated strings.
Returns:	USHORT	The return code from the BSCLIB call.	

BAPI Call: Opcode 3, Subopcode 21

Description: In multi-point mode, stores multiple addresses to recognize when selected. Calling this function overrides any address set with the **BSCReadStoreMultiPointAddress** function.

BSCReadToETX

Parameters:	HBSC	HBC	Pointer to BSC handle.
	PUCHAR	PBuf	Buffer to copy data into.
	USHORT	nReadBytes	Length of the provided buffer.
	PUSHORT	pnBytesRead	Pointer to where number of bytes received is returned on blocking calls (no callback function provided).
	PULONG	PdwFlags	Pointer to READ flags as defined in BCB (both input and output).
	USHORT	NTimeout	Number of seconds to wait for data. If 0, wait indefinitely.
	AWLPROC	CallbackProc	Callback function to notify when the data is available. If NULL, then block until data is read.
	PVOID	pCBParms	Optional pointer to be passed back to the callback, to give it context information to be used within the callback function. Set to NULL if not used.
Returns:	USHORT	The return code from the BSCLIB call.	

BAPI Call: Opcode 3, Subopcode 4

Description: Performs standard READs until an ETX is received, at which point the application should begin writing data. Used in a conversational reply.

BSCReadUnidirectional

Parameters:	HBSC	hBC	Pointer to BSC handle
	PUCHAR	pBuf	Buffer to copy data into.
	USHORT	nReadBytes	Length of the provided buffer.
	PUSHORT	pnBytesRead	Pointer to number of bytes read into buffer.
	PULONG	pdwFlags	Pointer to READ flags as defined in BCB (both input and output).
	USHORT	nTimeout	Number of seconds to wait for data. If 0, wait indefinitely.
	AWLPROC	CallbackProc	Callback function to notify when the data is available. If NULL, then block until data is read.
	PVOID	pCBParms	Optional pointer to be passed back to the callback, to give it context information to be used within the callback function. Set to NULL if not used.
Returns:	USHORT	The return code from the BSCLIB call.	

BAPI Call: Opcode 3, Subopcode 3

Description: Performs "uni-directional" or "simplex" receives.

BSCReceiveFile

Parameters:	HBSC	hBC	Pointer to BSC handle.
	PCHAR	szFileName	Name of file to write receive data to.
	ULONG	dwFlags	READ flags as defined in BCB.
	IOPROC	CallbackProc	Callback function to notify when the receive file is complete. May be NULL if notification is not required.
	PVOID	pCBParms	Optional pointer to be passed back to the callback, to give it context information to be used within the callback function. Set to NULL if not used.
Returns:	USHORT	The return code from the BSCLIB call.	

BAPI Call: Multiple

Description: Receives a file.

BSCCloseHandle

Parameters:	HBSC	hBC	Pointer to BSC handle.

Returns: n/a

BAPI Call: n/a

Description: Releases the resources allocated by BSCLIB. This should be the last BSCAWL call.

BSCSaveSettings

Parameters:	HBSC	hBC	Pointer to BSC handle.
	PCHAR	szIniFile	BSCAWL .ini file.
	PCHAR	szSection	Optional section name to reference in .ini file. If NULL, default section [BSCLIB] is used. May be useful to keep separate configurations for multiple ports.
Returns:	BOOLEAN	TRUE if successfully read .ini file; FALSE if failed.	

BAPI Call: n/a

Description: Writes the configuration values pointed by the hBC handle to the specified file. If no file name is specified, the function returns FALSE. If the specified .ini file does not exist, the function creates a file with all default values. See Paragraph 3.5 for more information on the contents of the BSCAWL .ini file.

BSCSendFile

Parameters:	HBSC	hBC	Pointer to BSC handle.
	PCHAR	szFileName	Name of file to send.
	ULONG	dwFlags	WRITE flags as defined in BCB.
	IOPROC	CallbackProc	Callback function to notify when the send file is complete. May be NULL if notification is not required.
	PVOID	pCBParms	Optional pointer to be passed back to the callback, to give it context information to be used within the callback function. Set to NULL if not used.
Returns:	USHORT	The return code from the BSCLIB call.	

BAPI Call: Multiple

Description: Sends a file.

BSCSetBlockRespTimeout

Parameters:	HBSC	hBC	Pointer to BSC handle.
	USHORT	nValue	Time-out value in seconds.
Returns:	USHORT	The return code from the BSCLIB call.	

BAPI Call: Opcode 12, Subopcode 9

Description: Sets the block response time-out. This is the time BSCLIB waits for a block response before sending an ENQ or other appropriate action.

BSCSetDTR

Parameters:	HBSC	hBC	Pointer to BSC handle.
	BOOLEAN	bState	0=off, 1=on.
Returns:	USHORT	The return code from the BSCLIB call.	

BAPI Call: Opcode 12, Subopcode 4, Subopcode 5

Description: Raises or lowers DTR modem signal.

BSCSetMaxBaudRate

Parameters:	HBSC	hBC	Pointer to BSC handle.
	USHORT	nValue	12=1200 bps; 24=2400 bps; 48=4800 bps; 96=9600 bps; 19=19200 bps; 38=38,400 bps
Returns:	USHORT	The return code from the BSCLIB call.	

BAPI Call: Opcode 12, Subopcode 10

Description: Set maximum DTE baud rate for asynchronous modem dialing.

BSCSetTerminalID

Parameters:	HBSC	hBC	Pointer to BSC handle.
	PCHAR	pID	Buffer containing the desired terminal ID (up to 20 characters).
Returns:	USHORT	The return code from the BSCLIB call.	

BAPI Call: n/a

Description: Sets the terminal ID used in point-to-point connections.

BSCStatus

Parameters:	HBSC	hBC	Pointer to BSC handle.
	PUCHAR	pActiveOpcode	Pointer to currently active Opcode
Returns:	USHORT	The return code from the BSCLIB call.	

BAPI Call: Opcode 6

Description: Retrieves the status of a pending OPEN, ABORT, or WRITE command.

BSCStoreATAnswerString

Parameters:	HBSC	hBC	Pointer to BSC handle.
	PCHAR	szAnswerString	Modem answer string to set.

Returns:	USHORT	The return code from the BSCLIB call.	
BAPI Call:	Opcode 12, Subopcode 15		
Description:	Sets the AT modem answer initialization string.		

BSCStoreATInitString

Parameters:	HBSC	hBC	Pointer to BSC handle.
	PCHAR	szInitString	Modem initialization string to set.
Returns:	USHORT	The return code from the BSCLIB call.	
BAPI Call:	Opcode 12, Subopcode 13		
Description:	Sets the AT modem initialization string.		

BSCStoreStatistics

Parameters:	HBSC	hBC	Pointer to BSC handle.
	PSTATS	pStats	Statistics structure to store.
Returns:	USHORT	The return code from the BSCLIB call.	
BAPI Call:	Opcode 7, Subopcode 0		
Description:	Sets BSCLIB's internal statistics from the passed in Statistics Parameter Block (SP B).		

BSCStoreTranslationTable

Parameters:	HBSC	hBC	Pointer to BSC handle.
	ULONG	dwFlags	BCB flags to select table.
	PUCHAR	pTable	The buffer from which the supplied table will be copied.
Returns:	USHORT	The return code from the BSCLIB call.	
BAPI Call:	Opcode 0, Subopcode 2		
Description:	Stores the ASCII to EBCDIC or EBCDIC to ASCII translation tables used by BSCLIB to convert incoming and outgoing data.		

BSCTrace

Parameters:	HBSC	hBC	Pointer to BSC handle.
	BOOLEAN	bTraceON	Whether to start or stop the trace.
	PCHAR	szTraceFile	Path of trace file to use.
Returns:	USHORT	The return code from the BSCLIB call.	
BAPI Call:	Opcode 8, various subopcodes		
Description:	Depending on the value of bTraceON, BSCTrace either starts tracing to szTraceFile or stops the active trace.		

BSCWrite

Parameters:	HBSC	hBC	Pointer to BSC handle.
	PUCHAR	pBuf	Buffer to send.
	USHORT	nBytesToWrite	Number of bytes to send.
	PUSHORT	pnBytesWritten	Pointer to where number of bytes sent is returned on blocking calls (no callback function provided).
	PULONG	pdwFlags	Point to WRITE flags as defined in BCB (both input and output).
	USHORT	nTimeout	Number of seconds to wait for data to be transmitted. If 0, wait indefinitely.
	AWLPROC	CallbackProc	Callback function to notify when the data has been sent. If NULL, then block until complete.
	PVOID	pCBParms	Optional pointer to be passed back to the callback, to give it context information to be used within the callback function. Set to NULL if not used.
Returns:	USHORT	The return code from the BSCLIB call.	
BAPI Call:	Opcode 4, Subopcode 0		
Description:	Adds the data as a record in the current transmit buffer (block).		

BSCWriteETB

Parameters:	HBSC	hBC	Pointer to BSC handle.
	PUCHAR	pBuf	Buffer to send.
	USHORT	nBytesToWrite	Number of bytes to send.

	PUSHORT	pnBytesWritten	Pointer to where number of bytes sent is returned on blocking calls (no callback function provided).
	PULONG	pdwFlags	Point to WRITE flags as defined in BCB (both input and output).
	USHORT	nTimeout	Number of seconds to wait for data to be transmitted. If 0, wait indefinitely.
	AWLPROC	CallbackProc	Callback function to notify when the data has been sent. If NULL, then block until complete.
	PVOID	pCBParms	Optional pointer to be passed back to the callback, to give it context information to be used within the callback function. Set to NULL if not used.

Returns: USHORT The return code from the BSCLIB call.

BAPI Call: Opcode 4, Subopcode 1

Description: Adds the data as a record in the current transmit buffer then sends the buffer as an ETB terminated block.

BSCWriteETX

Parameters:	HBSC	hBC	Pointer to BSC handle.
	PUCHAR	pBuf	Buffer to send.
	USHORT	nBytesToWrite	Number of bytes to send.
	PUSHORT	pnBytesWritten	Pointer to where number of bytes sent is returned on blocking calls (no callback function provided).
	PULONG	pdwFlags	Point to WRITE flags as defined in BCB (both input and output).
	USHORT	nTimeout	Number of seconds to wait for data to be transmitted. If 0, wait indefinitely.
	AWLPROC	CallbackProc	Callback function to notify when the data has been sent. If NULL, then block until complete.
	PVOID	pCBParms	Optional pointer to be passed back to the callback, to give it context information to be used within the callback function. Set to NULL if not used.

Returns: USHORT The return code from the BSCLIB call

BAPI Call: Opcode 4, Subopcode 2

Description: Adds the data as a record in the current transmit buffer then sends the buffer as an ETX terminated block.

BSCWriteETXEOT

Parameters:	HBSC	hBC	Pointer to BSC handle.
	PUCHAR	pBuf	Buffer to send.
	USHORT	nBytesToWrite	Number of bytes to send.
	PUSHORT	pnBytesWritten	Pointer to where number of bytes sent is returned on blocking calls (no callback function provided).
	PULONG	pdwFlags	Point to WRITE flags as defined in BCB (both input and output).
	USHORT	nTimeout	Number of seconds to wait for data to be transmitted. If 0, wait indefinitely.
	AWLPROC	CallbackProc	Callback function to notify when the data has been sent. If NULL, then block until complete.
	PVOID	pCBParms	Optional pointer to be passed back to the callback, to give it context information to be used within the callback function. Set to NULL if not used.
Returns:	USHORT	The return code from the BSCLIB call	

BAPI Call: Opcode 4, Subopcode 3

Description: Adds the data as a record in the current transmit buffer then sends the buffer as an ETX terminat ed block followed by an EOT. This call will terminate the WRITE operation upon completion. **BSCWriteETXEOT** must be called once to terminate a WRITE.

BSCWriteGetFlags

Parameters:	HBSC	hBC	Pointer to BSC handle.
Returns:	ULONG	The current value for WRITE flags.	

BAPI Call: n/a

Description: Gets the WRITE flags currently set in the BSC context. This is useful for applications that cannot access the contents of the hBC context.

BSCWriteGetTimeout

Parameters:	HBSC	hBC	Pointer to BSC handle.
Returns:	USHORT	The current value for WRITE time-out.	

BAPI Call: n/a

Description: Gets the WRITE time-out currently set in the BSC context. This is useful for applications that cannot access the contents of the hBC context.

BSCWriteSendEOT

Parameters:	HBSC	hBC	Pointer to BSC handle.
	ULONG	dwFlags	Write flags as defined in BCB.
Returns:	USHORT	The return code from the BSCLIB call.	
BAPI Call:	Opcode 4, Subopcode 11		
Description:	Transmit an EOT.		

BSCWriteSendForwardAbort

Parameters:	HBSC	hBC	Pointer to BSC handle.
	ULONG	dwFlags	Write flags as defined in BCB.
Returns:	USHORT	The return code from the BSCLIB call.	
BAPI Call:	Opcode 4, Subopcode 4		
Description:	Issues WRITE call and send forward abort to cancel transmission.		

BSCWriteSendLineTickle

Parameters:	HBSC	HBC	Pointer to BSC handle.
	ULONG	dwFlags	Write flags as defined in BCB.
Returns:	USHORT	The return code from the BSCLIB call.	
BAPI Call:	Opcode 4, Subopcode 10		
Description:	Sends a “line tickle” which is an empty transmission to indicate to the remote that the BSCLIB application is connected and alive.		

BSCWriteSetSelectInterval

Parameters:	HBSC	hBC	Pointer to BSC handle.
	USHORT	nTimeout	Number of seconds to wait between sending selects in multi-point mode when configured as control station..
Returns:	n/a		
BAPI Call:	n/a		

Description: In multi-point mode, sets select interval when retry count is greater than one. The interval is the period of time BSCLIB waits if a select is refused or ignored before another is sent.

BSCWriteStoreMultiPointAddress

Parameters:	HBSC	HBC	Pointer to BSC handle.
	PCHAR	pAddress	Buffer containing address as a NULL terminated string.
Returns:	USHORT	The return code from the BSCLIB call.	

BAPI Call: Opcode 4, Subopcode 20

Description: In multi-point mode, stores single address to select (when a control station) or a poll address (when a tributary station).

BSCUninstall

Parameters:	HBSC	HBC	Pointer to BSC handle.
Returns:	USHORT	The return code from the BSCLIB call.	

BAPI Call: Opcode 10

Description: Stops all BSCLIB processes. This should always be the last BSCAWL function called by your application.

3.5 BSCAWL Callback Functions

When non-blocking calls are made to the BSCLIB API Wrapper Library (AWL) on opens, reads, or writes, a callback function mechanism is provided to enable BSCAWL to notify the application of events. All callback functions pass back the same arguments, but not all are used on every call. The four fundamental callback functions are described below.

Note: Not all programming languages support the use of callback functions.

IOCallbackProc

Parameters:	USHORT	nErrorCode	BSCLIB return code.
	PVOID	pCBParms	Pointer to callback parameters structure supplied to the originating call to the **BSCSendFile**, or **BSCReceiveFile** function.
Returns:	void		

Description: Optional user defined callback function called when a **BSCSendFile** or **BSCReceiveFile** function completes.

OpenCallbackProc

Parameters:	HBSC	hBC	Pointer to BSC handle.
	USHORT	nErrorCode	BSCLIB return code.
	ULONG	dwFlags	BCB flags returned from BSCLIB.
	USHORT	nCount	(not used)
	void *	pBuffer	(not used)
	PVOID	pCBParms	Pointer to callback parameters structure supplied to the originating call to the **BSCConnect**, **BSCDial**, or **BSCAnswer** function.
Returns:	void		

Description: User defined callback function called when a **BSCConnect**, **BSCDial**, or **BSCAnswer** function completes.

ReadCallbackProc

Parameters:	HBSC	hBC	Pointer to BSC handle.
	USHORT	nErrorCode	BSCLIB return code.
	ULONG	dwFlags	BCB flags returned from BSCLIB.
	USHORT	nCount	Number of bytes in pBuffer.
	PUCHAR	pBuffer	Pointer to buffer contained read data.
	PVOID	pCBParms	Pointer to callback parameters structure supplied to the originating call to the **BSCRead** function.
Returns:	void		

Description: User defined callback function called when a the **BSCRead** function returns data, an error occurs, or the function completes.

WriteCallbackProc

Parameters:	HBSC	hBC	Pointer to BSC handle.
	USHORT	nErrorCode	BSCLIB return code.
	ULONG	dwFlags	BCB flags returned from BSCLIB.

	USHORT	nCount	Number of bytes written.
	void *	pBuffer	(not used)
	PVOID	pCBParms	Pointer to callback parameters structure supplied to the originating call to the **BSCWrite** function.
Returns:	void		
Description:	User defined callback function called when a the **BSCWrite** function need more data, an error occurs, or the function completes.		

See the **bscawl_defs.h** source module for the definition of the PVOID structure.

3.6 BSCAWL ".ini" File Definitions

BSCAWL is configured by way of an ".ini" file. This file is manipulated using the **BSCSaveSettings** and **BSCLoadSettings** functions or may be modified manually as required. The following parameters are under a default section entitled **[BSCLIB]**.

The section names are user definable and may be used to maintain separate configuration settings for multiple environments and/or multiple ports. For example, multiple port configurations could be maintained under titles such as **[Port 1]**, **[Port 2]**, etc.

The use of the BSCAWL ".ini" file is optional. If you wish to set the parameters in the BSC context directly without the use of the file, you may do so.

Value	Default	Description
OpenTimeout	20	Default time-out used on **BSCOpen** function calls (expressed in seconds). "0" is an infinite time-out.
ReadWriteTimeout	20	Default time-out used on **BSCRead** and **BSCWrite** function calls (expressed in seconds). "0" is an infinite time-out.
AutoDialNumber	-	Default phone number to use in **BSCDial** function .
AutoAnswer	No	Enables auto answer mode on **BSCOpen** function.
Emulation	3780	Select "2780" or "3780" protocol emulation when sending data. (The protocol is automatically detected and handled when receiving.)
StationType	Primary	Select "Primary" or "Secondary" station type.

ModemType	Other	Select default modem type. Choices are "UDS", "SADL", "AT Command Set", "V25bis", or "Other".
LineType	Switched	Select "Switched" or "Leased" line type.
Duplex	Full	Select "Full" or "Half" duplex.
MaxAsyncBaudRate	96	Set the default value for use with the **BSCSetMaxBaudRate** function.
MaxXmtBufrSize	512	Set the maximum transmit buffer size. Values up to 4096 are permitted.
MaxRcvBufrSize	520	Set the maximum receive buffer size. Values up to 4096 are permitted.
RdrRecSize	512	Set maximum reader record size. Values up to MaxXmtBufrSize are permitted.
PtrRecSize	512	Set maximum printer record size. Values up to MaxRcvBufrSize are permitted.
PunRecSize	80	Set maximum punch record size. Values up to MaxRcvBufrSize are permitted.
RecsBlk	0	Set maximum reader records per transmission block. "0" means insert as many that fit.
NoActivityTimeout	0	Set the no activity time-out (expressed in seconds). "0" is an infinite time-out.
BidRetryLimit	15	Set the bid retry limit. "0" results in an infinite number of retries.
EnqRetryLimit	6	Set the ENQ retry limit. "0" results in an infinite number of retries.
NakRetryLimit	6	Set the NAK retry limit. "0" results in an infinite number of retries.
TerminalID	-	Specify up to 20 characters for a default terminal ID.
XmtTransparentMode	No	Set default transmission mode to be EBCDIC transparent.
XmtBinaryMode	No	Set default transmission mode to be binary transparent.
XmtCompressSpaces	No	Set default transmission mode to compress spaces (in 3780 mode); to truncate spaces (in 2780 mode).
XmtSuppressTranslation	No	Set default transmission mode to suppress ASCII to EBCDIC translation.
XmtSuppressIRS	No	Set default transmission mode to suppress transmission of IRS record separators.
XmtTimeout	20	Default time-out used on **BSCWrite** function calls (expressed in seconds). "0" is an infinite time-out.
RcvBinaryMode	No	Set default reception mode to be binary transparent.
RcvExpandSpaces	Yes	Set default reception mode to expand compressed spaces (in 3780 mode).

RcvSuppressVFConETBETX	No	Set default reception mode to ignore VFC found immediately before an end-of-block control character.
RcvDecodeVFC	Yes	Set default reception mode to recognize VFC sequences.
RcvStripVFC	Yes	Set default reception mode to strip unrecognized VFC sequences.
RcvSuppressTranslation	No	Set default reception mode to suppress EBCDIC to ASCII translation.
RcvRecognizeNL	No	Set default reception mode to recognize EBCDIC NL (NewLine) character.
RcvSuppressIRSIUS	No	Set default reception mode to suppress recognition of IRS/IUS record separators.
RcvAbortPrevention	No	Enable/disable abort prevention mode when calling the **BSCRead** function
RcvTimeout	20	Default time-out used on **BSCRead** function calls (expressed in seconds). "0" is an infinite time-out.
SleepPeriod	10	Set internal delay time-out value (expressed in milliseconds). This value should not be changed except at the suggestion of a Serengeti Systems support engineer.
MutexFailsafeTimeout	10	Set internal delay time-out value (expressed in milliseconds). This value should not be changed except at the suggestion of a Serengeti Systems support engineer.
PollAddress	-	In multi-point mode, the default poll address.
SelectAddress	-	In multi-point mode, the default select address.
TraceBufferSize	32,000	Set trace buffer size for non-SmartSync/DCP environments.
Debug	No	Enable/disable BSCAWL debug log file.
DebugFileName	-	Specify the BSCAWL debug log file name.

4 LOW-LEVEL BAPI PROGRAMMING

In addition to BSCAWL, BSCLIB provides a low-level API referred to as BAPI (BSCLIB API). Use of BAPI requires bit-level manipulation of data structures to perform BSCLIB functions and generally requires more code to achieve the same results as BSCAWL functions. On the other hand, BAPI provides access to certain BSCLIB capabilities that BSCAWL does not, such as multi-point capabilities and unidirectional receive functionality.

Commands and data are passed between your program and BAPI via a data structure called the BSC Control Block (BCB). Your program calls BAPI with a properly configured BCB to perform the following functions:

- initialize configuration
- install the driver
- issue auto-dial modem commands
- issue modem setup commands
- open communications
- enable receive and read inbound records
- enable transmit and write outbound records
- abort I/O
- check I/O status
- initialize or read statistics
- control line trace
- close communications
- uninstall the driver

Typically, your program opens the communications link, reads and writes data records as required, then closes the link. Except for error processing such as recovering from an abnormal line disconnect, your program does not need to be concerned with the details of the BSC protocol and how the information is physically transferred.

BAPI provides a single entry point through which your program controls and accesses the BSC communications link. Calls into BAPI pass information via the BSC Control Block (BCB) structure. Your program passes a command, sub-command (function), and optional parameters to BSCLIB each time a BSCLIB operation is to be performed. The command, function, and its optional parameters are inserted into the BCB before issuing the subroutine call that transfers control to BSCLIB.

BAPI copies the BCB to local storage, then decodes the opcode byte in the BCB to determine which operation to perform. The specified operation is

performed by the appropriate record manager routine. When the operation is complete, control is returned to the application interface, the local BCB is copied back to your program's BCB and control returns to your program.

The BAPI entry point function is **bscllif**. The following C code sample shows how BSCLIB may be called:

```
#define OPEN  2
extern void bscllif (struct bcb *);
extern struct bcb_type bcb;

bcb.opcode = OPEN;
bscllif(&bcb);
```

Unix BAPI applications need to link the static library, **libbsc.a**, installed in **/usr/lib**. Applications using the SmartSync/DCP board use **libdcp.a** instead. To link these libraries, add the **-lbsc** or **-ldcp** option to your link command.

Windows BAPI applications will need to link to **rmlib.dll** using the **rmlib.lib** export library. In addition, **rmlib.dll** will load either **xbsc.dll** or **xdcpdrvr.dll**, depending on the communication adapter being used.

4.1 CTEST -- BAPI Sample Application

A BAPI sample application, **ctest**, and its complete C source code are provided with the BSCLIB Software Development Kit. This program exercises most of the BAPI functions. Source code is located in the **samples/ctest** subdirectory under the install directory.

For Unix development, a make file, **Makefile**, is provided to compile and link the program. This make file demonstrates the correct compiler and link options to use when developing a program that uses BSCLIB. To build the sample program, simply run **make** from the **samples/ctest** subdirectory.

For Windows development, project files compatible with Microsoft Visual Studio 6 are provided to compile and build various versions of the program. The project file is located in the **samples/ctest** subdirectory.

The resulting program is useful as both a demonstration and test program for BSCLIB in a point-to-point or contention mode BSC environment. (Compiler switches allow you to build a version that is for multi-point BSC environments, as well.) You can exercise virtually all of BSCLIB's functionality from menus and prompts within the program. All inputs to the

BAPI are displayed before each call and the BSCLIB result codes are displayed immediately after a call returns. This program is a valuable learning aid for new BSCLIB programmers who elect to use BAPI.

For applications using the SmartSync/DCP adapter, the following command line options apply:

-b *board* **Board number [1-6] (SmartSync/DCP only)**

The **-b** switch specifies the SmartSync/DCP adapter, if more than one is installed. You must indicate the board number, 1 through 6, as appropriate. Each boards number corresponds to the order it was configured. If omitted, the default is board 1.

-p *port* **Port number [1-8]**

The **-p** switch specifies the port to access. If omitted, the default is port 1.

4.2 BSCLIB Control Block (BCB)

Below is a diagram of BSCLIB Control Block (BCB) data layout. The following section provides detailed explanation of each field and its options.

Byte Offset	Field Description	
0	Opcode	Return Code
2	Subopcode	State/Type
4	Flags	
6	32-Bit Buffer Address	
8		
10	Parms/Write/Read Value	
12	Write/Read Count	
14	Time-out	
16	Aux Flags/Shared Mem ID	
18	Trace Flags	Raw Value
20	Byte #1	Byte #2
22	Port	Board
24	Tail (-1 = FFFFh)	

Figure 7. BCB Structure

IMPORTANT

BSCLIB expects the member alignment of this structure (and all BSCLIB structures) to be 8 bytes. Be sure to set your compiler this way or the structure you define in your application may not align with what BSCLIB expects.

4.2.1 BSCLIB Control Block (BCB) Definitions

The opcodes, subopcodes, and flags passed via the BCB are described below:

Opcode	Command	Subopcode	Function Description
0	INITIALIZE	0	Store Parameters
		1	Read Translation Table
		2	Store Translation Table
		10	Clear Received Terminal ID
1	INSTALL	-	Install BSCLIB
2	OPEN	0	Dial Modem
		1	Open Communications
3	READ	0	Standard
		1	Abort Read
		2	Initiate RVI
		3	Uni-Directional Standard Read
		4	Read To ETX Block (For Sending Conversational Replies)
		10	Get Received Terminal ID
		11	Get Selected Station Address
		20	Store Multi-Point Poll/Select Address
		21	Store Multiple Select Address List
		22	Add Select Address To List
		23	Remove Select Address From List
4	WRITE	0	Standard
		1	Send ETB Block
		2	Send ETX Block
		3	Send ETX Block and EOT
		4	Send Forward Abort
		10	Send Line Tickle
		11	Send EOT
		20	Store Multi-Point Poll/Select Address
5	ABORT	-	
6	STATUS	-	

Opcode	Command	Subopcode	Function Description
7	STATISTICS	0	Initialize SPB
		1	Read SPB
		10	Set 'TTDs Sent' field in SPB to count TTDs received
		11	Reset 'TTDs Sent' field in SPB to count TTDs transmitted
8	TRACE	0	Initialize
		1	Start
		2	Stop
		3	Read Queue
		4	Reset
9	CLOSE	0	Standard
		1	Leave DTR On
10	UNINSTALL	-	
11	(Reserved)	-	
12	HARDWARE COMMANDS	0	Return Hardware Type
		1	Return Driver Serial Number
		4	Raise DTR Modem Signal
		5	Drop DTR Modem Signal
		8	Return Number of DCP Adapters
		9	Set Block Response Time-Out
		10	Set Maximum Async Baud Rate
		12	Get AT Dial Command String
		13	Store AT Dial Command String
		14	Get AT Answer Command String
		15	Store AT Answer Command String
		16	Restore Default Command Strings
		20	Enable Buffered Reads
		21	Disable Buffered Reads
		22	Set Select Response to NAK
		23	Set Select Response to EOT
		24	Set Select Response to WACK
		30	Turn on BAPI debug loggiing
		31	Turn off BAPI debug loggiing
		32	Enable detailed BAPI logging
		33	Disabled detailed BAPI logging
		34	Set max BAPI debug log file size

Flags Summary:

Bit 0 –	Non-blocking I/O
Bit 1 –	Enable EBCDIC Transparency (WRITE) Transparent Data Detected (READ)(*)
Bit 2 –	Perform Binary I/O
Bit 3 –	Skip Async Initialization
Bit 4 –	Select Remote Punch (WRITE) Local Punch Selected (READ) (*)
Bit 5 –	Compress Write/Expand Read Spaces
Bit 6 –	Suppress VFC at ETB/ETX (READ)
Bit 7 –	Leave IRS at ETB/ETX (WRITE) Decode VFC (READ)
Bit 8 –	Strip VFC
Bit 9 –	Record Truncated to Buffer Size (READ) (*)
Bit 10 -	Record Truncated to Printer/Punch Size (READ) (*)
Bit 11 –	Forced Close
Bit 12 –	Suppress DLE-EOT (CLOSE) Suppress Block Responses (Uni-Directional READ)
Bit 13 –	Table Select: ASCII or EBCDIC (INITIALIZE) Do Not Discard NAK'd Blocks (Uni-Directional READ) Enable Record Truncation When in 3780 Mode (WRITE)
Bit 14 –	Ignore Record (WRITE) Block Acknowledgement Control (READ)
Bit 15 –	Last Record Before ETX (READ) (*) Retransmit Block After NAK on ENQ (WRITE)

Auxiliary Flags Summary:

Bit 0 –	Enable/Disable Printer
Bit 1 –	Enable/Disable Punch
Bit 2 –	Master Device Control
Bit 3 –	Enable/Disable ASCII↔EBCDIC Translation
Bit 4 –	Recognize EBCDIC New Lines
Bit 5 –	Suppress IRS on Write
Bit 6 –	Suppress IRS/IUS on Read
Bit 7 -	Start Outbound Block With SOH (WRITE) Record Immediately Follows SOH (READ) (*)
Bit 8 –	Residual Transmit Record Pending (*)
Bit 9 –	Enable Pulse Dial (Auto-Dial OPEN) Monitor Ringing (Auto-Answer OPEN)
Bit 10 –	Trial Open (OPEN) Enable Automatic 3780/2780 Protocol Detection (READ)

Bit 11 –	Ignore DSR (CLOSE) Conversational Reply Allowed (WRITE) 2780 Protocol Transmission Detected (READ)(*)
Bit 12 –	Enable Auto-Answer / Manual Dial
Bit 13 -	No Modem Reply Expected Last Record in Data Block Before ETB/ETX (READ)(*)
Bit 14 -	Disable ABORT Receive Delay
Bit 15 -	Disable ABORT if Receive Data is Present (READ) Leave DTR modem signal high (CLOSE)

(*) Note: Flags marked with (*) are *set by* BSCLIB and should be interpreted by your program when BSCLIB returns control. If you wish to use flags not marked, your program should set them prior to calling BSCLIB.

All calls to BSCLIB share the following BCB fields. The descriptions of BSCLIB command and functions that follow otherwise show only those BCB fields that apply specifically to the particular operation.

Port/Adapter:

The port and adapter the call refers to. In single-port applications, these fields should always be set to zero. In multi-port SyncPCI applications, the port field should correspond to a valid SyncPCI (one to four) port. In SmartSync/DCP applications, the port field should correspond to a port on a given adapter and the adapter should correspond to a valid SmartSync/DCP adapter (one to six).

Return Code:

See Appendix A for Return Codes and definitions.

4.2.2 Return Codes

Each BAPI function call returns a return code in byte 1 of the BCB. Specific error or failure codes will vary for each function call, but a small number of return codes are common to many functions.

For example:

0	Successful completion
255 (-1)	Function still in progress

Depending on the need for robustness, a BSCLIB application can be written to examine only a very small number of return codes, simply failing (with an error message) if any other value is received. Or it can be written to

attempt recovery from the widest range of potential return codes. Some of the more commonly encountered return codes are described in this chapter for the more complex and performance-critical commands: OPEN, READ and WRITE. A complete list of Return codes is included in Appendix A.

4.2.3 Blocking vs. Non-Blocking BAPI Calls

Your program may issue two types of BAPI calls to BSCLIB: blocking and non-blocking (referred to as execute and initiate I/O respectively in previous versions of BSCLIB). The selection of one versus the other is determined by the complexity of your program.

A blocking call performs its function entirely before return control to the calling program, thus "blocking" the calling program during that time. A non-blocking call returns control to the calling program immediately, but continues to process in parallel with the calling program.

The choice of blocking or non-blocking, for any call for which it is supported, is made in Bit 0 of the BCB Flags field (bytes 4 and 5).

0 = Blocking I/O
1 = Non-Blocking I/O

Non-blocking calls would add minimal benefit for simple status checks and/or configuration changes, and this bit flag is not applicable for many BSCLIB opcodes. Five BAPI function calls (OPEN, CLOSE, READ, WRITE and ABORT) can be issued as either blocking or non-blocking. Non-blocking calls can provide significant performance improvement when completion of the call may require time-consuming data transmission or be delayed by the status of a modem or a program running in another computer. Non-blocking calls are of particular benefit when this process or another port could be making beneficial use of the time.

After a non-blocking call is issued, it will be the program's responsibility to periodically test its status with a STATUS function call (or, in the case of a READ function, with another READ). A return code of 255 (-1) indicates that the call is still in processing. When it completes, you should receive either a return code of 0, indicating successful completion, or some other value, indicating a failure and its reason.

Non-blocking BAPI function calls are preferable in production applications because a program is not suspended while waiting for BSCLIB operations to complete.

4.2.4 Record-Oriented Interface

BSCLIB's READ and WRITE commands are record sequential. Reading and writing to/from a BSC connection is analogous to performing file reads and writes. Data is normally passed as *logical records* that could be equated to a line from a video display, a record within a text file, or a print line. This enables you to work with BSCLIB similarly to how you work with other common I/O routines.

Both READ and WRITE can support *physical record I/O* as well. In this mode, your program is responsible for the blocking and deblocking of logical records within a communication buffer, which is passed between your program and BSCLIB with a single call. This mode is useful for sophisticated applications that require nonstandard data formats.

You will need to be familiar with the concept of "blocks" and "records" in BSC communications in order to understand the descriptions of the READ and WRITE functions. In BSC communications, a "block" of data is a physical transmission that must be acknowledged (ACK) by its recipient before another block will be transmitted.

A block is a data transfer “packet” begun and ended by one or two characters special protocol characters. Blocks begin with a STX (or DLE STX for transparent data) and end with ETB (or DLE ETB), which indicates end of block, additional blocks follow, or with ETX (or DLE ETX), which indicates end of text and end of file. When working with logical records, the BSCLIB developer does not necessarily need to be aware of the physical end of a block since BSCLIB detects and handles them appropriately.

A block contains one or more "logical records". Logical records may be formatted in either of two ways: fixed length (as agreed upon by both the sender and the recipient) or variable length (end of record is signaled by another special character, inter-record separator, or IRS in 3780 mode or IUS in 2780 mode). Again, when working with logical records, the developer does not need to be aware of record separators since BSCLIB handles them automatically based on values configured in the Parameter Initialization Block (PIB) set by the INITIALIZE function call.

4.3 Point-to-Point vs. Multi-Point Operation

BSCLIB supports communications in either point-to-point or multi-point mode. In multi-point mode, a further distinction is made between tributary operation and control station mode.

Point-to-point means this single station communicates via modem or modem eliminator with one other station, whether in the next room or across a long distance telephone line. There are no "identity" considerations; the connection itself is the link that determines who the parties to the conversation will be. Point-to-point connections are typically what are used in Remote Job Entry (RJE) 2780 and 3780 terminal emulation environments.

In multi-point communications, a single station, called the control station, manages communications with anywhere from 1 to 32 other stations, called tributary stations. The tributaries do not communicate with one another, but only with the control station. This concept is also called a "master-slave" relationship, in which the control station is the "master" and the tributaries are the "slaves". In multi-point communications, data to or from any single tributary is nonetheless physically transmitted to and seen by all of the tributaries. Therefore, additional information is required in each transmission to identify the tributary that is the source or intended destination. Multi-point connections are typically used in 3270 display station terminal emulation environments.

Point-to-point and multi-point BSC environments are mutually exclusive. The vast majority of BSCLIB applications are of the point-to-point variety.

4.4 BSCLIB API (BAPI) Commands

The BAPI command set is described in the following subsections.

4.4.1 Opcode 0 - INITIALIZE Command

Prerequisite Calls:

None

Subopcodes:

0 – Store Parameter Initialization Block (PIB)
1 – Read Translation Table
2 – Store Translation Table
10 – Clear Received Terminal ID

Input BCB Fields:

Byte 0	Opcode
Byte 2	Subopcode
Byte 4,5	Flags
Byte 6-9	32-Bit Buffer Address

Byte 10,11	Buffer Size
Byte 22	Port (SyncPCI and SmartSync/DCP)
Byte 23	Adapter (SmartSync/DCP)

Returned BCB Fields:

Byte 1	Return Code

4.4.1.1 Subopcode 0 - Store PIB

The Parameter Information Block (PIB) is a structure containing BSCLIB configuration parameters. Its layout is shown in the figure below and described in detail in the next section.

The Store PIB function of the INITIALIZE command is used to change the BSCLIB configuration Parameters. This function is typically made prior to issuing an INSTALL command.

Not all configuration Parameters can be set once an INSTALL has been performed. However, by setting Bit 14 of the PIB Flags, your program can issue this function at any time to change record sizes, time-outs, and retry limits (any Parameter in bytes 16-33.) The remaining Parameters take effect only when this function is called with BSCLIB uninstalled.

Buffer Address:

The Buffer Address field of the BCB points to the PIB Structure.

Buffer Size:

Buffer Size field contains the size of the optional terminal ID string that may be contained within the PIB.

Byte Offset	Field Description	
0	Flags	
2	Modem Type	(Reserved)
4	(Reserved)	(Reserved)
6	Transmit Block Size	
8	Receive Block Size	
10	(Reserved)	
12	(Reserved)	
14		
16	Reader Record Size	
18	Printer Record Size	
20	Punch Record Size	
22	Records Per Block	
24	No Activity Time-out	
26	Inter-Character Time-out	
28	Bid Retry Limit	
30	ENQ Retry Limit	
32	NAK Retry Limit	
34	ID byte #1	ID byte #2
52	ID byte #19	ID Byte #20
54	Tail (-1 = FFFFh)	

Figure 8. PIB Structure

Flags:

Bit #	Function	Bit = 0	Bit = 1
0	Emulation Mode	2780	3780
1	Station Type (if Multi-Point)	Secondary Tributary	Primary Control
2	Line Type	Leased	Switched
3	Terminal ID	Disabled	Enabled
4	Reserved		
5	Duplex	Half	Full
6	Reserved		
7	Auto DLE-EOT Read	Disabled	Enabled
8	Data Link Control	EBCDIC	ASCII
9	Redundancy Test	CRC-16	LRC
10	Multi-Point Mode	No	Yes
11	Parity for ASCII DLC and Multi-Point Mode	None	Odd
12	Recognize Select on WRITE	No	Yes
14	Partial Update	No	Yes

Modem Types:

1 – Motorola/UDS models 201C/D, 208B/D, 2140, or 2860
2 – manual dial, other, or none
3 – Racal-Vadic model 4850PA
5 – AT-command auto-dial (e.g., Hayes Optima)
8 – V.25bis auto-dial

4.4.1.2 Subopcode 1 - Read Translation Table

The Read Translation Table function of the INITIALIZE command is used to read the ASCII to EBCDIC or EBCDIC to ASCII translation tables used by BSCLIB to convert incoming and outgoing data. The translation table may be read anytime except when a READ or WRITE command is active.

Flags:

Bit 13 – Table Select:

0 = EBCDIC to ASCII
1 = ASCII to EBCDIC

If Table Select = 1, BSCLIB copies the ASCII to EBCDIC table into your program's buffer, otherwise the EBCDIC to ASCII table is copied.

Buffer Address:

The Buffer Address field of the BCB contains a pointer to a 256-byte buffer to which BSCLIB can copy the requested translation table.

Buffer Size:

The value passed in the Buffer Size field of the BCB specifies the size in bytes of the buffer pointed to by the Buffer Address field. This value must be 256.

4.4.1.3 Subopcode 2 - Store Translation Table

The Store Translation Table function of the INITIALIZE command is used to update the ASCII to EBCDIC or EBCDIC to ASCII translation tables used by BSCLIB to convert incoming and outgoing data. The translation table may be stored anytime that a READ or WRITE command is not active.

Flags:

Bit 13 – Table Select:

0 = EBCDIC to ASCII
1 = ASCII to EBCDIC

If Table Select = 1, BSCLIB copies your program's buffer into the ASCII to EBCDIC table, otherwise the EBCDIC to ASCII table is updated.

Buffer Address:

The Buffer Address field of the BCB contains a pointer to a 256-byte buffer from which BSCLIB can copy to the requested translation table.

Buffer Size:

The value passed in the Buffer Size field of the BCB specifies the size in bytes of the buffer pointed to by the Buffer Address field. This value must be 256.

4.4.1.4 Subopcode 10 - Clear Received Terminal ID

The Clear Received Terminal ID function of the INITIALIZE command is used to clear the memory used to save any terminal ID that may have been received.

4.4.2 Opcode 1 - INSTALL Command

This command prepares installs and initializes BSCLIB.

Prerequisite Calls:

None

Input BCB Fields:

Byte 0	Opcode
Byte 4,5	Flags
Byte 6-9	32-Bit Buffer Address (Unix)
Byte 10,11	Buffer Size (Unix)
Byte 16,17	Shared Memory ID (Unix)
Byte 22	Port (SyncPCI and SmartSync/DCP)
Byte 23	Adapter (SmartSync/DCP)

Returned BCB Fields:

Byte 1 Return Code

In single-port and multi-port applications using the SyncPCI adapter, this command loads **emubsc**, the BSC protocol handler, and initializes shared memory. In the Windows environments, BSCLIB expects to find **emubsc** in the current directory or in one of the paths defined in the PATH environment variable.

In the Unix environment, BSCLIB expects to find **emubsc** in the current directory unless an alternate path is passed via the Buffer Address field of the BCB.

In the multi-port version using the SmartSync/DCP adapter, this command starts the corresponding BSC process running within the adapter.

This function must be called before calling the OPEN command.

Flags:

Bit 14 – Skip Async Initialization

0 = No
1 = Yes

When configured for AT-command set modems and Skip Async Initialization = 1, BSCLIB does not initialize the communications adapter during initialization. This is necessary if your application has previously uninstalled BSCLIB while the line remained connected (DTR left high) and you wish to re-establish the

connection. Calling INSTALL otherwise causes the connection to be lost.

Buffer Address:

In Unix environments, the Buffer Address field of the BCB may contain a pointer to a buffer containing the fully qualified pathname to where the **emubsc** program file is located. If set to 0, the BSCLIB install directory, **/usr/lib/bsclib** (**/usr/lpp/bsclib** on AIX), will be used.

Buffer Size:

In Unix environments, the value passed in the Buffer Size field of the BCB specifies the size in bytes of the pathname pointed to by the Buffer Address. For example:

```
char *emubsc_path;

emubsc_path = "/usr/local/MyApp";
lbcb.bufr = emubsc_path;
lbcb.val = strlen(emubsc_path);
```

Shared Memory ID:

In Unix environments, the value passed in the Shared Memory field of the BCB specifies the shared memory ID to be used to connect with **emubsc**, the daemon BSC protocol emulation process. BSCLIB uses this identifier to locate and share information with the **emubsc** process. If set to 0, the default ID, hexadecimal value 311, is used.

4.4.3 Opcode 2 - OPEN Command

This command is used to establish a connection with another BSC station. A connection may be established via auto-dial or auto-answer modems on normal dial-up (POTS) telephone lines; direct connections via leased line modems or synchronous modem eliminators; or network connections via CSU/DSU or FRAD devices.

Prerequisite Calls:

INITIALIZE
INSTALL

Subopcodes:

0 – Dial Modem
1 – Open Communications

Input BCB Fields:

Byte 0	Opcode
Byte 2	Subopcode
Byte 4,5	Flags
Byte 6-9	32-Bit Buffer Address
Byte 10,11	Buffer Size
Byte 14, 15	Time-out
Byte 16, 17	Auxiliary Flags
Byte 22	Port (SyncPCI and SmartSync/DCP)
Byte 23	Adapter (SmartSync/DCP)

Returned BCB Fields:

Byte 1	Return Code

4.4.3.1 Subopcode 0 - Dial Modem

When used with supported auto-dial modems, this function is used to send a telephone number to the modem and to monitor for a successful or unsuccessful connection.

Your program places the telephone number into a buffer pointed to by the Buffer Address field and the length of the string into the Buffer Size field.

Flags:

Bit 0 – Non-blocking I/O:

0 = No
1 = Yes

If Non-blocking I/O = 0 then BSCLIB does not return control to your program until the Dial Modem function completes. Setting the flag = 1 instructs BSCLIB to return control immediately without waiting for the dial to complete. In this case, your application should issue STATUS calls periodically until the dial operation completes.

Buffer Address:

The Buffer Address field of the BCB contains a pointer to a buffer containing the telephone number and any command modifiers supported by the target modem (i.e., wait for tone, pause five seconds and continue, etc.)

Buffer Size:

The value passed in the Buffer Size field of the BCB specifies the size in bytes of the buffer pointed to by the Buffer Address.

Time-out:

The value passed in the Time-out field of the BCB specifies the time in units of 100 milliseconds (tenths of seconds) the dial process will "keep trying", in the absence of successful completion or specific failure, before returning a return code of 186 (Dialer Time-out). Other failures such as 185 or 189 (Modem command errors) or 181 (Busy signal detected) could also be returned if a connection can not be established. Always establish a reasonable non-zero interval for this parameter.

Auxiliary Flags:

Bit 9 – Pulse Dial (on Auto-Dial OPEN):

0 = No
1 = Yes

If Pulse Dial = 1 when performing a dial operation, BSCLIB adds the appropriate command to the dial command string to enable pulse (rather than tone) dialing on supported modems.

Bit 12 – Enable Manual Dial:

0 = No
1 = Yes

Enable Manual Dial = 1 facilitates a manual dial operation. The net result of the option is to assert the DTR modem signal. The connection to the remote system is established when the local modem turns on the DSR modem signal.

Return Codes:

For blocking calls, return codes may be:

0	Successful completion, connection established

Temporary failures, for which you may wish to wait and try again later:

181	Busy signal
183	No dial or answer back tone detected
186	Dialer time-out, no connection established

Any other return codes may be regarded as a permanent failure, for which you may need to change your program or your modem.

After non-blocking calls, you may receive any of the above, as well as the following additional return codes, from a STATUS call:

255 In progress, no specific event
- 180 In progress - ring detected*
- 182 In progress - dial tone detected*
- 184 In progress - answer back tone detected*
- 187 Dial command aborted

* Codes 180, 182, 184 are not returned by all modems, and may be considered equivalent to 255 if received.

4.4.3.2 Subopcode 1 - Open Communications

The Open Communications function of the OPEN command is used for all connections other than auto-dialing modems. It instructs BSCLIB to raise the control signal DTR and wait for the DSR signal to be returned from the modem indicating that a connection with a remote station has been established.

Flags:

Bit 0 – Non-blocking I/O:

0 = No

1 = Yes

If Non-blocking I/O = 0, then BSCLIB does not return control to your program until the connection is established or the Time-out expires. Setting the Non-blocking I/O flag = 1 instructs BSCLIB to return control immediately without waiting for DSR. Return Code 255 is returned indicating that the OPEN is in progress. See the STATUS command for an explanation of how to check for completion of the OPEN.

Auxiliary Flags:

Bit 9 – Monitor Ringing (on Auto-Answer OPEN):

0 = No

1 = Yes

If Monitor Ringing = 1 when performing an auto-answer operation on AT command set modems, BSCLIB looks for ring indicator to be returned from the modem prior to a connection. If a ring is detected, a STATUS call (called to check for OPEN completion) will return result code 180 once for each ring. This result code will be randomly interspersed with 255 result codes until the OPEN completes, so your application must allow for this.

Your application may make note of this intermediate result code to know that a connection is pending and to not issue an ABORT (or take other action) that may disrupt an incoming call.

Bit 10 – Trial Open:

0 = No

1 = Yes

If Trial Open = 1, BSCLIB only checks to see if the DSR modem signal is present on an OPEN command. A successful open is indicated to the application program, but BSCLIB takes no action.

Bit 12 – Enable Auto-Answer:

0 = No

1 = Yes

This flag is valid only with an external AT command set modem such as the Hayes Optima. When Enable Auto-Answer = 1, BSCLIB sends the appropriate commands to the modem to enable auto-answer mode. By default, the number of rings is set to 1. An abort of the OPEN command causes auto-answer mode to be reset. BSCLIB reports a connection when the DSR modem signal is detected from the local modem.

Time-out:

The value passed in the Time-out field of the BCB specifies the time in units of 100 milliseconds (tenths of seconds) the process will "keep trying", in the absence of successful completion or specific failure, before returning a return code of 11 (Command Time-out). A Time-out value of 0 instructs BSCLIB to wait indefinitely for a connection. A non-blocking OPEN call may be aborted by issuing the ABORT command.

Return Codes:

For blocking calls, return codes may be:

0	Successful completion, connection established
11	Command timed out, connection not established
185	Modem error occurred

Any other return code signals failure to achieve connection.

After non-blocking calls, you may receive either of the above, as well as the following return code from a STATUS call:

180	Ring detected
255	Open in progress

4.4.4 Opcode 3 - READ Command

In point-to-point operation, this command is used to instruct BSCLIB that the application is ready to accept a line bid in order to start receiving data. In multi-point control station operation, this command instructs BSCLIB to transmit one or more poll sequences to Get data from a remote tributary station. In multi-point tributary operation, this command instructs BSCLIB to positively respond to the next select that matches the defined address such that data can be received from the remote control station. One of the READ command's subopcodes (i.e. Send RVI) may also be used to tell the remote station to stop sending data.

Prerequisite Calls:

INSTALL
OPEN

Subopcodes:

0 – Standard Read
1 – Abort Read
2 – Initiate RVI
3 – Uni-Directional Standard Read
4 – Read To ETX Terminated Block (in preparation to send a conversational reply)
10 – Get Received Terminal ID
11 – Get Selected Station Address
20 – Store Multi-Point Address
21 – Store Multiple Select Address List
22 – Add Select Address To List
23 – Remove Select Address From List

Input BCB Fields:

Byte 0	Opcode
Byte 2	Subopcode
Byte 4,5	Flags
Byte 6-9	32-Bit Buffer Address
Byte 10,11	Buffer Size
Byte 14,15	Time-out
Byte 16,17	Auxiliary Flags
Byte 22	Port (SyncPCI and SmartSync/DCP)
Byte 23	Adapter (SmartSync/DCP)

Returned BCB Fields:

Byte 1	Return Code
Byte 4,5	Flags
Byte 12,13	I/O Size

4.4.4.1 Subopcode 0 - Standard Read

In point-to-point operation, the Standard Read function of the READ command instructs BSCLIB to accept a line bid and data blocks from the remote station. Similarly in multi-point tributary mode, this function instructs BSCLIB to respond positively to a select that matches a specified address. The select address is defined via the Store Multi-Point Address function (subopcode 20) or the Store Multiple Select Address List function (subopcode 21) before the first Standard Read is issued.

Line Bids and selects from the remote station are NAKed until the READ command is called. If a read has begun and BSCLIB's data blocks are full, WACKs are sent to the remote station until a data block is emptied by issuing READ commands.

In multi-point control station mode, the Standard Read function results in poll sequences being transmitted to the remote tributary station. The poll address is specified via the Store Multi-Point Address function prior to the first Standard Read is issued. The number of polls transmitted is controlled by the Bid Retry Limit in the Parameter Initialization Block (PIB).

Records are extracted sequentially from each data block and returned in your program's BCB Buffer. Until BSCLIB receives an EOT, DLE-EOT, or a disconnect occurs, READ commands must be issued continuously to retrieve each record from the received data blocks. DO NOT use the STATUS command to poll for inbound data. Rather, issue READ commands at regular intervals with the non-blocking bit set. Examine the Returned I/O Size field to determine if data has been received: greater than 0 indicates data is in the buffer; 0 indicates nothing has been received since the last READ command.

Flags:

Bit 0 – Non-blocking I/O:

0 = No
1 = Yes

If Non-blocking I/O = 0, then BSCLIB does not return control to your program until a record from a received data block is copied into your program's BCB Buffer or the Time-out expires. Setting

the Non-blocking I/O flag = 1 instructs BSCLIB to return control immediately without waiting for data to be received. Return Code 255 is returned indicating that the READ is in progress.

Bit 2 – Binary File Mode:

0 = No
1 = Yes

If Binary File Mode = 1, BSCLIB does not perform an EBCDIC to ASCII translation of received transparent data blocks. Because this is a non-standard protocol, the receiving station must be expecting to receive binary data. If non-transparent data blocks are received this flag has no effect.

Bit 5 – Expand Spaces:

0 = No
1 = Yes

If Expand Spaces = 1, BSCLIB will expand received space compression sequences when data records are copied to your program's buffer. Expansion will not be performed if the Expand Spaces flag = 0.

Bit 6 – Suppress VFC at ETB/ETX:

0 = No
1 = Yes

VFC is an abbreviation for Vertical Forms Control. When BSCLIB finds a record terminated by an ETB or ETX and Suppress VFC = 1, the record is returned to your application program without having carriage control appended to the end. When Suppress VFC = 0, a CR/LF or LF will be appended to the record unless another recognized VFC sequence preceded the record. This option is typically used to avoid a single CR/LF or LF being returned to your application program after a record ending with a record separator (IRS/IUS) is found to be followed immediately by an ETB or ETX. The block in this case looks like this:

...<Rec 1><IRS>...<Rec n><IRS><ETX>...

Bit 7 – Decode VFC:

0 = No
1 = Yes

If Decode VFC = 1, BSCLIB will convert received Vertical Forms Control characters to ASCII carriage control characters. Conversion will not be performed if the Decode VFC = 0. If VFC

is not decoded, BSCLIB defaults to a single carriage return / line feed (single space.) The following VFC sequences are recognized:

Esc T – triple space
Esc S – double space
Esc / – single space
Esc M – suppress space
Esc A – top of form

Bit 8 – Strip VFC:

0 = No
1 = Yes

If Decode VFC = 1, Strip VFC is ignored. If Decode VFC = 0 and Strip VFC = 1, BSCLIB will discard received Vertical Forms Control characters from the data blocks. If both Decode VFC and Strip VFC = 0, the VFC sequences are passed intact to your application program.

Bit 12 – Suppress Block Response:

0 = No
1 = Yes

If Suppress Block Response = 1 and a Uni-Directional Read is in progress, BSCLIB does not transmit a block response (i.e., ACK/0 or NAK). This results in a continuous stream of unacknowledged data blocks being accepted by BSCLIB. This results in a purely one-way data transfer.

Bit 13 – Accept NAK'd Data Blocks:

0 = No
1 = Yes

If Accept NAK'd Data Blocks = 1 and a Uni-Directional Read is in progress, BSCLIB does not discard data blocks received with a CRC or LRC error. The contents of the block is passed to your application program as if the block was received error free – any error detection is the responsibility of your application.

Bit 14 – Block Acknowledgement Control:

0 = Disabled
1 = Enabled

When Block Acknowledgement Control = 1 on the first READ command, BSCLIB enables a mode in which your application program has control over whether an ACK is sent in response to an otherwise valid data block. This flag must be set on the first

READ or it will be ignored. This feature is supported only when configured for 3780 emulation.

This feature permits your application to examine data received from the remote system and make a decision if the data is to be accepted or not. A typical application would be the rejection of an invalid sign-on record.

A block is positively acknowledged when your application issues the next READ. If the block is to be rejected, your application should issue an ABORT. If your application does not issue a command quickly enough (within approximately two seconds), BSCLIB sends WACK responses until a READ or ABORT is issued.

Auxiliary Flags:

Bit 0 – Disable Printer:

0 = No
1 = Yes

If Disable Printer = 1 and Master Device Control = 1, BSCLIB disables the printer receive device. All incoming data bound for the printer is refused with an immediate response of an EOT. When another READ command is issued with Disable Printer = 0 and Master Device Control = 1, the printer receive device is re-enabled.

Bit 1 – Disable Punch:

0 = No
1 = Yes

If Disable Punch = 1 and Master Device Control = 1, BSCLIB disables the punch receive device. All incoming data bound for the punch is refused with an immediate response of an EOT. When another READ command is issued with Disable Punch = 0 and Master Device Control = 1, the punch receive device is re-enabled.

Bit 2 – Master Device Control:

When Master Device Control = 1, BSCLIB examines the Disable Printer and Disable Punch flags and performs appropriately. These flags are ignored when Master Device Control = 0.

Bit 3 – Disable Translation:

0 = No
1 = Yes

If Disable Translation = 1, BSCLIB bypasses EBCDIC to ASCII translation on inbound data, whether it be transparent or non-

transparent data. This option may be used in conjunction with the Suppress VFC at ETX/ETB and Suppress IRS/IUS options to permit your application program to receive the raw non-transparent data sent by the remote system. All interpretation of the data is left to your program.

Bit 4 – Recognize New Lines:

0 = No
1 = Yes

If Recognize New Lines = 1, BSCLIB recognizes EBCDIC New Line (NL) characters as record delimiters in received data blocks. BSCLIB translates each NL character to CR/LF. All NL characters are included in records as data if the flag is set to 0.

Bit 6 – Suppress IRS/IUS:

0 = No
1 = Yes

If Suppress IRS/IUS = 1, BSCLIB ignores the IRS (in 3780 mode) or the IUS (in 2780 mode) control characters that are normally used to mark the end of a logical record in the data stream. The IRS/IUS is returned to your program if Disable Translation = 1and BSCLIB uses the Printer or Punch Record Size value to determine how many characters to return. No VFC or CR/LF sequences are returned to your program so determination of logical record formatting is left to your application program.

Bit 10 – Enable Automatic 3780/2780 Protocol Detection:

0 = No
1 = Yes

If Enable Automatic 3780/2780 Protocol Detection = 1, BSCLIB will inspect the incoming data stream for telltale indicators of either a 3780 or 2780 data stream and automatically configure itself accordingly for the duration of the connection. The following indicators are used:

3780 Data Stream
IRS record separator characters
IGS space compression characters

2780 Data Stream
EM record separator character
ITB intermediate block check characters

The success or failure of this detection feature depends on the contents of a specific data stream. It is possible for BSCLIB to not be able to make a protocol determination. This is especially true of uncompressed text, transparent, or binary data streams. **It is incumbent on the BSCLIB application developer to be aware of the nature of the possible data streams before depending on this feature of BSCLIB for accurate protocol detection**.

For this option to work properly, your application must configure BSCLIB initially for 3780 emulation.

Bit 14 – Disable Abort Receive Delay:

0 = No
1 = Yes

If Disable Abort Receive Delay = 1, BSCLIB will immediately respond to a request to abort a receive operation. Enabling this option may be necessary in certain applications when there is a high volume of bidirectional message traffic, especially if the messages are short. Otherwise, do not change this option.

Bit 15 – Enable Abort Prevention:

0 = No
1 = Yes

If Enable Abort Prevention = 1, BSCLIB ignores an ABORT request on a READ if it detects that a line bid has been received. If this occurs, return code 131 is returned on an ABORT command and your application should resume issuing READs to get the received data.

Buffer Address:

The Buffer Address field of the BCB contains a pointer to a buffer in which BSCLIB will return a received data record.

Buffer Size:

The value passed in the Buffer Size field of the BCB specifies the size in bytes of the buffer pointed to by the Buffer Address field. If a data record is received which is larger than the specified Buffer Size, it will be truncated to fit (see returned Flags). The buffer must be large enough to fit the largest record you will receive, plus any extra VFC characters (i.e. CR/LF).

Time-out:

The value passed in the Time-out field of the BCB specifies the number of 100 milliseconds intervals to wait for data to be

available before returning a time-out Return Code. A Time-out value of 0 instructs BSCLIB to wait indefinitely for data.

Returned Flags:

Bit 1 – Transparent Mode Detected:

0 = No
1 = Yes

If BSCLIB returns with Transparent Mode Detected = 1, an incoming transparent transmission has been detected. If your program set Binary Data Mode = 1 when making the READ command, your program may safely assume that binary data is being received. If the incoming data is not transparent, the value of this flag will be 0.

Bit 4 – PUNCH Select:

0 = No
1 = Yes

If PUNCH Select = 1, BSCLIB has received a Punch Select sequence from the remote station. If this bit is not set, you can assume the printer has been selected.

Bit 9 – Buffer Overflow:

0 = No
1 = Yes

If Buffer Overflow = 1, BSCLIB has found a record which is larger than the size specified in the Buffer Size field of the BCB.

Bit 10 – Record Truncated:

0 = No
1 = Yes

If Record Truncated = 1, BSCLIB has found a record which is larger than the configured Printer or Punch Record Size. Excess characters are returned on the next READ.

Bit 15 – Last Record:

0 = No
1 = Yes

If Last Record = 1, BSCLIB is returning the last record in an ETX terminated block. Such a record is always the last record in an inbound block. This flag may be used to detect the end of a logical grouping within a transmission.

Returned Auxiliary Flags:

Bit 7 – SOH Received:

0 = No
1 = Yes

BSCLIB returns with SOH Received = 1 on a READ that returns a record found immediately following a SOH character. Such a record is always the first record in an inbound block.

Bit 11 – Receive Protocol Detected:

0 = 2780
1 = 3780

This flag is set to indicate whether a 3780 or 2780 data stream is presently being processed by BSCLIB. This flag has no real value unless Enable Automatic 3780/2780 Protocol Detection is enabled on the READ because it simply reflects the originally configured protocol. If the detection feature is enabled, however, this flag will reflect which protocol BSCLIB actually detected.

Bit 13 – Last Record in Data Block:

0 = No
1 = Yes

If Last Record in Data Block = 1, BSCLIB is returning the last record in an ETB terminated block. Such a record is always the last record in an inbound block. This bit will also be set it bit 15 in Flags is set.

Returned I/O Size:

The value returned in the I/O Size field of the BCB indicates the number of bytes BSCLIB copied into your program's buffer. This field should always be examined for a non-zero value as an indicator that data has been copied into the buffer even if a Return Code other than I/O in progress was returned.

4.4.4.2 Subopcode 1 - Abort Read

The Abort Read function of the READ command instructs BSCLIB to stop accepting data or a line bid from the remote station. If a Line Bid has already been accepted, an End of Transmission (EOT) is sent to the remote station as an instruction to stop sending data and an indication that BSCLIB has stopped processing data already received. The STATUS command must be used to check for the completion of the Abort Read function.

4.4.4.3 Subopcode 2 - Initiate RVI (Reverse Interrupt)

The Initiate RVI function of the READ command instructs BSCLIB to request control of the communication line by sending an RVI to the remote station after the next data block is received. The RVI is an acknowledgment that the data block has been properly received. It requests that the remote station stop sending data and to send an EOT in order to end the data transfer. After your application sends an RVI, the remote system expects your application to begin transmitting. If a Line Bid has not been received yet, the Initiate RVI function is treated as an Abort.

Standard READ functions should be continued until all data records have been processed and an EOT received Return Code is returned. If the remote station does not support RVIs, then the ABORT command should be used instead.

4.4.4.4 Subopcode 3 - Uni-Directional Standard Read

The Uni-Directional Standard Read function of the READ command enables what is referred to as "uni-directional" or "simplex" receives on the part of BSCLIB. This function may be used only in point-to-point connections. A uni-directional receive operation is for the most part identical to a Standard Read function, but differs from a normal BSCLIB receive in the following ways:

- no line bid is allowed; BSCLIB expects the inbound data to begin immediately with an SOH, STX or DLE-STX followed by a normal BSC data block

- BSCLIB always responds to the received data block with ACK/1; NAK or WACK; if Suppress Block Response = 1, BSCLIB suppresses all block responses completely

This type of read is useful in one-way data feeds where less than 100% data integrity is permissible. Standard and Uni-Directional Standard Read's should not be mixed during the same session.

4.4.4.5 Subopcode 4 - Read To ETX Terminated Block

The Read To ETX Terminated Block function of the READ command is the first step to enabling your application to send a limited conversational reply. This function otherwise identical to a Standard READ, but will return result code 132 when an ETX terminated block is received. At this point the READ operation is complete, and your application must issue WRITE's in order to transmit the data making up the conversational reply.

4.4.4.6 Subopcode 10 - Get Received Terminal ID

This function returns the most recently received terminal ID string if any. Terminal ID's are optional and are used only in point-to-point connections.

Buffer Address:

The Buffer Address field of the BCB contains a pointer to a buffer in which BSCLIB will return the terminal ID string. This string can be up to 20 bytes in length. YOUR PROGRAM MUST PROVIDE A POINTER TO A SUFFICIENTLY LARGE BUFFER.

Returned I/O Size:

The size of the string is returned in the Buffer Address field. If no terminal ID has been received, zero is returned.

4.4.4.7 Subopcode 11 - Get Selected Station Address

This function returns the received select address that matched an entry in the select address list created with the Store Multiple Select Address List function (subopcode 21). When BSCLIB is configured to recognize multiple select address as a tributary station in multi-point mode, this function is how your program knows which station the remote system has selected for the data arriving during this READ command. Your program should call this function immediately after the first Standard READ returns indicating that data has been received.

Buffer Address:

The Buffer Address field of the BCB contains a pointer to a buffer in which BSCLIB will return the select address string. Allow for a string up to 20 bytes in length. YOUR PROGRAM MUST PROVIDE A POINTER TO A SUFFICIENTLY LARGE BUFFER.

Returned I/O Size:

The size of the string is returned in the Buffer Address field.

4.4.4.8 Subopcode 20 - Store Multi-Point Address

The Store Multi-Point Address function of the READ command specifies the poll or select address associated with the READ command.

For an application configured as a Control Station, this function specifies the polling address of the tributary station to be read from. If the Control Station is polling to multiple tributary stations, this address must be changed prior to initiating a new read intended for a different station.

For an application configured as a Tributary Station, this function specifies the station's select address. Alternatively, you may specify one or more select addresses using READ Subopcode 21 Store Multiple Select Address List function described below, but both methods cannot be used. Calling this function cancels the use of a select address list if one has been defined.

Buffer Address:

The Buffer Address field of the BCB contains a pointer to a null terminated string which defines the multi-point address. This string should be six characters or less.

4.4.4.9 Subopcode 21 - Store Multiple Select Address List

The Store Multiple Select Address List function of the READ command allows your program to specify one or more select addresses to be associated with the READ command. If your application is emulating a control unit with multiple devices, this function allows you to define all individual devices or stations for which data is to be accepted.

This function only applies to tributary stations and is mutually exclusive with the Store Multi-Point Address function. This means that when this function is called, the addresses in the list are used to recognize a select from the remote overriding the select address that may have been set with READ Subopcode 20 Store Multi-Point Address.

When multiple select addresses are defined, BSCLIB compares each address received with the contents of this list. If there is a match, BSCLIB accepts the select. If necessary, your application may use the "Get Selected Station Address" function (subopcode 11) to get the address that was matched – do this immediately upon detecting inbound data.

Buffer Address:

The Buffer Address field of the BCB contains a pointer to a list of up to 64 null terminated strings that define the multi-point addresses. Each address string should be six characters or less. The end of the list is marked by one more nulls.

4.4.4.10 Subopcode 22 - Add Select Address To List

The Add Select Address To List function of the READ command adds one address to the select address list. The list may have been previously created by the Store Multiple Select Address List function or added to one at a time using this function. BSCLIB maintains up to 64 20-character select addresses in the select address list. Addresses can be added to the list at any time when a READ is not in progress.

Buffer Address:

The Buffer Address field of the BCB contains a pointer to a null terminated string that defines the address to be added to the list. This string should be six characters or less.

4.4.4.11 Subopcode 23 - Remove Select Address From List

The Remove Select Address From List function of the READ command removes one address from the select address list created by the Store Multiple Select Addresses function or added to by the Add Select Address To List function (subopcode 22). BSCLIB maintains up to 64 20-character select addresses in the select address list. Addresses can be removed from the select address list at any time when a READ is not in progress.

Buffer Address:

The Buffer Address field of the BCB contains a pointer to a null terminated string which specifies the address to be removed from the list. This string should be six characters or less.

4.4.5 Opcode 4 - WRITE Command

In point-to-point operation, this command is used to instruct BSCLIB to request control of the line and to transmit data to the remote station. In multi-point control station operation, this command instructs BSCLIB to transmit one or more select sequences to initiate a transmission to a remote tributary station. In multi-point tributary operation, this command instructs BSCLIB to positively respond to the next poll that matches the defined address such that data can be transmitted to the remote control station.

Prerequisite Calls:

INSTALL
OPEN

Subopcodes:

0 – Standard
1 – Send ETB Block
2 – Send ETX Block
3 – Send ETX Block and EOT
4 – Send Forward Abort
10 – Send Line Tickle
11 – Send EOT
20 – Store Multi-Point Address

Input BCB Fields:

Byte 0	Opcode
Byte 2	Subopcode
Byte 4,5	Flags
Byte 6-9	32-Bit Buffer Address
Byte 10,11	Buffer Size
Byte 14,15	Time-out
Byte 16,17	Auxiliary Flags
Byte 22	Port (SyncPCI and SmartSync/DCP)
Byte 23	Adapter (SmartSync/DCP)

Returned BCB Fields:

Byte 1	Return Code
Byte 12,13	I/O Size

4.4.5.1 Subopcode 0 - Standard Write

The Standard Write function of the WRITE command instructs BSCLIB to copy the data record from your program's BCB buffer into the BSCLIB's active transmit buffer.

In point-to-point mode, when the first buffer is full, BSCLIB will transmit a line bid and wait for an acknowledgment (ACK) from the remote station – at this time the first data block is sent. Subsequent blocks are sent as soon as a transmit buffer is filled.

In multi-point control station mode, BSCLIB sends a select sequence when the first buffer is full. The select address is defined via the Store Multi-Point Address function (subopcode 20) before the first Standard Write is issued.

In multi-point tributary mode, the Standard Write function instructs BSCLIB to recognize a poll sequence once the first buffer is full in preparation to transmit of data to the control station. The poll address is defined by issuing the Store Multi-Point Address function before the first Standard Write is issued.

Flags:

Bit 0 – Non-blocking I/O:

0 = No

1 = Yes

If Non-blocking I/O = 0, then BSCLIB does not return control to your program until the record is copied to the transmit buffer and,

if the buffer becomes full, the buffer is transmitted, or the Time-out expires. Setting this flag = 1 instructs BSCLIB to return control immediately without waiting for the data block to be transmitted.

When Non-blocking I/O = 1, the STATUS command must be called to check for the completion of the transmission or an error condition after the last record is written.

Bit 1 – Transparent Mode:

0 = No
1 = Yes

If Transparent Mode = 1, BSCLIB sends all data blocks in the session using EBCDIC transparency. Any data character in the outbound stream that would be interpreted as a BSC control character (e.g., ETX) has a DLE character inserted before it. All transparent data blocks begin with a DLE-STX and end with a DLE-ETB or DLE-ETX indicating to the remote station that the blocks contain transparent data.

Bit 2 – Binary File Mode:

0 = No
1 = Yes

If Binary File Mode = 0, BSCLIB translates ASCII characters to EBCDIC in accordance with its built-in translation table before moving them to the transmit buffer. The translation table may be modified using the INITIALIZE command's Store Translation Table function (subopcode 2).

If Binary File Mode = 1, BSCLIB does no ASCII-to-EBCDIC translation of outbound data when it is copied to BSCLIB's transmit buffer. Data blocks are then sent in transparent mode. Because this is a non-standard protocol, the remote station must be expecting to receive binary data.

Bit 4 – Punch Select:

0 = No
1 = Yes

If Punch Select = 1, BSCLIB sends a Punch select sequence (DC3) in front of the first data block transmitted to the remote station.

Bit 5 – Compress Spaces:

0 = No
1 = Yes

If Compress Spaces = 1, BSCLIB will compress strings of spaces into space compression sequences when data records are copied to BSCLIB's transmit buffer. Compression will not be performed if Compress Spaces = 0.

Space compression should not be enabled unless it is known that the remote system has space expansion enabled.

Bit 7 – Leave Final IRS:

0 = No
1 = Yes

If Leave Final IRS = 1, BSCLIB does not remove the Inter-Record Separator (IRS) from the last record in an outbound data block. The block is formatted as shown:

...<Rec 1><IRS>...<Rec n><IRS><ETX>...

By default BSCLIB removes the final IRS in a block (any IRS that immediately precedes the ETB or ETX.) The block is:

...<Rec 1><IRS>...<Rec n><ETX>...

Bit 13 – Enable Record Truncation in 3780 Mode:

0 = No
1 = Yes

If Enable Record Truncation in 3780 Mode = 1, BSCLIB strips any trailing spaces in records before placing them in a communication buffer for transmission. This option has no effect is 2780 mode is selected.

Bit 15 – Retransmit Block After NAK on ENQ:

0 = No
1 = Yes

This flag permits your application to control how BSCLIB attempts to recover from an unusual bisync transmission error. By default, BSCLIB disconnects the line if the following condition occurs:

```
<Data Block> →
             ← (no response)
       ENQ →
             ← NAK
(Disconnect) →
```

If Retransmit Block After NAK on ENQ = 1, BSCLIB retransmits <Data Block> rather than disconnecting should this condition occur. It is the opinion of Serengeti Systems that the safe thing to do (to preclude the transmission of duplicate data) is to disconnect, but in some situations retransmission of the block is the preferred action.

Auxiliary Flags:

Bit 3 – Disable Translation:

0 = No
1 = Yes

If Disable Translation = 1, BSCLIB bypasses ASCII to EBCDIC translation on outbound data. Your program is responsible for insuring that the outbound data is in a valid format and contains no characters that could be misinterpreted as BSC control characters. The presence of control characters in a non-transparent transmission could inadvertently be recognized as BSC control characters by the remote system and result in transmission failure.

Bit 5 – Suppress IRS:

0 = No
1 = Yes

If Suppress IRS = 1, BSCLIB does not insert a record separator after each logical record within the outbound data stream. This flag applies to 3780 mode only.

Bit 7 – Start Block With SOH:

0 = No
1 = Yes

If Start Block With SOH = 1, BSCLIB begins each outbound block with an SOH character (instead of an STX). A second WRITE into the same block results in an STX followed by the written data to be inserted into the block. All subsequent WRITEs into the block are unaffected. For example, after three WRITE commands the block is as follows:

<SOH><rec 1><STX><rec2><rec3> ...

This flag cannot be used on transparent or binary WRITE commands. As long as this flag is set, all outbound blocks are handled in this way. It is up to your application to manage the placement of records that must be between each SOH and STX.

SOH headers are not commonly used and necessary only if required by the remote system.

Buffer Address:

The Buffer Address field of the BCB contains a pointer to a buffer from which BSCLIB will copy the data record to be transmitted.

Buffer Size:

The value passed in the Buffer Size field of the BCB specifies the size in bytes of the buffer pointed to by the Buffer Address field.

Time-out:

The value passed in the Time-out field of the BCB specifies the number of tenths of seconds (100 milliseconds) to wait for data to be transmitted before returning a time-out Return Code. A Time-out value of 0 instructs BSCLIB to try indefinitely to transmit (See also BID Retry, ENQ Retry, and NAK limits.) If WRITE is called with the Non-blocking I/O Flag = 1 and a Time-out = 0, the Standard Write function may be aborted by issuing the ABORT command.

Returned I/O Size:

The value returned in the I/O Size field of the BCB indicates the number of bytes BSCLIB copied into its transmit buffer.

4.4.5.2 Subopcode 1 - Send ETB Block

The Send ETB Block function of the WRITE command instructs BSCLIB to copy the data record from your program's BCB buffer into the BSCLIB's transmit buffer and send the buffer as a data block ending with an ETB (end of transmission block.) An ETB is the normal block terminating character of all blocks other than the last block sent in a file transmission.

Flags:

Bit 14 – Ignore Record:

0 = No
1 = Yes

If Ignore Record (formally “No Empty Record”) = 1, BSCLIB ignores anything that may be in the Buffer Address and Buffer Size fields and force the transmission of data.

In effect this option tells BSCLIB to ignore any record present in this WRITE call’s BCB and to initiate transmission of any previously written records. Any pending transmit buffer is terminated with an ETB or ETX (as appropriate) and the transmitted.

If Ignore Record = 0, even if the Buffer Size field was 0, BSCLIB would add an empty record to the end of the buffer before the ETB or ETX.

4.4.5.3 Subopcode 2 - Send ETX Block

The Send ETX Block function of the WRITE command instructs BSCLIB to copy the data record from your program's BCB buffer into the BSCLIB's transmit buffer and send the buffer as a data block ending with an ETX (end of text.) An ETX is the normal block terminating character of last block sent in a transmission. ETX blocks are often used to indicate the end of each file when sending multiple files.

Flags:

Bit 14 – Ignore Record:

0 = No
1 = Yes

If Ignore Record (formally "No Empty Record") = 1, BSCLIB ignores anything that may be in the Buffer Address and Buffer Size fields and force the transmission of data.

In effect this option tells BSCLIB to ignore any record present in this WRITE call's BCB and to initiate transmission of any previously written records. Any pending transmit buffer is terminated with an ETB or ETX (as appropriate) and the transmitted.

If Ignore Record = 0, even if the Buffer Size field was 0, BSCLIB would add an empty record to the end of the buffer before the ETB or ETX.

Auxiliary Flags:

Bit 11 – Conversational Reply Allowed:

0 = No
1 = Yes

If Conversational Reply Allowed = 1, BSCLIB will accept a conversational reply to a transmitted data block that ends with an ETX. If a conversational reply is detected, BSCLIB returns result code 111 to your application. At this time, the WRITE has completed and your application should issue READ's immediately to read the data contained in the conversational reply and any subsequent data that may be received.

4.4.5.4 Subopcode 3 - Send ETX Block and EOT

The Send ETX Block and EOT function of the WRITE command instructs BSCLIB to copy the data record from your program's BCB buffer into the BSCLIB's transmit buffer and send the buffer as a data block ending with an ETX. After the block has been sent and acknowledged by the remote station, and EOT (end-of-transmission) is sent, releasing control of the line.

Flags:

Bit 14 – Ignore Record:

0 = No
1 = Yes

If Ignore Record (formally "No Empty Record") = 1, BSCLIB ignores anything that may be in the Buffer Address and Buffer Size fields and force the transmission of data.

In effect this option tells BSCLIB to ignore any record present in this WRITE call's BCB and to initiate transmission of any previously written records. Any pending transmit buffer is terminated with an ETB or ETX (as appropriate) and the transmitted.

If Ignore Record = 0, even if the Buffer Size field was 0, BSCLIB would add an empty record to the end of the buffer before the ETB or ETX.

Auxiliary Flags:

Bit 11 – Conversational Reply Allowed:

0 = No
1 = Yes

If Conversational Reply Allowed = 1, BSCLIB will accept a conversational reply to a transmitted data block that ends with an ETX. If a conversational reply is detected, BSCLIB returns result code 111 to your application. At this time, the WRITE has completed and your application should issue READs immediately to read the data contained in the conversational reply and any subsequent data that may be received.

Buffer Size:

Same as Standard otherwise a Buffer Size of 0 instructs BSCLIB to terminate the current block after the previously buffered data record.

Returned Auxiliary Flags:

Bit 8 – Residual Transmit Record Pending:

0 = No
1 = Yes

If Residual Transmit Record Pending = 1, on return to your program all or part of the previous record written to BSCLIB did not fit into the communication buffer. When Residual Transmit Record Pending = 1 after the program has written the last record (using the Write ETX Block and EOT function) your program should consider the WRITE command to be incomplete. One additional WRITE command must be made to force this partial record to be transmitted. This final call should be a zero length write with Bit 14 of BCB flags set.

4.4.5.5 Subopcode 4 - Send Forward Abort

The Send Forward Abort function of the WRITE command instructs BSCLIB send a Forward Abort sequence to the remote station, releasing control of the line and aborting the WRITE command. This informs the remote station that the sender must abnormally end the transmission and perhaps to disregard data blocks sent during the current session.

Flags:

Bit 0 – Non-blocking I/O:

0 = No
1 = Yes

If Non-blocking I/O = 0 then BSCLIB does not return control to your program until the Forward Abort has been sent or the Time-out expires. Setting Non-blocking I/O = 1 instructs BSCLIB to return control immediately without waiting for the transmission to complete. The STATUS command must be used to check for the completion of the transmission or an error condition.

4.4.5.6 Subopcode 10 - Send Line Tickle

The Send Line Tickle function of the WRITE command instructs BSCLIB send a Line Tickle sequence to the remote station. When a Line Tickle is requested, a Line Bid is sent followed by an EOT after the Line Bid has been ACKed by the remote station. This may be used by an application to keep the remote station's No Activity Time-out from expiring.

A WRITE command with this function may not be used if a WRITE is already in progress.

4.4.5.7 Subopcode 11 - Send EOT

The Send EOT (End of Transmission) function of the WRITE command instructs BSCLIB send an EOT to the remote station. This may be used by an application to insure that the remote station knows that control of the line has been relinquished.

A WRITE command to send an EOT may not be used if a WRITE is already in progress (use the Write ETX Block and EOT function).

4.4.5.8 Subopcode 20 - Store Multi-Point Address

The Store Multi-Point Address function of the WRITE command specifies the poll or select address associated with the WRITE command. For an application configured as a Control Station, the WRITE multi-point address specifies the select address of the tributary station to be sent to. If the Control Station is sending to multiple tributary stations, this address must be changed prior to initiating a write whenever the tributary station changes. For an application configured as a Tributary Station, the WRITE multi-point address specifies the station's polling address.

Buffer Address:

The Buffer Address field of the BCB contains a pointer to null terminated string which defines the multi-point address. This string should be six characters or less.

4.4.6 Opcode 5 - ABORT Command

This command instructs BSCLIB to abort a pending OPEN command or to abort a READ or WRITE command in progress. When a READ is in progress this command is identical to the Abort Read function of the READ command; when a WRITE is in progress this command is identical to the Send Forward Abort function of the WRITE command.

Prerequisite Calls:

INSTALL
OPEN
READ
WRITE

Subopcodes:

None

Input BCB Fields:

Byte 0	Opcode
Byte 4,5	Flags
Byte 22	Port (SyncPCI and SmartSync/DCP)
Byte 23	Adapter (SmartSync/DCP)

Returned BCB Fields:

Byte 1	Return Code

Flags:

Bit 0 – Non-blocking I/O:

0 = No

1 = Yes

If Non-blocking I/O = 0 then BSCLIB does not return control to your program until the abort completes. Setting Non-blocking I/O = 1 instructs BSCLIB to return control immediately without waiting for the abort to complete. The STATUS command must be used to check for completion or an error condition.

4.4.7 Opcode 6 - STATUS Command

This command instructs BSCLIB return the status of a pending OPEN, ABORT, or WRITE command in progress. If the command has not completed and has not encountered an error condition, a 255 Return Code (I/O in progress) is returned indicating the command is still in progress.

Your application program can call this command regularly to check for the completion of a non-blocking I/O command such as an auto-dial OPEN, ABORT or WRITE. When a READ command has been initiated, your application should issue additional READ commands until completion rather than use this command.

Prerequisite Calls:

INSTALL

Subopcodes:

None

Input BCB Fields:

Byte 0	Opcode
Byte 22	Port (SyncPCI and SmartSync/DCP)
Byte 23	Adapter (SmartSync/DCP)

Returned BCB Fields:

Byte 1	Return Code
Byte 2	Subopcode
Byte 3	BSCLIB State

Subopcode:

The opcode of current/last I/O operation in progress is returned in this field.

BSCLIB State:

The value returned in this field indicates the current state of the BSCLIB command in progress. See Appendix B for a list of State Codes and definitions.

4.4.8 Opcode 7 - STATISTICS Command

Use of the STATISTICS command and all its functions is optional.

Prerequisite Calls:

INSTALL

Subopcodes:

0 – Initialize Statistics
1 – Read Statistics
10 – Set 'TTDs Sent' value in SPB to count TTDs received
11 – Reset 'TTDs Sent' value in SPB to count TTDs transmitted

Input BCB Fields:

Byte 0	Opcode
Byte 2	Subopcode
Byte 6-9	32-Bit Buffer Address
Byte 22	Port (SyncPCI and SmartSync/DCP)
Byte 23	Adapter (SmartSync/DCP)

Returned BCB Fields:

Byte 1 Return Code

4.4.8.1 Subopcode 0 - Initialize Statistics

The Initialize Statistics function of the STATISTICS command is used to initialize BSCLIB's Statistics Parameter Block (SPB) structure, which is used to collect statistics of the communications session. When this function

is called all field values in the supplied SPB structure are copied into BSCLIB's internal structure and thus initialized.

Buffer Address:

The Buffer Address field of the BCB contains a pointer to a buffer containing an SPB Structure loaded with the desired initialization values (typically all 0's.)

4.4.8.2 Subopcode 1 - Read Statistics

The Read Statistics function of the STATISTICS command is used to copy BSCLIB's structure used to collect statistics of the communications session into your program's copy of the SPB.

Buffer Address:

The Buffer Address field of the BCB contains a pointer to a buffer into which BSCLIB will copy a SPB structure.

Statistics Information Block (SPB) structure is shown in Figures 11, 12, and 14.3. All fields, except for the BSCLIB Link State, are unsigned 16-bit integers and tally the number of occurrences of the indicated event. BSCLIB Link State values are listed in Appendix B. See Appendix C for a detailed description of SPB fields.

4.4.8.3 Subopcode 10 - Set 'TTDs Sent' to Count TTDs Received

This function of the STATISTICS command is used to redefine use of a field in the SPB. The 'TTDs Sent' field, by default, counts the number of TTDs sent by BSCLIB. Calling this function refines this field to count TTDs received instead.

4.4.8.4 Subopcode 11 - Reset 'TTDs Sent' to Count TTDs Sent

This function of the STATISTICS command is used to restore the default behavior of the 'TTDs Sent' field of the SPB.

4.4.9 Opcode 8 - TRACE Command

This command is used to control the communications line trace capabilities of BSCLIB. BSCLIB tracing may be used to produce a bisync protocol trace that can be invaluable in troubleshooting communications problems. Serengeti Systems strongly recommends that all BSCLIB application include this functionality.

Prerequisite Calls:

None

Subopcodes:

0 – Initialize
1 – Start
2 – Stop
3 – Read
4 – Reset

Input BCB Fields:

Byte 0	Opcode
Byte 2	Subopcode
Byte 22	Port (SyncPCI and SmartSync/DCP)
Byte 23	Adapter (SmartSync/DCP)

Returned BCB Fields:

Byte 1	Return Code
Byte 18	Trace Flags
Byte 19	Raw (binary) Value
Byte 20	Trace Byte #1
Byte 21	Trace Byte #2

4.4.9.1 Subopcode 0 - Initialize Trace

The Initialize function of the TRACE command prepares BSCLIB internal tracing and should be the first command issued.

4.4.9.2 Subopcode 1 - Start Trace

The Start function of the TRACE command instructs BSCLIB to start collecting trace information.

Prerequisite Calls:

INITIALIZE TRACE

4.4.9.3 Subopcode 2 - Stop Trace

The Stop function of the TRACE command instructs BSCLIB to stop collecting trace information.

Prerequisite Calls:

INITIALIZE TRACE

4.4.9.4 Subopcode 3 - Read Trace Buffer

The Read function of the TRACE command instructs BSCLIB to return one byte of trace information to your program. In SmartSync/DCP adapter environments, your application program should not use this function to dump the trace buffer. Instead you should run DCPTRACE utility simultaneously with your application.

Prerequisite Calls:

INITIALIZE TRACE
START TRACE

Returned Trace Flags:

Bit 0 – Transmit Byte:

0 = No
1 = Yes
If Transmit Byte = 1, the byte returned in Raw Value (in binary form) was transmitted to the remote station by BSCLIB.

Bit 1 – Receive Byte:

0 = No
1 = Yes
If Receive Byte = 1, the byte returned in Raw Value (in binary form) was received from the remote station by BSCLIB.

Bit 6 – Buffer Overflow:

0 = No
1 = Yes
If Buffer Overflow = 1, the beginning of the trace buffer has been overrun and an indeterminable amount of trace data has been lost. This tells your program that either the trace buffer is too small or saved trace data is not being removed from the buffer quickly enough. The byte returned on this call is valid but BSCLIB has no way to know if it is a transmitted or received character.

Bit 7 – First Byte:

0 = No
1 = Yes
If First Byte = 1, the byte returned in Raw Value is the first transmitted or received (in binary form) after a line turn around.

Returned Raw Value:

This field contains a copy of the character transmitted or received.

Returned Trace Byte #1:

This field contains an ASCII character representing the hexadecimal value of the most significant nibble of the Raw Value. For example, if an ENQ character is returned (EBCDIC 2D) this byte would be '32' which is ASCII for '2.'

The return trace bytes may be used to when writing a ASCII readable trace log file.

Returned Trace Byte #2:

This field contains an ASCII character representing the hexadecimal value of the least significant nibble of the Raw Value. For example, if an ENQ character is returned (EBCDIC 2D) this byte would be '44' which is ASCII for 'D.'

4.4.9.5 Subopcode 4 - Reset Trace

The Reset function of the TRACE command instructs BSCLIB to discard any information remaining in the trace buffer, making the entire buffer available for additional information collection.

4.4.10 Opcode 9 - CLOSE Command

This command instructs BSCLIB to end the communications session and not respond to any requests from the remote station. If the communications line is configured as a point-to-point switched line, a Disconnect Sequence (DLE-EOT) is sent to the remote station.

Prerequisite Calls:

INSTALL
OPEN

Subopcodes:

0 – Standard
1 – Leave DTR

Input BCB Fields:

Byte 0	Opcode
Byte 2	Subopcode
Byte 4,5	Flags
Byte 14,15	Time-out
Byte 16,17	Auxiliary Flags
Byte 22	Port (SyncPCI and SmartSync/DCP)

Byte 23 Adapter (SmartSync/DCP)

Returned BCB Fields:

Byte 1 Return Code

4.4.10.1 Subopcode 0 - Standard Close

The Standard function of the CLOSE command instructs BSCLIB to end the communications session and drop the DTR modem signal.

Flags:

Bit 11 – Forced Close:

0 = No
1 = Yes

If Forced Close = 1, BSCLIB ends the communications session even if a READ or a WRITE is currently in progress. Setting the Forced Close = 0 results in an error being returned if I/O is in progress.

Bit 12 – Suppress DLE-EOT:

0 = No
1 = Yes

If Suppress DLE-EOT = 1, BSCLIB will not send a Disconnect Sequence. Setting Suppress DLE-EOT = 0, sends a Disconnect Sequence when a switched line is configured.

Bit 15 – Leave DTR High:

0 = No
1 = Yes

If Leave DTR High = 1, the Standard Close function is the same as Leave DTR function.

Auxiliary Flags:

Bit 11 – Ignore DSR:

0 = No
1 = Yes

If Ignore DSR = 1, a CLOSE command to BSCLIB returns immediately to your program regardless of the state of the DSR modem signal. If Ignore DSR = 0, BSCLIB does not return to your program until DSR drops or an internal 20 second time-out occurs.

4.4.10.2 Subopcode 1 - Leave DTR

The Leave DTR function of the CLOSE command instructs BSCLIB to end the communications session exactly the same as the Standard CLOSE except the DTR modem signal is to remain high.

4.4.11 Opcode 10 - UNINSTALL Command

This command instructs BSCLIB to terminate all BSC processes, release any memory resources that may have been allocated, and shut down BSCLIB.

Prerequisite Calls:

INSTALL
CLOSE (if OPEN has been called)

Subopcodes:

None

Input BCB Fields:

Byte 0	Opcode
Byte 22	Port (SyncPCI and SmartSync/DCP)
Byte 23	Adapter (SmartSync/DCP)

Returned BCB Fields:

Byte 1 Return Code

4.4.12 Opcode 12 - HARDWARE Command

This command permits your program to determine the computer's hardware configuration. The INSTALL command does not need to be called prior to calling this command.

Prerequisite Calls:

None

Subopcodes:

0 – Get hardware adapter type
1 – Return driver serial number
4 – Raise DTR modem signal
5 – Drop DTR modem signal
8 – Return Number of DCP Adapters
9 – Set Block Response Time-Out

10 – Set Maximum Async Baud Rate
12 – Get AT Modem Dial Command String
13 – Set AT Modem Dial Command String
14 – Get AT Modem Answer Command String
15 – Set AT Modem Answer Command String
16 – Restore Default Strings
20 – Enable Buffered Reads
21 – Disable Buffered Reads
22 – Set Select Response to NAK
23 – Set Select Response to EOT
24 – Set Select Response to WACK
30 – Turn on BAPI logging
31 – Turn off BAPI logging
32 – Turn on detailed BAPI logging
33 – Turn off detailed BAPI logging
34 – Set maximum BAPI log file size

Input BCB Fields:

Byte 0	Opcode
Byte 2	Subopcode
Byte 6-9	32-Bit Buffer Address
Bytes 10-11	Parms Value
Byte 22	Port (SyncPCI and SmartSync/DCP)
Byte 23	Adapter (SmartSync/DCP)

Returned BCB Fields:

Byte 1	Return Code
Byte 3	Hardware Type

4.4.12.1 Subopcode 0 - Return Hardware Interface Type

This function returns the type of hardware adapter installed in the computer.

Returned State/Type:

The value is returned in the Type field of the BCB. BSCLIB need not be installed before the hardware type is known. The coded hardware types are:

5 – SSI SyncPlus
7 – SSI SyncPCI
11 – AutoSync
94 – 8-Port SmartSync/DCP

4.4.12.2 Subopcode 1 - Return BSCLIB Serial Number

This function returns the BSCLIB serial number.

Buffer Address:

The Buffer Address field of the BCB contains a pointer to a buffer where the serial number will be copied. YOUR PROGRAM MUST ALLOW FOR A 15 char STRING TO BE RETURNED IN THIS BUFFER.

4.4.12.3 Subopcode 4 - Raise DTR Modem Signal

This function enables your application to raise the DTR modem signal independent of the OPEN command. Use this command only when it is necessary to signal the modem or other external device – do not use it to in an attempt to establish a connection. Use the OPEN command for that.

4.4.12.4 Subopcode 5 - Drop DTR Modem Signal

This function enables your application to drop the DTR modem signal independent of the OPEN command.

4.4.12.5 Subopcode 8 - Return Number of DCP Adapters

For SmartSync/DCP adapter users, this function returns the number of available DCP adapters in your system. The value is returned in bytes 10,11 of the BCB.

4.4.12.6 Subopcode 9 - Set Block Response Time-Out

This function changes the block response time-out from its default of two seconds to the number of seconds specified in the Parms Value field of the BCB. The block response time-out is the period of time after BSCLIB transmits a data block before a response is expected.

4.4.12.7 Subopcode 10 - Set Maximum Async Baud Rate

This function sets the maximum speed at which BSCLIB attempts to communicate when dialing an AT-command set modem. The following numeric values are placed into the Parms Value field of the BCB to select the maximum speed:

12 – 1200bps
24 – 2400bps

48 – 4800bps
96 – 9600bps
19 – 19,200bps
38 – 38,400bps

4.4.12.8 Subopcode 12 - Get AT Modem Dial Command String

This function returns the current AT modem dial command string. This is the command sent to an AT-command set modem prior to dialing. The default string is:

```
ATE0X4V0&M1&C1&D2S0=0<CR>
```

E0 – no echo
X4 – dial tone & busy signal return codes
V0 – result codes as digits
&M1 – sync mode when connected
&C1 – DCD on carrier only
&D2 – DTR loss drops line
S0=0 – disable auto-answer
<CR> – carriage return (hex digit 0xd)

This command is automatically sent by BSCLIB to the modem before every dial command (ATDT...) and after each disconnect. This function enables your application to get the default (or current) modem initialization string and modify it. Use the Set AT Modem Dial Command String function to install the modified command string. This function is not supported in the AutoSync 2 version of BSCLIB.

Buffer Address:

The Buffer Address field of the BCB contains a pointer to a buffer where the modem command string will be copied. YOUR PROGRAM MUST ALLOW FOR A 255 char STRING TO BE RETURNED IN THIS BUFFER.

Returned I/O Size:

The value returned in the I/O Size field of the BCB indicates the number of bytes in the returned modem command string.

4.4.12.9 Subopcode 13 - Set AT Modem Dial Command String

This function sets the AT modem dial command string within BSCLIB to a user defined value. This is the command sent to an AT-command set modem prior to dialing. The default string is described above.

Altering the default command strings is not recommended but may be necessary to use a particular modem with BSCLIB. The Set AT Modem Dial Command String function should be issued before issuing an INSTALL command. This function is not supported in the AutoSync 2 version of BSCLIB.

Buffer Address:

The Buffer Address field of the BCB contains a pointer to a buffer containing the new modem command string. The final byte of this string MUST be an ASCII CR character (0x0d).

The following example C code snippet sets the initialization string:

```
strcpy(new_cmd, "ATE0X4V0&M1&C1&D2S0=0");
//String must end with CR!!
strcat(new_cmd, "\r");
bcb.opcode = 12;
bcb.subop = 13;
bcb.buffer = new_cmd;
callbsc(&bcb);
```

4.4.12.10 Subopcode 14 - Get AT Modem Answer Command String

This function returns the current AT modem auto-answer command string. This is the command sent to an AT-command set modem prior setting the modem into auto-answer mode. The default string is:

```
AT&S1S0=1<CR>
```

&S1 – DSR on DTR only
S0=1 – answer after 1st ring
<CR> – carriage return (hex digit 0xd)

This function enables your application to get the default (or current) auto-answer initialization string and modify it. Use the Set AT Modem Answer Command String function to install the modified command string. This function is not supported in the AutoSync 2 version of BSCLIB.

Buffer Address:

The Buffer Address field of the BCB contains a pointer to a buffer where the modem command string will be copied. YOUR PROGRAM MUST ALLOW FOR A 255 char STRING TO BE RETURNED IN THIS BUFFER.

Returned I/O Size:

The value returned in the I/O Size field of the BCB indicates the number of bytes in the returned modem command string.

4.4.12.11 Subopcode 15 - Set AT Modem Answer Command String

This function installs a new AT modem auto-answer command string within BSCLIB. This is the command sent to an AT-command set modem prior setting the modem into auto-answer mode. The default string is described above.

Modifying the default command strings is not recommended but may be necessary to use a particular modem with BSCLIB. The Set AT Modem Dial Command String function should be issued before issuing an INSTALL command. This function is not supported in the AutoSync 2 version of BSCLIB.

Buffer Address:

The Buffer Address field of the BCB contains a pointer to a buffer containing the new modem command string. The final byte of this string MUST be an ASCII CR character (0x0d).

4.4.12.12 Subopcode 16 - Restore Default Strings

This function installs restores the default AT modem dial and auto-answer command strings within BSCLIB. The default strings are described above. This function is not supported in the AutoSync 2 version of BSCLIB.

4.4.12.13 Subopcode 20 - Enable Buffered Driver Reads

This function turns on buffered reads in the device driver. This driver always has a read active and buffers the reads prior to the system read calls made by BSCLIB. This option is valid only for Windows NT/2000/XP and Unix multi-point environments. This function is typically used when the host sends poll/select sequences as separate transmissions consisting of an EOT followed by the poll/select address. This function must be called prior to issuing an OPEN command.

4.4.12.14 Subopcode 21 - Disable Buffered Driver Reads

This function turns off the buffered reads enabled by Subopcode 20 above.

4.4.12.15 Subopcode 22 - Set Select Response to NAK

This function results in BSCLIB setting the “no traffic” response to a select sequence to a NAK when operating as a tributary station in multi-point mode. The NAK response is the proper response to a select sequence in this case as defined in the original BSC protocol specification, and is the default if neither Set Select Response to EOT (subopcode 23) or Set Select Response to WACK (subopcode 24) have been issued.

4.4.12.16 Subopcode 23 - Set Select Response to EOT

This function results in BSCLIB setting the “no traffic” response to a select sequence to be an EOT when operating as a tributary station in multi-point mode. Do not use this function unless it is specifically required in your environment for BSCLIB to reply with an EOT instead of a NAK when BSCLIB is selected and there is nothing to transmit.

4.4.12.17 Subopcode 24 - Set Select Response to WACK

This function results in BSCLIB setting the “no traffic” response to a select sequence to be a WACK when operating as a tributary station in multi-point mode. Do not use this function unless it is specifically required in your environment for BSCLIB to reply with a WACK instead of a NAK when BSCLIB is selected and there is nothing to transmit.

4.4.12.18 Subopcode 30 - Turn On BAPI Debug Logging

This function turns on BAPI logging. This logging records all BAPI calls and corresponding parameters on entry and then on exit, and writes this information to a file. This logging will normally not be necessary, but it can be helpful in troubleshooting problems.

4.4.12.19 Subopcode 31 - Turn Off BAPI Debug Logging

This function turns off BAPI logging.

4.4.12.20 Subopcode 32 - Turn On Detailed BAPI Debug Logging

This function enables detailed BAPI logging. Detailed logging records more information than “standard logging”, including all BAPI calls, call parameters and call exits.

4.4.12.21 Subopcode 33 - Turn Off Detailed BAPI Debug Logging

This function turns off detailed BAPI logging.

4.4.12.22 Subopcode 34 - Set Maximum BAPI Log File Size

This function specifies the maximum size of the BAPI log file. The default size is 20MB.

Parms Value:

The Parms Value field of the BCB contains the a numeric value specifying maximum size in the BAPI log file expressed in megabytes.

(Blank Page)

5 CONFIGURING BSCLIB

The 32-bit pointer to the structure shown in Figure 10 is passed to BSCLIB on an INITIALIZE command. The fields within the Parameter Initialization Block (PIB) structure define operating characteristics of BSCLIB. All fields, except for byte 2 and the Terminal ID, which is a 20-byte string, are 16-bit unsigned shorts.

By setting Partial Update = 1, the INITIALIZE command can be called at anytime to change the Parameters in bytes 16-33 of the PIB – even when a communication session is in progress. The remaining fields, however, cannot be changed after the INSTALL command is issued. Issue an UNINSTALL command prior to attempting to change these fields.

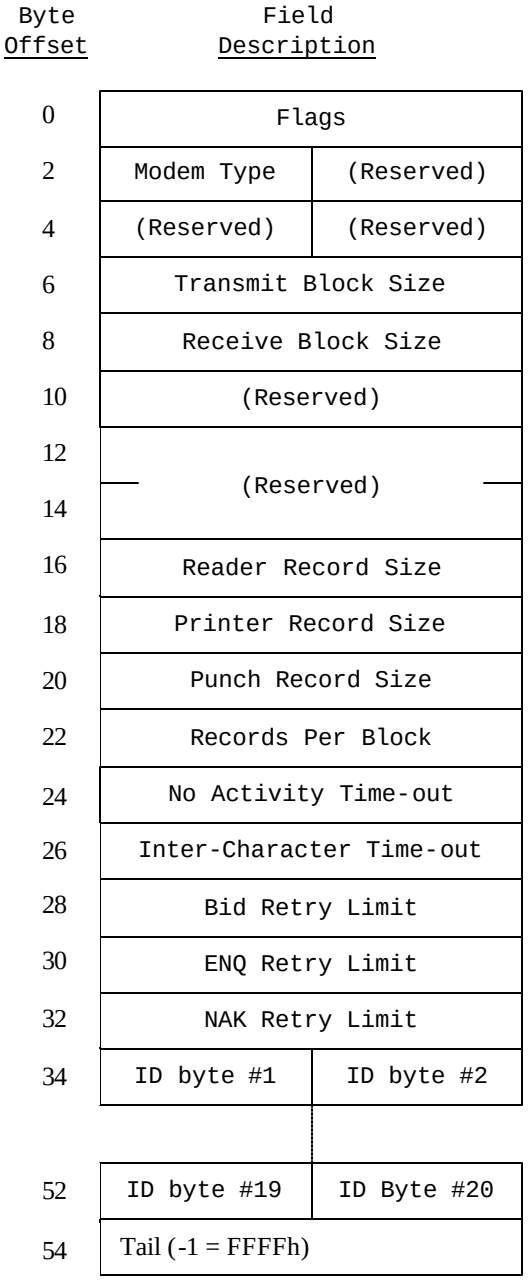

Figure 9. PIB Structure

Flags:

Bit 0 – Emulation Type:

0 = 2780
1 = 3780 (Default)

This flag selects the desired type of BSC protocol to be used when BSCLIB initiates a session with the remote system by transmitting first. This selection must match the protocol used by the remote station. In multi-point environments, select the default.

When receiving first, BSCLIB automatically detects either the 2780 or 3780 protocol and handles the inbound data stream accordingly. In addition, the proper protocol to use is set until the end of the session (i.e., the line disconnects). No specific action is required by your application in this case.

Bit 1 – Station Type:

For point-to-point operation:
0= Secondary
1= Primary

For multi-point operation:
0= Tributary
1 = Control

When configured for point-to-point operation, this flag is used to determine whether BSCLIB or the remote station yields first in a contention situation for control of the line. The Primary station bids for the line once a second while a Secondary station bids once every three seconds – this difference prevents continuous bid collisions.

When configured for multi-point operation, this flag defines whether the BSCLIB application is to be the Control or Tributary station.

Bit 2 – Line Type:

0 = Leased
1 = Switched (Default)

When Line Type = 1, the connection to the remote station is through a dial-up (switched) line, otherwise it is through a leased (or direct) connection. If a switched line is selected in a point-to-point environment, BSCLIB will recognize and send DLE-EOT disconnect sequences when a communication session is terminated. This option is ignored in multi-point mode.

Bit 3 – Terminal ID:

0 = No (Default)
1 = Yes

If Terminal ID = 1, BSCLIB includes a Terminal ID string in the first Line Bid or positive bid acknowledgment of a communication session. A Terminal ID may consist of up to 20 characters and is defined elsewhere in the PIB. This option is ignored in multi-point mode.

Bit 4 – (Reserved)

Bit 5 – Full-Duplex:

0 = No (Default)
1 = Yes

If Full Duplex = 1, BSCLIB operates in full-duplex, or 4-wire, mode. This is also known as "constant carrier" mode. Appropriate hardware and any hardware jumper settings must be present for this mode to operate. The significant difference between half-duplex (2-wire or "switched carrier" mode) and full-duplex (4-wire) is BSCLIB doesn't wait for the CTS modem signal prior to transmitting – this effectively eliminates line turn-around delays and thus increasing data throughput.

Bit 6 – (Reserved)

Bit 7 – Auto Read Enable:

0 = No (Default)
1 = Yes

If Auto Read Enable = 1 and point-to-point mode is configured, BSCLIB automatically issues a READ command at times after the communications line has been opened but the application program does not have a read or write in progress. The sole purpose of this is to detect a DLE-EOT disconnect sequence that may arrive unexpectedly. This special read is automatically killed when the application program issues a READ, WRITE, or CLOSE command. If a DLE-EOT is detected while the automatic read is active, your program will be notified on the next READ, WRITE, CLOSE, or STATUS command. This option is ignored in multi-point mode.

Bit 8 – Enable ASCII Data Link Control (CRC-16):

0 = No (Default)
1 = Yes

If Enable ASCII DLC = 1, BSCLIB configures itself for ASCII Data Link Control (DLC) operation using CRC-16 block check characters. In this mode all BSC protocol characters and data bytes are encoded using the ASCII character set. The default is the EBCDIC character set with BSCLIB automatically performing the ASCII↔EBCDIC translation where necessary.

Bit 9 – Use LRC With ASCII Data Link Control:

0 = No (Default)
1 = Yes

When Use LRC With ASCII DLC = 1 and Enable ASCII DLC = 1, BSCLIB sets itself for ASCII Data Link Control operation using an LRC block check character. This flag is ignored if Enable ASCII DLC = 0.

LRC is an alternate single character block check character often used instead of CRC-16 in ASCII data link control environments.

Bit 10 – Enable Multi-Point Mode:

0 = No (Default)
1 = Yes

If Enable Multi-Point Mode = 1, BSCLIB configures itself for multi-point operation. The default is point-to-point operation.

Bit 11 – Parity Control:

0 = n/a (Default)
1 = Odd

If Parity Control = 1, BSCLIB uses odd parity when using ASCII Data Link Control. The default is no parity. This flag is ignored if Enable ASCII DLC = 0.

Bit 12 – Recognize Select When WRITE Is Pending:

0 = No (Default)
1 = Yes

When BSCLIB is operating as a tributary station in a multi-point environment, setting Recognize Select = 1 causes BSCLIB to terminate a pending WRITE by returning result code 111 to your application whenever a select sequence is detected that matches a BSCLIB address. Your application may then issue a READ to respond to the *next* select sequence, or ignore the select by reissuing the WRITE and awaiting the next poll sequence.

Bit 14 – Partial Update:

0 = No (Default)
1 = Yes

If Partial Update = 1 on an INITIALIZE command only bytes 16 through 33 of the PIB are updated. All other fields in the PIB are ignored.

Modem Type:

Default: 1

When configured to use an external modem, this value tells BSCLIB the type of modem attached to the hardware adapter. The permitted values are:

1 – Motorola/UDS models 201C/D, 208B/D, 2140, or 2860
2 – manual dial, other, or none
3 – Racal-Vadic model 4850PA
5 – AT-command set (e.g., Hayes Optima)
8 – V.25bis compatible modem

Check the **readme.1st** file for Windows or **READ.ME.FIRST** file for Unix on your distribution media for other modem types that may be supported in the latest release of BSCLIB.

Transmit Block Size:

Default: 512

Range: 10 - 4096

This is the maximum size in bytes of the buffer into which BSCLIB copies data records before sending the buffer as a data block. Typically this size should be 512 for 3780 emulation and 400 for 2780 emulation. This size should be set to exactly match the receive block size of the remote station.

Receive Block Size:

Default: 512

Range: 10 - 4096

This is the maximum size in bytes of a data block that BSCLIB will receive from the remote station. Typically this size should be 512 for 3780 emulation and 400 for 2780 emulation. This size should be at least as large as the transmit block size of the remote station.

Reader Record Size:

Default: 80

Range: 1 - Transmit Block Size

This is the maximum size in bytes of each data record that will be copied into BSCLIB's buffer for transmission. In non-transparent operation, this size need only be set to accommodate the largest record that may be sent; however, in transparent operation, this size becomes important as both the transmitting and receiving stations must agree to the record size to facilitate the correct deblocking of received records.

Also, if 2780 emulation is selected and space truncation is not selected, BSCLIB pads all outbound records with spaces so all records are of this size.

Printer Record Size:

Default: 132

Range: 1 - Receive Block Size

This is the maximum size in bytes of each printer data record passed to your application from data blocks received from the remote station. In non-transparent operation when the inbound data stream is formatted with record separators, set the printer record size equal to the receive block size to insure correct printer data formatting. In transparent operation, this size must agree with the transmitting station's reader record size to facilitate the correct deblocking of received records.

Punch Record Size:

Default: 80

Range: 1 - Receive Block Size

This is the maximum size in bytes of each punch data record passed to your application from data blocks received from the remote station destined for the punch. In non-transparent operation when the inbound data stream is formatted with record separators, set the punch record size equal to the receive block size to insure correct punch data formatting. In transparent operation, this size must agree with the transmitting station's reader record size to facilitate the correct deblocking of received records.

Records Per Block:

Default: 0

This allows automatic transmission of a data block upon buffering the specified number of records. A value of 0 instructs BSCLIB to transmit when a buffer is full. Depending on the configuration of

the remote station, rules of 2780 emulation may restrict the number of records per block to a maximum of two or seven.

No Activity Time-out:

Default: 0

This value, in tenths of seconds, specifies the maximum period of time that may pass without any communications line activity before BSCLIB returns a time-out error. A value of 0 instructs BSCLIB to disable this time-out.

Inter-Character Time-out:

Default: 180

This value, in tenths of seconds, specifies the maximum period of time BSCLIB should allow for a single character to be transmitted or received before reporting an error. A value of 0 instructs BSCLIB not to time-out. A non-zero value should always be passed in this field. The occurrence of an inter-character time-out generally indicates a hardware failure.

Bid Retry Limit:

Default: 15

In point-to-point operation, this value represents the number of line bids that BSCLIB will transmit in an attempt to start a WRITE command. A value of 0 instructs BSCLIB to send Line Bids indefinitely.

In multi-point control station operation, this value is used to control the number of polls or selects sent that are either not responded to or receive an EOT or NAK response. For example, if you wish to send a single poll set this value equal to 1. If the tributary station does not reply with data, result code 102 is returned to your application. The BCB time-out value is used to control the period of time BSCLIB waits for a response to the poll or select.

If this value is greater than one and the tributary station is responding negatively (EOTs or NAKs are being sent), BSCLIB will retransmit the corresponding polls or selects in as rapid succession as possible – your BSCLIB application does not have control over the time interval. If a precise poll or select interval is required, you should set the “bid retry count” to 1 and perform the necessary timing in the application.

ENQ Retry Limit:

Default: 6

This value represents the number of ENQs that BSCLIB will transmit in an attempt to solicit a response after sending a data block or while waiting for an ACK after a WACK was received from the remote station during a WRITE command. A value of 0 instructs BSCLIB to send ENQs indefinitely.

NAK Retry Limit:

Default: 6

This value represents the number of times BSCLIB will retransmit a data block to which the remote station has responded with a NAK. A value of 0 instructs BSCLIB to retransmit indefinitely.

Terminal ID:

The Terminal ID may be required by some remote stations and is a string of characters up to 20 bytes long. If the desired string is less than 20 characters, a byte containing a value of 0 must follow the last character in the string. Additionally, the length of the ID (excluding the trailing 0 byte) must be passed in the Buffer Size field of the BCB on an INITIALIZE command. The Terminal ID and Buffer Size fields are only used when Flags Bit 3 is set to 1. This option is ignored in multi-point mode.

Tail:

This field must be set to -1. BSCLIB uses this field to validate the size of the PIB.

Appendix A. BSCLIB Return Codes

The following is a list of the return codes returned in Byte 1 of the BSC Control Block (BCB) or as a return code from a BSCAWL function. Note that some of these codes are produced internally within BSCLIB and may not appear to your program.

Base return codes:

255 Requested I/O command in progress
0 Command completed successfully
1 Duplicate driver installation
2 No hardware found
3 Invalid modem I/O port
4 Invalid modem hardware interrupt
5 BSC driver not installed
6 Invalid opcode
7 Port not open
8 Port is open (on uninstall)
9 Port not installed
10 Specified I/O not in progress (on kill I/O)
11 Command time-out
12 I/O aborted
13 Read/write already in progress
14 Port owned by another process
19 Invalid hardware type; port not defined in Registry
20 Already open
21 Open in progress
22 Cannot load protocol handler process
24 Invalid port number
25 CTS modem signal lost
26 DSR modem signal lost
27 Zero read/write count specified
28 Invalid external modem type
35 Shared memory alignment error; possible version mismatch
36 Receive character(s) lost
37 Receive FIFO overflow
38 Shared memory error
40 Operation not permitted
41 Receive character(s) lost

BSCAWL return codes:

80 Select address list empty
81 Select address list full
82 Select address not found in list
83 Duplicate address found in list
89 Buffer allocation error
90 Read initiated (non-fatal notification)
91 File read error
92 Currently installed
93 Cannot create mutex
94 File is empty (on **BSCSendFile**)
95 File open error
96 Invalid parameter found
97 Context (hBC) not defined
98 Fatal internal mutex error
99 Fatal internal error

Return codes from BSC protocol handler:

100 Port not open
101 Transmit initiated but no buffer ready to send
102 Bid retry error (point-to-point mode)
Poll/Select response retry error (multi-point mode)
103 DLE-EOT disconnect sequence received
104 No activity time-out occurred
105 RVI received
106 ENQ retry error
107 NAK retry error
Received NAK response(s) to select (multi-point mode)
108 Received forward abort / transmitter error
109 Abnormal receive termination
110 No buffer ready on conversational reply
111 Received a conversational reply (point-to-point mode)
WRITE aborted due to received select (multi-point mode)
112 Received EOT instead of ACK (point-to-point mode)
Received EOT response(s) to poll (multi-point mode)
113 Transmission complete
114 DLE-EOT transmitted
115 Receive successfully aborted
116 Receive completed successfully
117 Received SYN as data in normal text mode

118 Receive buffer overflow
119 Unable to send line bid
120 Read for first line bid timed out
121 Transmit successfully aborted
122 Printer selected but is disabled
123 Punch selected but is disabled
124 Received ENQ response to transmitted line bid
125 Too many consecutive WACKs transmitted
126 Receive FIFO underrun
127 Too many consecutive TTDs transmitted
128 EOT transmitted
129 Read enable error
130 No free buffer available when line bid received
131 Abort cancelled due to pending receive data
132 Send conversational reply now
150 BSCLIB reentered / memory model mismatch
160 Could not start BSC Process (DCP)
161 Could not end BSC Process (DCP)
162 BSC Process already active for port (DCP)
163 BSC Process launch timed out (DCP)
164 BSC Process already terminated (DCP)
165 Bad port on BSC Process launch (DCP)
166 Could not start BSC Process (DCP)
170 Failed to load **abscdrvr** AutoSync 2 process

Return codes from Auto-Dialer routines:

180 Ring detected
181 Busy signal detected
182 Dial tone detected
183 No dial or answer back tone detected
184 Answer back tone detected
185 Modem command error
186 Dialer time-out
187 Dial command aborted
188 Dial command complete
189 Modem command not supported
190 Unknown modem error
191 Cannot enable auto-answer

Return codes from Record Manager layer:

200 Duplicate open call
201 Open in progress
202 Open previously completed
203 Read in progress
204 Write in progress
205 Invalid subopcode
206 Write not in progress
207 No transmit buffers available; retry WRITE after short delay
208 Transmit buffer overflow
209 Reader record overflow
211 Read not in progress
213 Abort in progress
214 No I/O in progress on abort or status call
215 I/O in progress
216 Transmit aborted
217 Receive aborted
218 Invalid BCB structure
219 Some parameters ignored because driver installed (warning)
220 Trace already enabled
221 Trace buffer too small
222 Trace not enabled
223 Trace buffer empty
224 Trace buffer too large
225 **emubsc** daemon process fails to respond
226 Unable to locate **emubsc** shared memory
227 Unable to detach shared memory
228 Failed to kill I/O on forced close call
229 Invalid serial number found

Return codes from Configuration routines:

230 Invalid PIB structure
231 Invalid configuration flags
234 Invalid transmit buffer size
235 Invalid receive buffer size
236 Reader size cannot exceed transmit buffer size
237 Printer size cannot exceed receive buffer size
238 Punch size cannot exceed receive buffer size
240 Invalid PIB structure

Return codes from Translation table patching routines:

251 Translate table space too small

Appendix B. BSCLIB Link State Codes

The state codes shown below reflect the internal state of BSCLIB. The code is returned in Byte 3 of the BCB at the conclusion of each function call and in the first field of Statistics Information Block (SPB) of the Read Statistics function of a STATISTICS command. Since your application program is shielded from this level of detail of the BSC communications link, these state codes are returned for informational and diagnostic purposes only, and should not be used to determine program function.

State Name	State Code	State Description
BIDSENT	0	Waiting for bid response
BIDRESP1	1	Read first char after line bid
BIDRESP2	2	Read 2nd char after line bid
TXIDRESP	3	Accept terminal ID in bid reply
TTDRPLY	4	Here after TTD is sent
TTDRESP	5	Read for 1st byte of TTD reply
TTDRESP2	6	Read for 2nd byte of TTD reply
XMTDEOT	7	Here after EOT is sent
BUFRXMTD	8	Here after comm buffer is sent
BLKRESP1	9	Read 1st block response
BLKRESP2	10	Read 2nd block response
ENQRPLY	11	Read for delay 2780 ENQ reply
WAITBID	14	Read for line bid
DLERESP2	15	Possibly received a DLE-EOT
RXIDRESP	16	Accept terminal ID with bid
RCVBLK	17	Just sent ACK/0, wait for blk
XMTACK	18	Ready to xmt ACK/0 or ACK/1
WACKRPLY	19	Here after WACK is sent
WACKRESP	20	Read for WACK reply
RCV1ST	21	Read for 1st char in data block
DLERCVD	22	Read a DLE as 1st char of block
RCVTEXT	23	Receiving data bytes
WACKDLE	24	Read a DLE as WACK response
WAITCRC	25	Waiting to test received CRC
RXIDL	27	Common receive exit routine
XMTDISC	28	Here after sent DLE EOT
WAITDISC	29	Reading for DLE of DLE EOT
WAITDSC2	30	Reading for EOT of DLE EOT
OPENCOMP	31	Non-blocking open has completed
DIALCOMP	32	Command written to dialer
DIALRESP	33	Awaiting reply from dialer

Appendix C. Statistics Parameter Block (SPB)

The values in the Statistics Information Block (SPB) structure shown below are returned by the Read Statistics function of the STATISTICS command. All fields, except for the BSCLIB Link State, are 16-bit integer counts of the number of occurrences of the indicated event.

The values in the SPB have different meanings depending on the mode in which BSCLIB is configured. See Figures 11, 12, and 13.

Byte Offset	Field Description
0	BSCLIB Link State
2	Bids Sent
4	Bids NAK'ed
6	Bids ACK'ed
8	Bids WACK'ed
10	Bids Ignored
12	Unknown Responses
14	TTDs Sent
16	RVIs Received
18	WACKs Received
20	Forward Aborts Sent
22	Blocks NAK'ed
24	Blocks Sent
26	Records Sent
28	Unknown Data Received
30	Bids Received
32	Bids ACK'ed
34	WACKs Sent
36	Blocks Received
38	NAKs Sent
40	Records Received
42	Tail (-1 = FFFFh)

Figure 10. SPB Structure for Point-to-Point Mode

Byte Offset	Field Description
0	BSCLIB Link State
2	Selects Sent
4	Selects NAK'ed
6	Selects ACK'ed
8	Selects WACK'ed
10	Selects Ignored
12	Unknown Responses
14	TTDs Sent
16	RVIs Received
18	WACKs Received
20	Forward Aborts Sent
22	Blocks NAK'ed
24	Blocks Sent
26	Records Sent
28	Unknown Data Received
30	Polls Sent
32	n/a
34	WACKs Sent
36	Blocks Received
38	NAKs Sent
40	Records Received
42	Tail (-1 = FFFFh)

Figure 11. SPB Structure for Multi-Point Mode (Control Station)

Byte Offset	Field Description
0	BSCLIB Link State
2	Polls Received
4	n/a
6	n/a
8	n/a
10	n/a
12	Unknown Responses
14	TTDs Sent
16	RVIs Received
18	WACKs Received
20	Forward Aborts Sent
22	Blocks NAK'ed
24	Blocks Sent
26	Records Sent
28	Unknown Data Received
30	Selects Received
32	Selects ACK'ed
34	WACKs Sent
36	Blocks Received
38	NAKs Sent
40	Records Recieved
42	Tail (-1 = FFFFh)

Figure 12. SPB Structure for Multi-Point Mode (Tributary Station)

BSCLIB Link State:

See Appendix B for Link State values and descriptions.

Bids Sent:

Selects Sent:

Polls Received:

In point-to-point mode, this is the number of Line Bids transmitted requesting control of the line and permission to send data to the remote station.

In multi-point control station mode, this is the number of select sequences transmitted requesting permission to send data to the remote tributary station.

In multi-point tributary mode, this is the number of times a poll sequence has been received requesting transmission of data to the remote control station.

Bids NAK'ed:

Selects NAK'ed:

In point-to-point mode, this is the number of Line Bids sent which the remote station responded to with a NAK, denying permission to send data.

In multi-point control mode, this is the number of times the remote tributary station responded with a NAK, denying permission to send data.

Bids ACK'ed:

Selects ACK'ed:

In point-to-point mode, this is the number of Line Bids sent which the remote station responded to with a ACK, giving permission to send data.

In multi-point control mode, this is the number of select sequences sent which the remote station responded to with a ACK, giving permission to send data.

Bids WACK'ed:

Selects WACK'ed:

In point-to-point mode, this is the number of Line Bids sent to which the remote station responded with a WACK, denying permission to send data.

In multi-point control mode, this is the number of select sequences sent to which the remote station responded with a WACK, denying permission to send data.

Bids Ignored:

Selects Ignored:

In point-to-point mode, this is the number of Line Bids sent to which no response was received from the remote station, possibly indicating that the remote station has abnormally ended the communications session.

In multi-point control mode, this is the number of select sequences sent to which no response was received from the remote station, possibly indicating that the remote station has abnormally ended the communications session.

Unknown Responses:

The number of unrecognized responses received from the remote station. This may indicate a poor quality communications link or that the remote station is using an incompatible protocol.

TTDs Sent:

The number of TTDs sent indicating that there may have been long delays between WRITE commands.

RVIs Received:

The number of RVIs received from the remote station, requesting that the application end the WRITE command and issue a READ to receive data.

WACKs Received:

The number of WACKs received in response to a data block or ENQ. Reception of a WACK instructs BSCLIB to send ENQs until the remote station responds with an ACK at which time BSCLIB may send another data block.

Forward Aborts:

The number or Forward Aborts transmitted, instructing the remote station that BSCLIB will send no more data and to disregard the data transmitted since the last Line Bid.

Blocks NAK'ed:

The number of data blocks transmitted to which the remote station responded with a NAK, requiring retransmission of the block. If

this value is large in relation to the number of blocks sent, the communications line or equipment may be corrupting the data link.

Blocks Sent:

The number of data blocks transmitted.

Records Sent:

The number data records buffered and transmitted in data blocks.

Unknown Data Received:

The number of unrecognized transmissions and responses received from the remote station. This may indicate a noisy communication link or that the remote station is using an incompatible protocol.

Bids Received:

Polls Sent:

Selects Received:

In point-to-point mode, this is the number of Line Bids received from the remote station, requesting control of the line and permission to send data.

In multi-point control station mode, this is the number of poll sequences transmitted requesting transmission of data from the remote tributary station.

In multi-point tributary mode, this is the number of times a select sequence has been received requesting data to sent to the remote control station.

Bids ACK'ed:

Selects ACK'ed:

In point-to-point mode, this is the number of Line Bids received from the remote station to which BSCLIB responded with an ACK, indicating that BSCLIB was ready to receive data.

In multi-point tributary mode, this is the number of select sequences received from the control station to which BSCLIB responded with an ACK, indicating that BSCLIB was ready to receive data.

WACKs Sent:

The number of Line Bids received from the remote station to which BSCLIB responded with an WACK, indicating that BSCLIB's data buffers were full and that the remote station should not send any more data. This may indicate that the application has

delayed for long periods of time between READ operations, thus allowing BSCLIB's data buffers to become full.

Blocks Received:

The number of data blocks received.

NAKs Sent:

The number of data blocks received to which BSCLIB responded with a NAK, requiring retransmission of the block by the remote station. The NAK response may be the result of an erroneous CRC being received or a receive buffer may have overflowed.

Records Received:

The number data records received in data blocks from the remote station.

Appendix D. KILLBSC – Terminate EMUBSC

This program is used to properly terminate the **emubsc** daemon. It should be run at the end of any shell scripts that load a BSCLIB application. You may also run it directly from the command line to kill the daemon and remove it from memory. The optional command line options for **killbsc** are described below.

-f Forced kill

The **-f** switch may be used to force the termination of **emubsc** when it is not in the idle state.

-b *board* **Board number [1-6] (SmartSync/DCP only)**

The **-b** switch specifies which SmartSync/DCP associated with the BSC process to be killed. You must indicate the board number, 1 through 6, as appropriate. The board number corresponds to the order in which they were configured. If omitted, the default is board 1.

-p *port* Port number [1-8]

The -p switch specifies the port associated with the BSC process to be killed. If omitted, the default is port 1.

-s Silent mode

The -s switch suppresses all output to the terminal with the exception of error messages.

-i *id* **Alternate shared memory ID (Unix only)**

The **-i** switch must be used to pass the alternate shared memory identifier if one was used when **emubsc** was loaded.

Appendix E. EMUBSC – BSC Protocol Handler Process

The **emubsc** process is the background BSC protocol handler for applications using AutoSync or SyncPCI communications adapters. It is not applicable to SmartSync/DCP installations. The process is automatically started by the BSCLIB INSTALL function. On Unix, the program file is expected to be found in BSCLIB install directory.

Under certain conditions Serengeti Technical Support may instruct you to start **emubsc** manually using the following command or by simply clicking on the program icon in Windows:

```
emubsc -d -v
```

The **emubsc** command line options are described below:

-p *port* **Port number [1-8]**

The **-p** switch specifies the port associated with the BSC process to be killed. If omitted, the default is port 1.

-d Debug mode

The -d switch activates the debug option which writes internal debug information to a file named "emubsc.1".

-f *path* Alternate path to abscdrvr executable (Unix only)

The -f switch is used to specify an alternate path to the **abscdrvr** executable if it is not in the default **/usr/lib/bsclib** directory.

-i *id* Alternate shared memory ID (Unix only)

The -i switch is used to specify an alternate shared memory identifier when the default value of 311 is used by another process. In such cases, your application must specify the same shared memory ID specified with the -i switch.

-m *size* Maximum size of emubsc.1 file
By default, when **emubsc** is writing to **emubsc.1** the file is not allowed to grow larger than approximately 100K bytes. The -m switch changes the maximum size of **emubsc.1.** For example, -m50 sets the maximum file size to 50K bytes.

-s Silent mode (Unix only)

The -s switch suppresses all output to the terminal with the exception of error messages. Use this switch if you run your BSCLIB application from the same terminal used to load **emubsc**.

-t Block response time-out

The -t switch is provided to change the time-out used when waiting for a reply to a transmitted data block. The BSC protocol defines that this time-out be three seconds, but the -t switch enables you to change this time-out if necessary within your environment. For example, -t5 changes the block response time-out to five seconds.

-v Verbose mode

The -v switch must be used in conjunction with the -d switch to cause debug messages to be output to the display in addition to the debug file.

Appendix F. DCPLOAD – Load Process on DCP

This program is used to initialize each SmartSync/DCP co-processor and download the **amxbsc.bin** file. The **amxbsc.bin** file is expected to be found in the current subdirectory. The optional command line switches for **dcpload** are described below:

-b *board* **Specific board number to load [1-6]**

The **-b** switch specifies which SmartSync/DCP adapter **dcpload** is to access. You must indicate the board number, 1 through 6, as appropriate. The board number corresponds to the order in which they were configured. If omitted, **dcpload** loads all boards found.

-f *name* **Path/File to download**

The **-f** switch specifies the full path name of **amxbsc.bin** if it is not in the current subdirectory. For example:

```
dcpload -f /MyApp/amxbsc.bin
```

-v **Verbose load (Unix only)**

The **-v** switch causes **dcpload** to display detailed information during downloading. Use this switch the first few times you download **amxbsc.bin** to become familiar with the process and observe **dcpload** in operation.

Unix systems can be configured to automatically initialize SmartSync/DCP adapters during system startup. Below is an example of how to configure a Linux system by adding the following commands to the end of the **/etc/rc.d/rc** system file. If you are also activating the SmartSync/DCP device driver in the **rc** file, make sure that the **dcpload** command is after the **insmod**.

```
/sbin/insmod –f /usr/lib/dcplib/driver/xdcpdrvr.o
/usr/lib/dcplib/dcpload -f /usr/lib/dcplib/amxbsc.bin
```

Appendix G. DCPPEEK - DCP Process Status

This program is used to "peek" at each SmartSync/DCP and verify that the various processes running in the co-processor are active. This utility is intended primarily for trouble-shooting purposes, but can be run at any time to check the status of BSC processes. The optional command line switch for **dcppeek** are described below:

-b *board* **Specific board number to peek at [1-6]**

The **-b** switch specifies which SmartSync/DCP **dcppeek** is to access. If you indicate the board number specify 1 through 6, as appropriate. The board number corresponds to the order in which they were configured. If omitted, **dcppeek** displays status of all boards found.

On Windows, **dcppeek** opens a GUI Window. On Unix, results are output to the current display. The following is an example of output from **dcppeek** on Unix after running **dcpload**:

```
SmartSync/DCP Peeker v4.x.x
Copyright (C) 1993-2002 Serengeti Systems Incorporated.
ALL RIGHTS RESERVED.

Total of 1 SmartSync/DCP adapter(s) found:

    Ports Available: 8
Memory Installed: 1024K
    Board Address: bfff7800
 Window Address: 80104000

BSC Emulation Downloaded

BSC Task Monitor Running
BSC Interrupt Clock Running

BSC Process #1 Not Running
BSC Process #2 Not Running
BSC Process #3 Not Running
BSC Process #4 Not Running
BSC Process #5 Not Running
BSC Process #6 Not Running
BSC Process #7 Not Running
BSC Process #8 Not Running
```

The results are repeated for each SmartSync/DCP found unless the -b switch is used to specify a specific board.

The BSC Processes can have the following states: Not Running, Running in IDLE State, Running in XMT State, Running in RCV State, and Not Running in IDLE State (this is an error state).

The BSC Task Monitor and Interrupt Clock should always be 'Running' during normal operation. If the BSC Task Monitor or Interrupt Clock is ever in the 'Not Running' state, you can run **dcpload** to reinitialize the SmartSync/DCP board. This is also true if one the BSC Processes is hung in one of the 'Running' states (i.e., **killbsc** is unable to return it to the 'Not Running' state).

Appendix H. DCPDUMP - Dump DCP Debug

This program is used to access the onboard memory and registers of SmartSync/DCP. This utility is strictly for diagnostic purposes and should only be used under the direction of Serengeti Technical Support.

There are no command line options for this utility program.

Appendix I. DCPDEBUG - DCP Debug

This program is used to access debug messages generated by SmartSync/DCP BSC processes. This utility is used strictly for diagnostic purposes and should only be used under the direction of Serengeti Technical Support.

The debug messages obtained by **dcpdebug** are automatically written to both the screen and a file. If you've configured one SmartSync/DCP, the default file name is **debug.***x* where *x* is the port number. If you've configured more than one, the default is **debug.***xy* where *x* is the port number and *y* is the board number. By default the output file is reset when it reaches 100,000 bytes. This enables **dcpdebug** to run continuously without overflowing disk storage. The optional command line switches for **dcpdebug** are described below:

-b *board* **Board number to debug [1-6]**

The **-b** switch specifies which SmartSync/DCP **dcpdebug** is to access. You must indicate the board number, 1 through 6, as appropriate. The board number corresponds to the order in which they were configured. If omitted, **dcpdebug** defaults to board 1.

-p *port* **Port number to debug [1-8]**

The **-p** switch specifies which of the SmartSync/DCP ports to monitor for debug messages. You must indicate the port number, 1 through 8, as appropriate. If omitted, **dcpdebug** defaults to monitor port 1.

-f *name* **Alternative debug file name (Unix only)**

The **-f** switch specifies a file, other than the default, to record debug messages.

-a *name* **Append to debug file name (Unix only)**

The **-a** switch appends debug messages to an existing file.

-m *size* **Maximum size of debug file (kilobytes)**

The **-m** switch changes the default maximum output file size. By default the debug file is reset when it exceeds 100,000 bytes. To change this default, specify the maximum file size in kilobytes (1,024 bytes).

-s Silent mode

The **-s** switch prevents **dcpdebug** from echoing debug messages to the screen.

-l Priority mode (Windows only)

The **-l** switch causes **dcpdebug** to run at a higher priority level. If the message "DEBUG TRACKING MESSAGES LOST" appears on the screen or in your debug output file when running **dcpdebug**, first try using silent mode. If silent mode does not prevent the messages from appearing, use priority mode. When using priority mode, be sure to run **dcpdebug** in the background to lessen the effect on other running processes.

Appendix J. DCPTRACE - DCP Trace Dump

This program is used to access the line trace buffers updated by SmartSync/DCP BSC processes. This utility is strictly for diagnostic purposes and should only be used under the direction of Serengeti Technical Support.

In other versions of BSCLIB, reading the trace buffer is done through a BSCLIB function call. With the SmartSync/DCP dumping of the line trace information is handled by running **dcptrace**. Your program is still responsible for initializing the trace buffer and turning the line trace on.

The trace data retrieved by **dcptrace** is automatically written to both the screen and an output file. If you've configured one SmartSync/DCP, the default file name is **trace.***x* where *x* is the port number. If you've configured more than one, the default is **trace.***xy* where *x* is the port number and *y* is the board number. By default the output file is reset when it reaches 100,000 bytes. This enables **dcptrace** to run continuously without overflowing disk storage. The optional command line switches for **dcptrace** are described below:

-b *board* **Board number to trace [1-6]**

The **-b** switch specifies which SmartSync/DCP **dcptrace** is to access. You must indicate the board number, 1 through 6, as appropriate. The board number corresponds to the order in which they were configured. If omitted, **dcptrace** defaults to board 1.

-p *port* **Port number to trace [1-8]**

The **-p** switch specifies which of the SmartSync/DCP ports to monitor for trace data. You must indicate the port number, 1 through 8, as appropriate. If omitted, **dcptrace** defaults to monitor port 1.

-f *name* **Alternative trace file name**

The **-f** switch specifies a file, other than the default, to record trace data.

-a *name* **Append to trace file name (Unix only)**

The **-a** switch appends trace data to an existing file.

-m *size* **Maximum size of trace file (kilobytes)**

The **-m** switch changes the default maximum output file size. By default the trace file is reset when it exceeds 100,000 bytes. To change this default, specify the maximum file size in kilobytes (1,024 bytes).

-s Silent mode

The **-s** switch prevents **dcptrace** from echoing trace messages to the screen. On Windows, this option will decrease CPU utilization of the program. Also on Windows, this option can be enabled by unchecking the "Echo to Screen" checkbox under the "Options" menu.

Appendix K. XRESET – Reset SyncPCI Device Driver

This utility only applies to applications on Unix using the SyncPCI adapter. When an unexpected termination of **emubsc** leaves the XBSC Device Driver in an unresponsive state, use the **xreset** utility to close all devices associated with BSCLIB and reset the internal line trace buffer. This usually restores the XBSC Device Drivers to a working state.

WARNING: Running **xreset** will disconnect all active BSCLIB sessions.

The optional command line switches for **xreset** are described below:

-f Forced reset

The **-f** switch forces a driver reset when xreset indicates the driver is in use.

If you encounter XBSC Device Driver related errors when loading your BSCLIB application, run **xreset** and then reload you application. If the problem persists, reboot your system.

2

A

B

M

N

O

P

U

V

W

X

(Blank Page)