Bit-Stream Protocol Programmer's Guide

DC 900-1574A

Simpact, Inc.
9210 Sky Park Court
San Diego, CA 92123
April 1999

SIMPACT

Simpact, Inc.
9210 Sky Park Court
San Diego, CA 92123
(619) 565-1865

Bit-Stream Protocol Programmer's Guide

Contents

List of Figures 7

List of Tables 9

Preface 11

1 Introduction 15

1.1 Protocol Overview . . . 15
1.2 Supported Hardware . . . 15
1.2.1 Freeway Server . . . 15
1.2.2 Embedded ICPs . . . 16
1.2.2.1 Windows NT (Intel or Alpha) . . . 16
1.2.2.2 OpenVMS . . . 17
1.2.2.3 Digital UNIX . . . 17
1.3 Available Programming Interfaces . . . 18
1.3.1 Data Link Interface (DLI) . . . 18
1.3.2 Driver Interface . . . 19
1.3.3 Socket Interface (Freeway Server Only) . . . 19

2 Bit-Stream Protocol Theory of Operation 21

3 Typical Sequence of Operations 23

3.1 Initialization . . . 23
3.2 Attaching a Link . . . 25
3.3 Setting the ICP Buffer Size . . . 25
3.4 Configuring an Attached Link . . . 25
3.5 Binding an Attached Link . . . 26

3.6 Sending Data 26
3.7 Receiving Data 26
3.8 Unbinding a Link 27
3.9 Reading Reports 27
3.10 Detaching a Link 27
3.11 Termination 28

4 Commands and Responses 29

4.1 Access Category 30
4.1.1 Attach 30
4.1.2 Set ICP Buffer Size 30
4.1.3 Detach 30
4.2 Link Category 31
4.2.1 Configure Link 31
4.2.2 Bind 31
4.2.3 Unbind 33
4.3 Data Category 34
4.3.1 Send Data 34
4.3.2 Receive Acknowledgment 34
4.3.3 Receive Data 34
4.4 Report Category 35
4.4.1 Get Software Version 35
4.4.2 Get Statistics Report 35

5 Header Formats 37

5.1 Attach 42
5.2 Bind 43
5.3 Configure Link 44
5.4 Detach 45
5.5 Get Software Version 46
5.6 Get Statistics Report 47
5.7 Receive Data 48
5.8 Send Data 49
5.9 Set Buffer Size 50
5.10 Unbind 51

6 Error Conditions **53**

6.1 iICPStatus Field Codes. 53
6.2 iProtModifier Field Codes 54

A Include Files **57**

A.1 bsp_blks.h 58
A.2 bsp_defs.h 62
A.3 bsp_errs.h 65

Index **67**

List of Figures

Figure 1–1: Freeway Configuration 16
Figure 1–2: Embedded ICP Configuration 17
Figure 3–1: Typical Commands and Responses 24
Figure 4–1: Configure Link "C" Structure 31
Figure 4–2: Statistics Report "C" Structure 36
Figure 5–1: "C" Definition of Optional Arguments Structure (DLI Calls) 39
Figure 5–2: "C" Definition of ICP Packet Structure (Driver Calls) 40

List of Tables

Table 4–1: Command and Response Category Summary 29
Table 4–2: Link Configuration Options 32
Table 5–1: Command and Response Category Summary 38
Table 5–2: Comparison of Optional Arguments Usage (DLI versus Driver Calls) . . 41
Table 5–3: DLI_ICP_CMD_ATTACH Header Format 42
Table 5–4: DLI_PROT_SEND_BIND Header Format 43
Table 5–5: DLI_PROT_CFG_LINK Header Format 44
Table 5–6: DLI_ICP_CMD_DETACH Header Format 45
Table 5–7: DLI_PROT_GET_SOFTWARE_VER Header Format 46
Table 5–8: DLI_PROT_GET_STATISTICS Header Format 47
Table 5–9: DLI_PROT_RECV_DATA Header Format 48
Table 5–10: DLI_PROT_SEND_DATA Header Format 49
Table 5–11: DLI_PROT_SET_BUF_SIZE Header Format 50
Table 5–12: DLI_PROT_SEND_UNBIND Header Format 51
Table 6–1: Status Codes Reported in the iICPStatus Field 54
Table 6–2: Status Codes Reported in the iProtModifier Field 54
Table A–1: Include Files 57

Preface

Purpose of Document

This document describes the operation and programming interface required to use Simpact's Bit-Stream protocol software running on a Simpact intelligent communications processor (ICP).

Intended Audience

This document should be read by programmers who are interfacing an application program to the Bit-Stream protocol.

Organization of Document

Chapter 1 is an introduction to the Bit-Stream protocol and supported programming environments.

Chapter 2 summarizes the Bit-Stream protocol theory of operation.

Chapter 3 describes the Bit-Stream protocol typical sequence of operations.

Chapter 4 explains the Bit-Stream commands and responses.

Chapter 5 is the Bit-Stream protocol reference providing specific header formats for all Bit-Stream commands and responses.

Chapter 6 summarizes Bit-Stream error codes.

Appendix A shows the Bit-Stream include file contents.

Simpact References

The following documents provide useful supporting information, depending on the customer's particular hardware and software environments. Most documents are available on-line at Simpact's web site, www.simpact.com.

General Product Overviews

• *Freeway 1100 Technical Overview*	25-000-0419
• *Freeway 2000/4000/8800 Technical Overview*	25-000-0374
• *ICP2432 Technical Overview*	25-000-0420
• *ICP6000X Technical Overview*	25-000-0522

Hardware Support

• *Freeway 1100/1150 Hardware Installation Guide*	DC 900-1370
• *Freeway 1200 Hardware Installation Guide*	DC 900-1537
• *Freeway 1300 Hardware Installation Guide*	DC 900-1539
• *Freeway 2000/4000 Hardware Installation Guide*	DC 900-1331
• *Freeway 8800 Hardware Installation Guide*	DC 900-1553
• *Freeway ICP6000R/ICP6000X Hardware Description*	DC 900-1020
• *ICP6000(X)/ICP9000(X) Hardware Description and Theory of Operation*	DC 900-0408
• *ICP2424 Hardware Description and Theory of Operation*	DC 900-1328
• *ICP2432 Hardware Description and Theory of Operation*	DC 900-1501
• *ICP2432 Hardware Installation Guide*	DC 900-1502

Freeway Software Installation Support

• *Freeway Software Release Addendum: Client Platforms*	DC 900-1555
• *Freeway User's Guide*	DC 900-1333
• *Getting Started with Freeway 1100/1150*	DC 900-1369
• *Getting Started with Freeway 1200*	DC 900-1536
• *Getting Started with Freeway 1300*	DC 900-1538
• *Getting Started with Freeway 2000/4000*	DC 900-1330
• *Getting Started with Freeway 8800*	DC 900-1552
• *Loopback Test Procedures*	DC 900-1533

Embedded ICP Installation and Programming Support

- *ICP2432 User's Guide for Digital UNIX* DC 900-1513
- *ICP2432 User's Guide for OpenVMS Alpha* DC 900-1511
- *ICP2432 User's Guide for OpenVMS Alpha (DLITE Interface)* DC 900-1516
- *ICP2432 User's Guide for Windows NT* DC 900-1510
- *ICP2432 User's Guide for Windows NT (DLITE Interface)* DC 900-1514

Application Program Interface (API) Programming Support

- *Freeway Data Link Interface Reference Guide* DC 900-1385
- *Freeway Transport Subsystem Interface Reference Guide* DC 900-1386
- *QIO/SQIO API Reference Guide* DC 900-1355

Socket Interface Programming Support

- *Freeway Client-Server Interface Control Document* DC 900-1303

Toolkit Programming Support

- *Freeway Server-Resident Application and Server Toolkit Programmer's Guide* DC 900-1325
- *OS/Impact Programmer's Guide* DC 900-1030
- *Protocol Software Toolkit Programmer's Guide* DC 900-1338

Protocol Support

- *ADCCP NRM Programmer's Guide* DC 900-1317
- *Asynchronous Wire Service (AWS) Programmer's Guide* DC 900-1324
- *Addendum: Embedded ICP2432 AWS Programmer's Guide* DC 900-1557
- *AUTODIN Programmer's Guide* DC 908-1558
- *BSC Programmer's Guide* DC 900-1340
- *BSCDEMO User's Guide* DC 900-1349
- *BSCTRAN Programmer's Guide* DC 900-1406
- *DDCMP Programmer's Guide* DC 900-1343
- *FMP Programmer's Guide* DC 900-1339
- *Military/Government Protocols Programmer's Guide* DC 900-1602
- *SIO STD-1200A (Rev. 1) Programmer's Guide* DC 908-1359

- *SIO STD-1300 Programmer's Guide* DC 908-1559
- *X.25 Call Service API Guide* DC 900-1392
- *X.25/HDLC Configuration Guide* DC 900-1345
- *X.25 Low-Level Interface* DC 900-1307

Document Conventions

Program code samples are written in the "C" programming language.

Physical "ports" on the ICPs are logically referred to as "links." However, since port and link numbers are usually identical (that is, port 0 is the same as link 0), this document uses the term "link."

Revision History

The revision history of the *Bit-Stream Protocol Programmer's Guide*, Simpact document DC 900-1574A, is recorded below:

Revision	Release Date	Description
DC 900-1574A	April 1999	Original Release

Customer Support

If you are having trouble with any Simpact product, call us at 1-800-275-3889 Monday through Friday between 8 a.m. and 5 p.m. Pacific time.

You can also fax your questions to us at (619)560-2838 or (619)560-2837 any time. Please include a cover sheet addressed to "Customer Service."

We are always interested in suggestions for improving our products. You can use the report form in the back of this manual to send us your recommendations.

Chapter

1 Introduction

Simpact provides a variety of user-programmable, wide-area network (WAN) connectivity solutions for real-time financial, defense, telecommunications, and process-control applications. Simpact software includes a wide variety of legacy and specialty protocols that run on Simpact's intelligent communication processor (ICP) boards. The protocol software is independent of the hardware and operating system environment.

1.1 Protocol Overview

The Bit-Stream protocol is used to convey synchronous bit streams, with no modification or validation, between client applications and ICP serial links. Refer to Chapter 2 for the Bit-Stream protocol Theory of Operation.

1.2 Supported Hardware

Section 1.2.1 and Section 1.2.2 briefly describe the currently supported hardware and operating system environments. Refer to the *Simpact References* on page 12 for the applicable hardware installation documentation and user's guide for your particular hardware environment.

1.2.1 Freeway Server

The Freeway server is a stand-alone box with pre-installed ICPs. It provides multiple data links and a variety of network services to LAN-based clients. Figure 1–1 shows a Freeway server configuration where the client application communicates with the

Freeway ICPs using Simpact's data link interface (DLI). Client applications can use either the DLI or socket interface, as described in Section 1.3.1 and Section 1.3.3. A variety of client operating systems are supported (UNIX, VMS, and Windows NT).

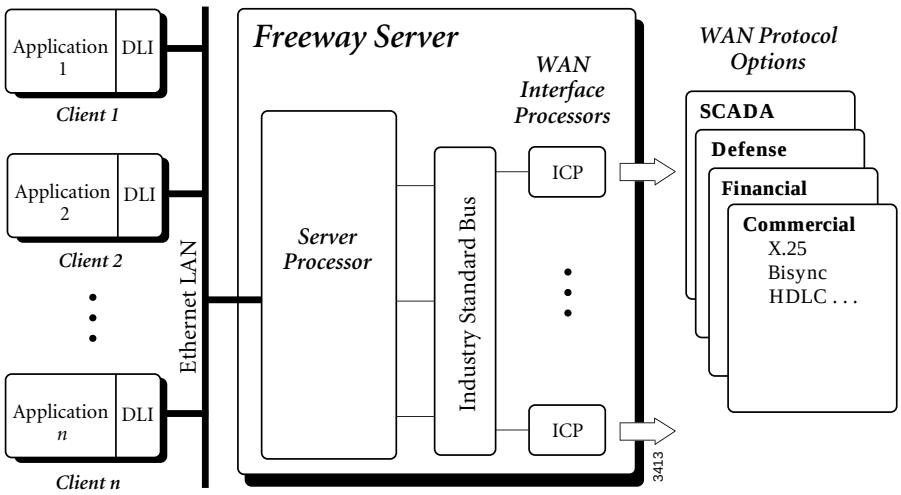

Figure 1–1: Freeway Configuration

1.2.2 Embedded ICPs

An ICP installed in a user-provided computer is called an "embedded" ICP. Figure 1–2 shows one possible embedded ICP environment where the client application uses Simpact's DLI programming interface. The currently supported operating systems and their respective supported programming interfaces are described in the following subsections.

1.2.2.1 Windows NT (Intel or Alpha)

The two supported programming interfaces for an embedded ICP in a Windows NT client computer are the DLI (using the embedded DLITE interface) and the driver interface, described in Section 1.3.1 and Section 1.3.2. The primary references are the

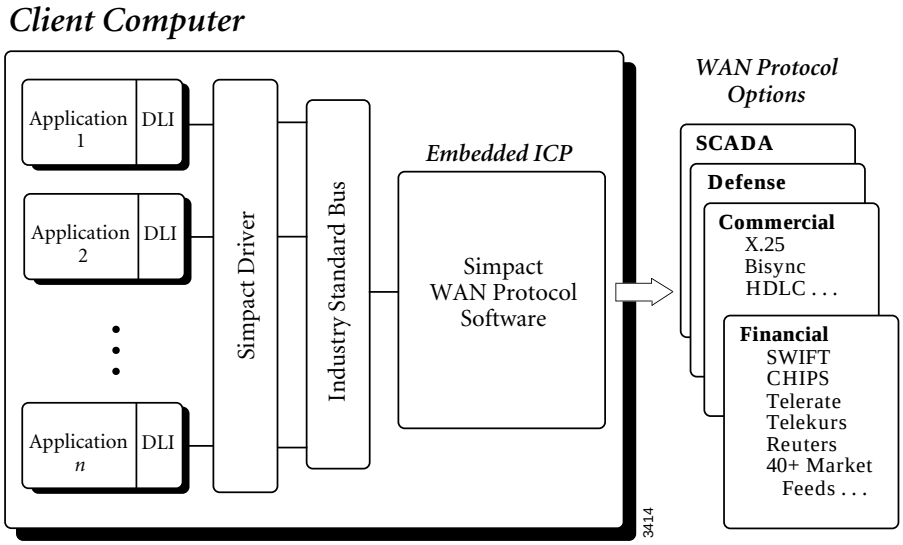

Figure 1–2: Embedded ICP Configuration

Freeway Data Link Interface Reference Guide and the user's guide for the applicable ICP (for example, the *ICP2432 User's Guide for Windows NT*).

1.2.2.2 OpenVMS

The supported programming interfaces for an embedded ICP in an OpenVMS client computer are the DLI (using the embedded DLITE interface) and the driver interface, described in Section 1.3.1 and Section 1.3.2. The primary reference is the user's guide for the applicable ICP (for example, the *ICP2432 User's Guide for OpenVMS Alpha*).

1.2.2.3 Digital UNIX

The supported programming interface for an embedded ICP in a Digital UNIX client computer is the driver interface, described in Section 1.3.2. The primary reference is the user's guide for the applicable ICP (for example, the *ICP2432 User's Guide for Digital UNIX*).

1.3 Available Programming Interfaces

The following sections briefly describe the currently supported application programming interfaces that can be used in conjunction with this *Bit-Stream Protocol Programmer's Guide*. Choice of programming interface is dependent upon your hardware environment, as described in the previous Section 1.2.

1.3.1 Data Link Interface (DLI)

Simpact's data link interface provides a high-level, session-oriented application programming interface for a variety of client hardware and operating system environments. below. The details of the DLI are described in the *Freeway Data Link Interface Reference Guide*.

In an embedded ICP environment, DLI support is provided by the embedded DLITE interface. Also refer to the *ICP2432 User's Guide* for your specific operating system regarding the minor differences for the DLITE interface.

A brief description of the DLI concepts that apply to the Bit-Stream protocol are:

- The DLI requires that the application first invoke the `dlInit` function to initialize the DLI, then call the `dlOpen` function to open an I/O path to the ICP.
- *Raw* operation — The Bit-Stream protocol uses DLI *Raw* operation, which means that the client application must employ the DLI optional arguments data structure (shown in Figure 5–1 on page 39) to issue `dlRead` and `dlWrite` commands to a session.
- DLI configuration parameters — The DLI configuration file needs to include only those session parameters whose values differ from the defaults. To use DLI *Raw* operation, you must specify the following parameter:

```
protocol = "raw"
```

- An example Bit-Stream test program and supporting DLI and TSI configuration files are provided with the Bit-Stream product, as listed below. To run the loopback test program, refer to the *Loopback Test Procedures* document (for the Freeway server) or the appropriate user's guide for your embedded ICP and operating system (for example, the *ICP2432 User's Guide for Windows NT*).

Non-blocking I/O Test Program	DLITE Configuration File Name (Embedded ICP)	DLI Configuration File Name (Freeway Server)	TSI Configuration File Name (Freeway Server)
`bspalp.c`	`bspembcfg`	`bspaldcfg`	`bspaltcfg`

1.3.2 Driver Interface

The driver interface requires the client application to make low-level calls directly to the ICP's device driver. The driver interface is used when the DLI is not available for a particular environment, or when there is a programming requirement to bypass the DLI. To perform I/O functions, refer to the driver interface document (the typical document naming convention is *ICP2432 User's Guide for <Specific Operating System>)*. Then refer to Chapter 5 of this *Bit-Stream Protocol Programmer's Guide* for the protocol-specific header formats. The header formats are very similar between the DLI and driver interfaces.

1.3.3 Socket Interface (Freeway Server Only)

The socket interface is available only for the Freeway server environment. Client applications can submit I/O requests directly to the BSD socket or, alternatively, the application can interface to Simpact's transport subsystem interface (TSI) layer, which allows the TSI to perform some of the header management. The primary reference is the *Freeway Client-Server Interface Control Document*, which provides complete header format descriptions.

Chapter

2 Bit-Stream Protocol Theory of Operation

The Bit-Stream protocol conveys one or more synchronous bit streams of data, with no modification or validation, between client applications and ICP serial links (ports). This process is bidirectional and full-duplex. One bit stream per serial link is supported.

Links are prepared for activity by an optional Configure Link client command message to assign their variable characteristics. Link I/O is activated by an Enable Link client command message. Link I/O is deactivated by a Disable Link client command message.

The application has control of several configuration features for the links individually and for the ICP as a whole. Individual links can be independently configured for the following features:

- Data rate (default 9600 baud)
- External vs. internal clocking source (default external)
- Electrical interface type, for ICPs which can control this from software (the default is EIA-232)
- Standard vs. inverted bit-encoding (NRZ vs. NRZB; the default is NRZ)
- 8-bit transmit idle pattern for time-fill (the default is all ones)
- Data direction (receive-only, transmit-only, or both directions)

The ICP as a whole can be configured for its link transmit/reception buffer size, by the Configure Buffer Size client command message. The buffers form a pool available for all

links. They are also used for all messages exchanged at the interface to the client. Data is DMA'ed for all link I/O, using the configured data rate, clocking source, electrical interface (when software-controllable), bit-encoding, transmit-idle bit pattern, and data direction(s).

Buffers supplied by the client for link transmission in Output Data client command messages are limited in size by the configured ICP buffer size. These buffers are conveyed as-is to the link for transmission. The buffered data bytes are sent with the bit-order of client-msb-first. Each completed transmission is acknowledged to the client, by Transmit Acknowledgment ICP response messages. When no transmit buffers are available from the client, the link transmits the configured 8-bit time-fill idle pattern. The client application is entirely responsible for creating in its data buffers all the structure and content of the bit stream that the link's far end may demand (apart from the transmit idle pattern).

The bit order of link reception is also client-msb-first. Received data is buffered fully into the ICP buffers, then conveyed as-is to the client by Input Data ICP response messages. There are no synchronization bytes, start and end bit-groups, framing flags, block check bytes, external interrupts, etc. used in handling the link's bit stream, to delimit messages from time-fill. The entire far-end transmission is conveyed to the client. The client application is entirely responsible for identifying in the data buffers all the structure and content of the bit stream that it expects from the link's far end.

Efficiency and evenness of data flow both ways is provided by the ability to specify a link's data rate and the ICP's buffer size. In addition, transmission flow control is aided by the transmit acknowledgments.

Statistics of transmitted and received buffers are kept separately for each link. They are sent to the client in a Statistics Report ICP response message, when elicited by a Request Statistics client command message.

Chapter

3 Typical Sequence of Operations

3.1 Initialization

Section 1.3 on page 18 summarizes the programming interfaces available for use with the ICP. In all cases, the application first must perform any initialization required by the selected programming interface, and then must open an I/O path to the ICP prior to writing commands to or reading responses from the protocol software on the ICP.

After the application establishes an I/O path to the ICP, it writes commands to the ICP, and reads responses from the ICP. See Chapter 4 and Chapter 5 for additional details regarding the command and response formats. This chapter describes the typical sequence to execute selected specific operations. The sections that follow are presented in the order in which they are generally used.

Figure 3–1 is a diagram of the commands and responses described in Section 3.2 through Section 3.10.

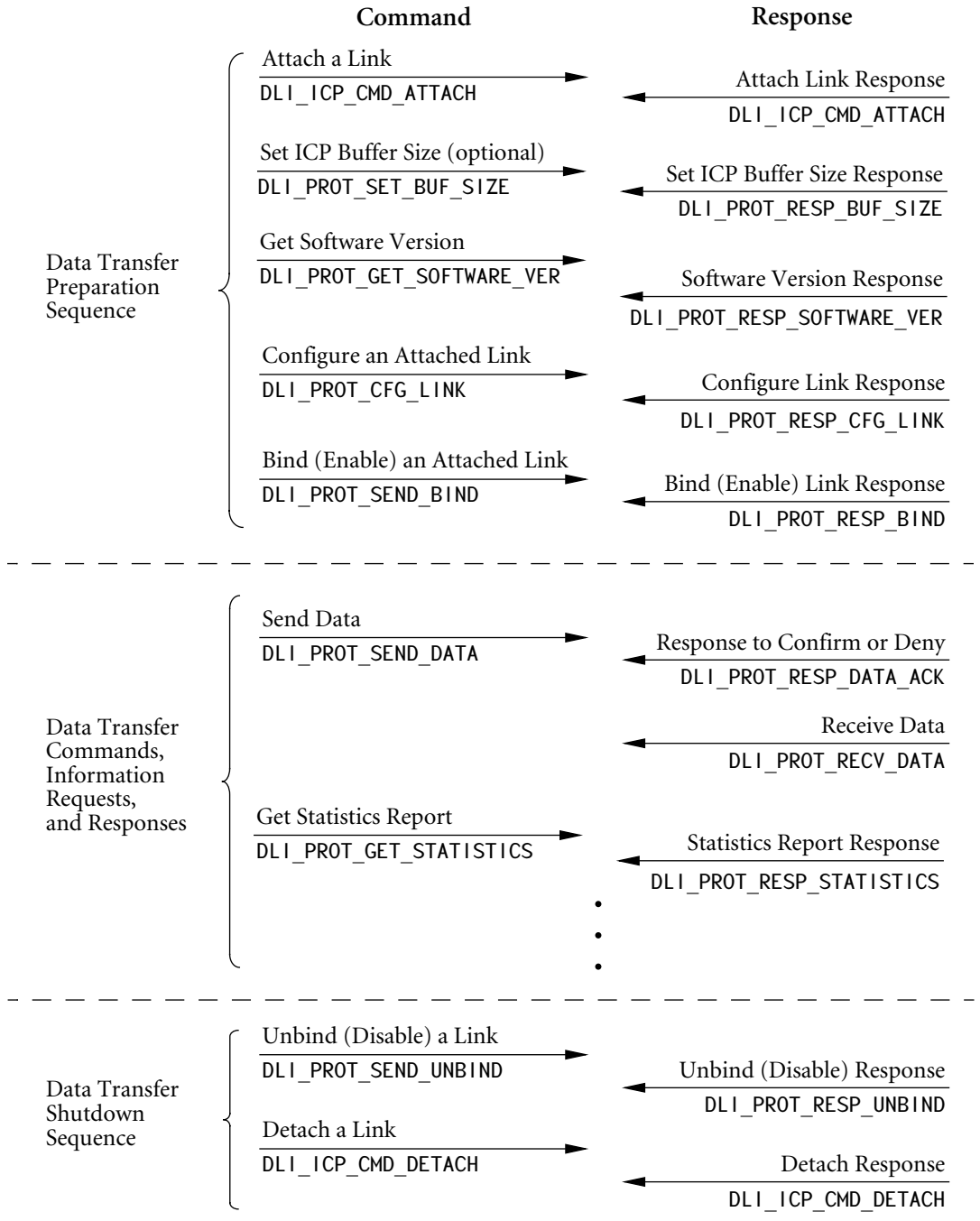

Figure 3–1: Typical Commands and Responses

3.2 Attaching a Link

The application must successfully attach an ICP link before it can configure, bind, read from, or write to that link. To attach a link, the application must perform the actions listed below.

- The application writes a DLI_ICP_CMD_ATTACH command. This allocates the ICP link for application use and associates a session with that link.
- The application reads a DLI_ICP_CMD_ATTACH response with a status field that confirms or denies the request. The returned session identifier is used on all subsequent DLI_ICP_CMD_WRITE requests.

3.3 Setting the ICP Buffer Size

After the application attaches an ICP link, it can optionally configure the ICP buffer size. If it is used, this command must be among the first four DLI_PROT commands, and must precede the first DLI_PROT_SEND_BIND command.

- The application writes an optional DLI_PROT_SET_BUF_SIZE command. This sets the size of the ICP buffers used by all ICP links.
- The application reads a DLI_PROT_RESP_BUF_SIZE response with a status field that confirms or denies the request. A 16-bit data area with the response reports the number of ICP buffers configured at the requested size.

3.4 Configuring an Attached Link

After the application attaches an ICP link, it can configure the link prior to binding the link. If a competing application currently has the link bound, the link cannot be configured. If the default configuration for the link is suitable, configuration is not required. To configure a link, the application must perform the actions listed below.

- The application writes a DLI_PROT_CFG_LINK command.

- The application reads a `DLI_PROT_RESP_CFG_LINK` response with a status field that confirms or denies the request.

3.5 Binding an Attached Link

After the application attaches an ICP link, it can bind (enable) the link, which places it online for data transfer. To bind a link, the application must perform the actions listed below.

- The application writes a `DLI_PROT_SEND_BIND` command.
- The application reads a `DLI_PROT_RESP_BIND` response with a status field that confirms or denies the request.

3.6 Sending Data

After the application attaches and binds the ICP link, it can send data on the ICP link. To write a message, the application must perform the actions listed below.

- The application writes a `DLI_PROT_SEND_DATA` command.
- The application reads a `DLI_PROT_RESP_DATA_ACK` response with a status field that confirms or denies the request.

3.7 Receiving Data

After the application attaches and binds the ICP link, it can receive data on the ICP link. To read a message, the application must perform the action listed below.

- The application reads a `DLI_PROT_RECV_DATA` response.

3.8 Unbinding a Link

After the application attaches and binds the ICP link, it can unbind (disable) the link. Unbinding a link places it offline to the application. To unbind a link, the application must perform the actions listed below.

- The application writes a `DLI_PROT_SEND_UNBIND` command.
- The application reads a `DLI_PROT_RESP_UNBIND` response with a status field that confirms or denies the request.

3.9 Reading Reports

After the application attaches an ICP link, it can read reports. The application can also read reports after binding the link. Several types of reports are available. To read a specific report, the application must perform the actions listed below.

- The application writes a command requesting the desired report. Supported report requests are:

  `DLI_PROT_GET_SOFTWARE_VER`

  `DLI_PROT_GET_STATISTICS`

- The application reads the report response of the following types:

  `DLI_PROT_RESP_SOFTWARE_VER`

  `DLI_PROT_RESP_STATISTICS`

3.10 Detaching a Link

After the application attaches an ICP link, it can detach the link. Detaching a link relinquishes application access to the link. If an application binds the ICP link, and detaches the link without first unbinding, the ICP quietly handles the implied unbind, then fin-

ishes detaching the application. To detach a link, the application must perform the actions listed below.

- The application writes a `DLI_ICP_CMD_DETACH` command.
- The application reads a `DLI_ICP_CMD_DETACH` response with a status field that confirms or denies the request.

3.11 Termination

Section 1.3 on page 18 summarizes the programming interfaces available for use with the ICP. In all cases the application should be designed to perform the termination sequence recommended by the selected programming interface prior to terminating application execution. An orderly shutdown may be required to successfully close the I/O path to the ICP, release allocated system resources, and complete the capture of trace or log information.

Chapter

4 Commands and Responses

This chapter describes the commands written to the ICP and the responses received from the ICP. After you are familiar with the functionality, refer to Chapter 5 for detailed header formats to aid in writing application programs to interface to the Bit-Stream protocol.

Table 4–1 summarizes the command and response categories, which are explained further in the remainder of this chapter.

Table 4–1: Command and Response Category Summary

Category	Command Code	Response Code	Functional Description Reference Section
Access (Section 4.1)	DLI_ICP_CMD_ATTACH	DLI_ICP_CMD_ATTACH	Section 4.1.1 on page 30
	DLI_PROT_SET_BUF_SIZE	DLI_PROT_RESP_BUF_SIZE	Section 4.1.2 on page 30
	DLI_ICP_CMD_DETACH	DLI_ICP_CMD_DETACH	Section 4.1.3 on page 30
Link (Section 4.2)	DLI_PROT_CFG_LINK	DLI_PROT_RESP_CFG_LINK	Section 4.2.1 on page 31
	DLI_PROT_SEND_BIND	DLI_PROT_RESP_BIND	Section 4.2.2 on page 31
	DLI_PROT_SEND_UNBIND	DLI_PROT_RESP_UNBIND	Section 4.2.3 on page 33
Data (Section 4.3)	DLI_PROT_SEND_DATA	DLI_PROT_RESP_DATA_ACK DLI_PROT_RECV_DATA	Section 4.3.1 on page 34 Section 4.3.2 on page 34
Report (Section 4.4)	DLI_PROT_GET_SOFTWARE_VER	DLI_PROT_RESP_SOFTWARE_VER	Section 4.4.1 on page 35
	DLI_PROT_GET_STATISTICS	DLI_PROT_RESP_STATISTICS	Section 4.4.2 on page 35

4.1 Access Category

The commands and responses in the Access category are used to establish access sessions for a specified ICP link. Only one application session is permitted per ICP link.

4.1.1 Attach

The DLI_ICP_CMD_ATTACH command obtains access to a specified ICP data link. The corresponding DLI_ICP_CMD_ATTACH response reports the success or failure of the request. If successful, the access session ID assigned to the application is reported. The header formats are detailed in Section 5.1 on page 42.

4.1.2 Set ICP Buffer Size

The optional DLI_PROT_SET_BUF_SIZE command sets the size of the ICP message buffers used by all ICP links. If the Set ICP Buffer Size command is used, it must be among the first four DLI_PROT commands after the ICP Bit-Stream software is downloaded, and must precede the first DLI_PROT_SEND_BIND command. If no DLI_PROT_SET_BUF_SIZE command is sent to the ICP, the default size of 128 bytes is used. The valid range is 16 to 8192 bytes. Sizes larger than 2048 bytes require the default server TSI configuration file (if applicable) to be modified for permitting larger buffers. The corresponding DLI_PROT_RESP_BUF_SIZE response reports the success or failure of the request. The header formats are detailed in Section 5.9 on page 50. The 16-bit data area with the response contains the number of ICP buffers configured at that size.

4.1.3 Detach

The DLI_ICP_CMD_DETACH command relinquishes access to the ICP link associated with a specified application session. The corresponding DLI_ICP_CMD_DETACH response reports the success or failure of the request. The header formats are detailed in Section 5.4 on page 45.

4.2 Link Category

The commands and responses in the Link category are used to enable, disable or configure a specified link.

4.2.1 Configure Link

The DLI_PROT_CFG_LINK command contains configuration for the specified ICP data link. The link must not be enabled. The corresponding DLI_PROT_RESP_CFG_LINK response reports the success or failure of the request. The header formats are detailed in Section 5.3 on page 44. Figure 4–1 defines the "C" structure used for the data area of both the DLI_PROT_CFG_LINK command and the DLI_PROT_RESP_CFG_LINK response.

```
typedef struct conf_type        /* 9 bytes */
{
    unsigned int   rate;        /* baud rate                    */
    unsigned char  clock;       /* 0 = external, 1 = internal   */
    unsigned char  electrical;  /* electrical interface         */
    unsigned char  encoding;    /* 0 = NRZ, 1 = NRZB            */
    unsigned char  xmtidle;     /* transmit idle pattern        */
    unsigned char  datadir;     /* 0 = rcv & xmt, 1 = rcv only */
                                /*                2 = xmt only */
} CONF_TYPE;
#define CONF_SIZE  sizeof(CONF_TYPE)
```

Figure 4–1: Configure Link "C" Structure

Table 4–2 lists the link configuration options and their valid values. If the Configure Link command is not sent, the configuration options default to the values shown in the "Default" column of Table 4–2. If the Configure Link command is sent, each option field of the structure defined in Figure 4–1 must contain a valid value.

4.2.2 Bind

The DLI_PROT_SEND_BIND command enables an ICP link to which the application has obtained access. The link must not be already enabled. The corresponding DLI_PROT_RESP_BIND response reports the success or failure of the request. The

Table 4–2: Link Configuration Options

Option	Default (✓)	Optional Values	Comments
Data rate	✓	0 = 9600 1 = 300 2 = 600 3 = 1200 4 = 2400 5 = 4800 6 = 9600 7 = 19200 8 = 38400 9 = 56000 10 = 57600 11 = 64000 12 = 76800 13 = 115200 14 = 122880 15 = 153600 16 = 184320 17 = 230400 18 = 307200 19 = 368640 20 = 460800 21 = 614400 22 = 737280 23 = 921600 24 = 1228800	Values 0-24 are indexes into list of baud rates Values 50 and higher are true baud rates Values 25-49 are invalid
Clock source	✓	0 = external 1 = internal	
Bit encoding	✓	0 = NRZ 1 = NRZB	NRZB is NRZ with inverted bit sense
Transmit idle	all ones		Any 8-bit pattern is valid
Data direction	✓	0 = both ways 1 = receive only 2 = transmit only	

Table 4–2: Link Configuration Options *(Cont'd)*

Option	Default (✓)	Optional Values	Comments
Electrical interface	✓	0 = EIA-232 1 = MIL-188C 2 = EIA-232 3 = EIA-562 4 = EIA-422 5 = EIA-485 6 = MIL-188-114B 7 = invalid 8 = EIA-423 9 = MIL-188-114U 10 = invalid 11 = invalid 12 = EIA-449 13 = EIA-530 14 = V.35 15 = invalid	Above 15 is invalid. Must be 0 if ICP does not program this data

`iProtModifier` field of the command specifies the number of client-supplied transmit buffers to be accumulated by the link before first enabling transmission (a value of 0 defaults to one transmit buffer). The header formats are detailed in Section 5.2 on page 43. This command also clears the link's statistics counters.

4.2.3 Unbind

The `DLI_PROT_SEND_UNBIND` command disables an ICP link that the application previously enabled. The corresponding `DLI_PROT_RESP_UNBIND` response reports the success or failure of the request. The header formats are detailed in Section 5.10 on page 51.

4.3 Data Category

The commands and responses in the Data category are used to transfer data on the specified ICP link.

4.3.1 Send Data

The application writes the `DLI_PROT_SEND_DATA` command to the ICP to transmit a message through the WAN interface. The header formats are detailed in Section 5.8 on page 49.

The data is provided as an array of bytes to be transmitted from lowest byte address to highest byte address. All eight bits of each byte are used. The most significant bit of a byte is transmitted first.

4.3.2 Receive Acknowledgment

The application receives a `DLI_PROT_RESP_DATA_ACK` response after a `DLI_PROT_SEND_DATA` command successfully goes out on the link. The header formats are detailed in Section 5.8 on page 49.

4.3.3 Receive Data

The application reads a `DLI_PROT_RECV_DATA` response from the ICP to receive a message through the WAN interface. The header formats are detailed in Section 5.7 on page 48.

4.4 Report Category

The Report category is used to request specific reports from the ICP.

4.4.1 Get Software Version

The DLI_PROT_GET_SOFTWARE_VER command requests software version information from the protocol software on the ICP. The ICP DLI_PROT_RESP_SOFTWARE_VER response contains the requested version information. The header formats are detailed in Section 5.5 on page 46.

When the application writes a DLI_PROT_GET_SOFTWARE_VER command to the ICP, it does so with a zero-length data area. When the ICP sends the corresponding DLI_PROT_RESP_SOFTWARE_VER response, the iProtModifier field of the successful response contains a positive value (instead of the normal zero) which is the number of ports on the ICP. The data area contains an ASCII text string with embedded new-line characters ('\n') and is terminated by a null ('\0') character.

4.4.2 Get Statistics Report

The DLI_PROT_GET_STATISTICS command requests statistics for a specified ICP link. The ICP DLI_PROT_RESP_STATISTICS response contains the requested statistics. The header formats are detailed in Section 5.6 on page 47.

When the application writes a DLI_PROT_GET_STATISTICS command to the ICP, it does so with a zero-length data area. The ICP DLI_PROT_RESP_STATISTICS response data area contains statistics for the specified ICP link using the "C" structure format shown in Figure 4–2, where "UINT32" is a 32-bit unsigned integer data type.

```
typedef struct stats_type
{
    unsigned int  frame_rcvd;   /* # of frames received       */
    unsigned int  frame_sent;   /* # of frames transmitted    */
    unsigned int  rcv_ovrrun;   /* # of receiver overruns     */
    unsigned int  xmt_undrun;   /* # of transmitter underruns */
} STATS_TYPE;
#define STATS_SIZE  sizeof(STATS_TYPE)
#define NUM_STATS   (STATS_SIZE /sizeof(UINT32))
```

Figure 4–2: Statistics Report "C" Structure

Chapter

5 Header Formats

The application and the protocol software on the ICP communicate by sending commands and responses. This chapter identifies the format of each command and response. Table 5–1 on page 38 gives an overview of the categories of commands and responses used by the protocol software. The individual header formats are presented alphabetically by command/response name in Section 5.1 through Section 5.10.

Figure 5–1 gives the "C" definition of the optional arguments structure (`DLI_OPT_ARGS`) used when making calls to the DLI interface. Figure 5–2 shows a typical definition of the comparable structures used when making calls directly to a driver interface. Table 5–2 compares the fields used to interface to the DLI versus a driver and also gives a general overview of the typical field values, though there are some request-specific differences noted in the following sections. Refer to the *Freeway Client-Server Interface Control Document* for using the socket interface.

Note

Most of the fields within the `ICP_PACKET` structure are in network byte-order. However, each field name that starts with "`usProt`" or "`iProt`" is in client byte-order. The client must declare its byte-order characteristics in the `iICPStatus` field of each command when making non-DLI calls.

Table 5–1: Command and Response Category Summary

Category	Command Code	Response Code	Header Format Reference Section
Access	DLI_ICP_CMD_ATTACH	DLI_ICP_CMD_ATTACH	Section 5.1 on page 42
	DLI_PROT_SET_BUF_SIZE	DLI_PROT_RESP_BUF_SIZE	Section 5.9 on page 50
	DLI_ICP_CMD_DETACH	DLI_ICP_CMD_DETACH	Section 5.4 on page 45
Link	DLI_PROT_CFG_LINK	DLI_PROT_RESP_CFG_LINK	Section 5.3 on page 44
	DLI_PROT_SEND_BIND	DLI_PROT_RESP_BIND	Section 5.2 on page 43
	DLI_PROT_SEND_UNBIND	DLI_PROT_RESP_UNBIND	Section 5.10 on page 51
Data	DLI_PROT_SEND_DATA	DLI_PROT_RESP_DATA_ACK DLI_PROT_RECV_DATA	Section 5.8 on page 49 Section 5.7 on page 48
Report	DLI_PROT_GET_SOFTWARE_VER	DLI_PROT_RESP_SOFTWARE_VER	Section 5.5 on page 46
	DLI_PROT_GET_STATISTICS	DLI_PROT_RESP_STATISTICS	Section 5.6 on page 47

```
typedef struct      _DLI_OPT_ARGS
{
     unsigned short  usFWPacketType;    /* Server's packet type      */
     unsigned short  usFWCommand;       /* Server's cmd sent or rcvd */
     unsigned short  usFWStatus;        /* Server's status of I/O ops*/
     unsigned short  usICPClientID;     /* old su_id - not used      */
     unsigned short  usICPServerID;     /* old sp_id - not used      */
     unsigned short  usICPCommand;      /* ICP's command.            */
     short           iICPStatus;        /* ICP's command status      */
     unsigned short  usICPParms[3];     /* ICP's extra parameters    */
     unsigned short  usProtCommand;     /* protocol command          */
     short           iProtModifier;     /* protocol cmd's modifier   */
     unsigned short  usProtLinkID;      /* protocol link ID          */
     unsigned short  usProtCircuitID;   /* protocol circuit ID       */
     unsigned short  usProtSessionID;   /* protocol session ID       */
     unsigned short  usProtSequence;    /* protocol sequence         */
     unsigned short  usProtXParms[1];   /* protocol extra parameters */
}    DLI_OPT_ARGS;
typedef DLI_OPT_ARGS *PDLI_OPT_ARGS;
#define DLI_OPT_ARGS_SIZE sizeof(DLI_OPT_ARGS)
```

Figure 5–1: "C" Definition of Optional Arguments Structure (DLI Calls)

```
typedef struct       _ICP_PACKET
{
    ICP_HDR          icp_hdr;
    PROT_HDR         prot_hdr;
    DATA             data;
}   ICP_PACKET;

typedef struct      _ICP_HDR
{
   unsigned short   usICPClientID;    /* Old su_id                       */
   unsigned short   usICPServerID;    /* Old sp_id                       */
   unsigned short   usICPCount;       /* Size of PROT_HDR plus data      */
                                      /*   (Used only in non-DLI calls)  */
   unsigned short   usICPCommand;     /* ICP's command                   */
            short   iICPStatus;       /* ICP's command status            */
   unsigned short   usICPParms[3];    /* ICP's extra parameters          */
} ICP_HDR;

typedef struct      _PROT_HDR
{
   unsigned short   usProtCommand;    /* Protocol command                */
            short   iProtModifier;    /* Protocol command's modifier     */
   unsigned short   usProtLinkID;     /* Protocol link ID                */
   unsigned short   usProtCircuitID;  /* Protocol circuit ID             */
   unsigned short   usProtSessionID;  /* Protocol session ID             */
   unsigned short   usProtSequence;   /* Protocol sequence               */
   unsigned short   usProtXParms[2];  /* Protocol extra parameters       */
} PROT_HDR;
typedef union  _DATA
{
    unsigned int     lwords[1];  /* data as 32-bit units */
    unsigned short   words [2];  /* data as 16-bit units */
    unsigned char    bytes [4];  /* data as  8-bit units */
} DATA;
```

Figure 5–2: "C" Definition of ICP Packet Structure (Driver Calls)

Note

The `usICPCount` field in the ICP header is unique to non-DLI calls. It is not used in DLI calls.

Table 5–2: Comparison of Optional Arguments Usage (DLI versus Driver Calls)

DLI_OPT_ARGS Field Name (DLI Calls)	ICP_PACKET Field Name (Driver Calls)	Typical Value
Server Header Fields (network byte-order)		
usFWPacketType	Omitted	FW_DATA
usFWCommand	Omitted	FW_ICP_WRITE FW_ICP_READ
usFWStatus	Omitted	0
ICP Header Fields[a]		
usICPClientID	icp_hdr.usICPClientID	0
usICPServerID	icp_hdr.usICPServerID	0
Omitted	icp_hdr.usICPCount	Size of PROT_HDR plus data (non-DLI calls only)
usICPCommand	icp_hdr.usICPCommand	During session establishment: DLI_ICP_CMD_ATTACH DLI_ICP_CMD_DETACH During I/O operations: DLI_ICP_CMD_READ DLI_ICP_CMD_WRITE
iICPStatus[b]	icp_hdr.iICPStatus	Write: Client memory order: 0x0000 = Big Endian (*e.g.*, SUN) 0x4000 = Little Endian (*e.g.*, DEC) Read: Command status code: DLI_ICP_CMD_STATUS_OK or error code (Table 6–1 on page 54)
usICPParms[0]	icp_hdr.usICPParms[0]	ICP return node from DLI_ICP_CMD_ATTACH
usICPParms[1]	icp_hdr.usICPParms[1]	0
usICPParms[2]	icp_hdr.usICPParms[2]	0
Protocol Header Fields[a]		
usProtCommand	prot_hdr.usProtCommand	Request dependent (see Table 5–1 on page 38)
iProtModifier	prot_hdr.iProtModifier	Success or error code (Table 6–2 on page 54)
usProtLinkID	prot_hdr.usProtLinkID	ICP link number
usProtCircuitID	prot_hdr.usProtCircuitID	0
usProtSessionID	prot_hdr.usProtSessionID	Protocol Session ID from DLI_ICP_CMD_ATTACH response
usProtSequence	prot_hdr.usProtSequence	0
usProtXParms[0]	prot_hdr.usProtXParms[0]	0
usProtXParms[1]	prot_hdr.usProtXParms[1]	0
Data: DLI calls – data is buffered separately. Non-DLI calls – data array immediately follows headers.		

[a] For non-DLI calls, ICP Header fields must be network byte-order, and Protocol Header fields must be client byte order.
[b] The iICPStatus field must specify client memory order for non-DLI calls only.

5.1 Attach

Table 5–3 shows the DLI_ICP_CMD_ATTACH header format. See Section 4.1.1 on page 30 for a functional description. There is no associated data area.

Table 5–3: DLI_ICP_CMD_ATTACH Header Format

Field	Command Value (Write)	Response Value (Read)
Freeway Header (network byte-order; used in DLI calls only):		
usFWPacketType	= FW_DATA	= FW_DATA
usFWCommand	= FW_ICP_WRITE	= FW_ICP_READ
usFWStatus	= 0	= 0
ICP Header[a]		
usICPClientID	= 0	= 0
usICPServerID	= 0	= 0
[a]usICPCount	= 16 (non-DLI calls only)	= 16 (non-DLI calls only)
usICPCommand	= DLI_ICP_CMD_ATTACH	= DLI_ICP_CMD_ATTACH
[a]iICPStatus	= Client memory order (Table 5–2 on page 41)	= Status code (Table 6–1 on page 54)
usICPParms[0]	= ICP return node[b]	= ICP return node
usICPParms[1]	= 0	= 0
usICPParms[2]	= 0	= 0
Protocol Header[a]		
usProtCommand	= DLI_ICP_CMD_ATTACH	= DLI_ICP_CMD_ATTACH
iProtModifier	= 0	= Status code (Table 6–2 on page 54)
usProtLinkID	= Link number	= Link number
usProtCircuitID	= 0	= 0
usProtSessionID	= 0	= Session ID assigned by ICP[c]
usProtSequence	= 0	= 0
usProtXParms[0]	= 0	= 0
usProtXParms[1]	= 0	= 0

[a] Requirements for non-DLI calls: ICP Header fields must be in network byte-order, Protocol Header fields must be in client byte order, and the ICP Header usICPCount and iICPStatus fields are required.
[b] The ICP requires that each DLI_ICP_CMD_ATTACH command written to the ICP declare the ICP node number (3–126) through which the ICP is to send responses to the application. The DLI application programming interface handles this automatically.
[c] This returned Session ID must be provided on all subsequent writes for this link.

5.2 Bind

Table 5–4 shows the DLI_PROT_SEND_BIND header format. See Section 4.2.2 on page 31 for a functional description. There is no associated data area.

Table 5–4: DLI_PROT_SEND_BIND Header Format

Field	Command Value (Write)	Response Value (Read)
Freeway Header (network byte-order; used in DLI calls only):		
usFWPacketType	= FW_DATA	= FW_DATA
usFWCommand	= FW_ICP_WRITE	= FW_ICP_READ
usFWStatus	= 0	= 0
ICP Header[a]		
usICPClientID	= 0	= 0
usICPServerID	= 0	= 0
[a]usICPCount	= 16 (non-DLI calls only)	= 16 (non-DLI calls only)
usICPCommand	= DLI_ICP_CMD_WRITE	= DLI_ICP_CMD_READ
[a]iICPStatus	= Client memory order (Table 5–2 on page 41)	= Status code (Table 6–1 on page 54)
usICPParms[0]	= 0	= ICP return node from Attach cmd
usICPParms[1]	= 0	= 0
usICPParms[2]	= 0	= 0
Protocol Header[a]		
usProtCommand	= DLI_PROT_SEND_BIND	= DLI_PROT_RESP_BIND
iProtModifier	= Number of transmit buffers to be accumulated before first enabling transmission (0 same as 1)	= Status code (Table 6–2 on page 54)
usProtLinkID	= Link number	= Link number
usProtCircuitID	= 0	= 0
usProtSessionID	= Session ID from Attach response	= Session ID from Attach response
usProtSequence	= 0	= 0
usProtXParms[0]	= 0	= 0
usProtXParms[1]	= 0	= 0

[a] Requirements for non-DLI calls: ICP Header fields must be in network byte-order, Protocol Header fields must be in client byte order, and the ICP Header usICPCount and iICPStatus fields are required.

5.3 Configure Link

Table 5–5 shows the DLI_PROT_CFG_LINK header format. See Section 4.2.1 on page 31 for a functional description and information regarding the associated data area.

Table 5–5: DLI_PROT_CFG_LINK Header Format

Field	Command Value (Write)	Response Value (Read)
Freeway Header (network byte-order; used in DLI calls only):		
usFWPacketType	= FW_DATA	= FW_DATA
usFWCommand	= FW_ICP_WRITE	= FW_ICP_READ
usFWStatus	= 0	= 0
ICP Header[a]		
usICPClientID	= 0	= 0
usICPServerID	= 0	= 0
[a]usICPCount	= 16 + data size (non-DLI calls only)	= 16 + data size (non-DLI calls only)
usICPCommand	= DLI_ICP_CMD_WRITE	= DLI_ICP_CMD_READ
[a]iICPStatus	= Client memory order (Table 5–2 on page 41)	= Status code (Table 6–1 on page 54)
usICPParms[0]	= 0	= ICP return node from Attach cmd
usICPParms[1]	= 0	= 0
usICPParms[2]	= 0	= 0
Protocol Header[a]		
usProtCommand	= DLI_PROT_CFG_LINK	= DLI_PROT_RESP_CFG_LINK
iProtModifier	= 0	= Status code (Table 6–2 on page 54)
usProtLinkID	= Link number	= Link number
usProtCircuitID	= 0	= 0
usProtSessionID	= Session ID from Attach response	= Session ID from Attach response
usProtSequence	= 0	= 0
usProtXParms[0]	= 0	= 0[b]
usProtXParms[1]	= 0	= 0[b]

[a] Requirements for non-DLI calls: ICP Header fields must be in network byte-order, Protocol Header fields must be in client byte order, and the ICP Header usICPCount and iICPStatus fields are required.
[b] When iICPStatus contains a negative error code, usProtXParms[0] and usProtXParms[1] report the invalid configuration item number and value.

5.4 Detach

Table 5–6 shows the DLI_ICP_CMD_DETACH header format. See Section 4.1.3 on page 30 for a functional description. There is no associated data area.

Table 5–6: DLI_ICP_CMD_DETACH Header Format

Field	Command Value (Write)	Response Value (Read)
Freeway Header (network byte-order; used in DLI calls only):		
usFWPacketType	= FW_DATA	= FW_DATA
usFWCommand	= FW_ICP_WRITE	= FW_ICP_READ
usFWStatus	= 0	= 0
ICP Header[a]		
usICPClientID	= 0	= 0
usICPServerID	= 0	= 0
[a]usICPCount	= 16 (non-DLI calls only)	= 16 (non-DLI calls only)
usICPCommand	= DLI_ICP_CMD_DETACH	= DLI_ICP_CMD_DETACH
[a]iICPStatus	= Client memory order (Table 5–2 on page 41)	= Status code (Table 6–1 on page 54)
usICPParms[0]	= 0	= ICP return node from Attach cmd
usICPParms[1]	= 0	= 0
usICPParms[2]	= 0	= 0
Protocol Header[a]		
usProtCommand	= DLI_ICP_CMD_DETACH	= DLI_ICP_CMD_DETACH
iProtModifier	= 0	= Status code (Table 6–2 on page 54)
usProtLinkID	= Link number	= Link number
usProtCircuitID	= 0	= 0
usProtSessionID	= Session ID from Attach response	= Session ID from Attach response
usProtSequence	= 0	= 0
usProtXParms[0]	= 0	= 0
usProtXParms[1]	= 0	= 0

[a] Requirements for non-DLI calls: ICP Header fields must be in network byte-order, Protocol Header fields must be in client byte order, and the ICP Header usICPCount and iICPStatus fields are required.

5.5 Get Software Version

Table 5–7 shows the DLI_PROT_GET_SOFTWARE_VER header format. See Section 4.4.1 on page 35 for a functional description and the report format associated with the response.

Table 5–7: DLI_PROT_GET_SOFTWARE_VER Header Format

Field	Command Value (Write)	Response Value (Read)
Freeway Header (network byte-order; used in DLI calls only):		
usFWPacketType	= FW_DATA	= FW_DATA
usFWCommand	= FW_ICP_WRITE	= FW_ICP_READ
usFWStatus	= 0	= 0
ICP Header[a]		
usICPClientID	= 0	= 0
usICPServerID	= 0	= 0
[a]usICPCount	= 16 (non-DLI calls only)	= 16 + data size (non-DLI calls only)
usICPCommand	= DLI_ICP_CMD_WRITE	= DLI_ICP_CMD_READ
[a]iICPStatus	= Client memory order (Table 5–2 on page 41)	= Status code (Table 6–1 on page 54)
usICPParms[0]	= 0	= ICP return node from Attach cmd
usICPParms[1]	= 0	= 0
usICPParms[2]	= 0	= 0
Protocol Header[a]		
usProtCommand	= DLI_PROT_GET_SOFTWARE_VER	= DLI_PROT_RESP_SOFTWARE_VER
iProtModifier	= 0	= Number of ICP ports (if successful) or error code (Table 6–2 on page 54)
usProtLinkID	= Link number[b]	= Link number
usProtCircuitID	= 0	= 0
usProtSessionID	= Session ID from Attach response	= Session ID from Attach response
usProtSequence	= 0	= 0
usProtXParms[0]	= 0	= 0
usProtXParms[1]	= 0	= 0

[a] Requirements for non-DLI calls: ICP Header fields must be in network byte-order, Protocol Header fields must be in client byte order, and the ICP Header usICPCount and iICPStatus fields are required.
[b] Since this command is not link-specific, the link number of any attached link can be specified.

5.6 Get Statistics Report

Table 5–8 shows the DLI_PROT_GET_STATISTICS header format. See Section 4.4.2 on page 35 for a functional description and the report format associated with the response.

Table 5–8: DLI_PROT_GET_STATISTICS Header Format

Field	Command Value (Write)	Response Value (Read)
Freeway Header (network byte-order; used in DLI calls only):		
usFWPacketType	= FW_DATA	= FW_DATA
usFWCommand	= FW_ICP_WRITE	= FW_ICP_READ
usFWStatus	= 0	= 0
ICP Header[a]		
usICPClientID	= 0	= 0
usICPServerID	= 0	= 0
[a]usICPCount	= 16 (non-DLI calls only)	= 16 + data size (non-DLI calls only)
usICPCommand	= DLI_ICP_CMD_WRITE	= DLI_ICP_CMD_READ
[a]iICPStatus	= Client memory order (Table 5–2 on page 41)	= Status code (Table 6–1 on page 54)
usICPParms[0]	= 0	= ICP return node from Attach cmd
usICPParms[1]	= 0	= 0
usICPParms[2]	= 0	= 0
Protocol Header[a]		
usProtCommand	= DLI_PROT_GET_STATISTICS	= DLI_PROT_RESP_STATISTICS
iProtModifier	= 0	= Status code (Table 6–2 on page 54)
usProtLinkID	= Link number	= Link number
usProtCircuitID	= 0	= 0
usProtSessionID	= Session ID from Attach response	= Session ID from Attach response
usProtSequence	= 0	= 0
usProtXParms[0]	= 0	= 0
usProtXParms[1]	= 0	= 0

[a] Requirements for non-DLI calls: ICP Header fields must be in network byte-order, Protocol Header fields must be in client byte order, and the ICP Header usICPCount and iICPStatus fields are required.

5.7 Receive Data

Table 5–9 shows the DLI_PROT_RECV_DATA header format. See Section 4.3.3 on page 34 for a functional description and information regarding the associated data area.

Table 5–9: DLI_PROT_RECV_DATA Header Format

Field	Response Value (Read)
Freeway Header (network byte-order; used in DLI calls only):	
usFWPacketType	= FW_DATA
usFWCommand	= FW_ICP_READ
usFWStatus	= 0
ICP Header[a]	
usICPClientID	= 0
usICPServerID	= 0
[a]usICPCount	= 16 + data size (non-DLI calls only)
usICPCommand	= DLI_ICP_CMD_READ
iICPStatus	= Status code (Table 6–1 on page 54)
usICPParms[0]	= ICP return node from Attach cmd
usICPParms[1]	= 0
usICPParms[2]	= 0
Protocol Header[a]	
usProtCommand	= DLI_PROT_RECV_DATA
iProtModifier	= Status code (Table 6–2 on page 54)
usProtLinkID	= Link number
usProtCircuitID	= 0
usProtSessionID	= Session ID from Attach response
usProtSequence	= 0
usProtXParms[0]	= 0
usProtXParms[1]	= 0

[a] Requirements for non-DLI calls: ICP Header fields must be in network byte-order, Protocol Header fields must be in client byte order, and the ICP Header usICPCount field is required.

5.8 Send Data

Table 5–10 shows the DLI_PROT_SEND_DATA header format. See Section 4.3.1 on page 34 for a functional description and information regarding the associated data area.

Table 5–10: DLI_PROT_SEND_DATA Header Format

Field	Command Value (Write)	Response Value (Read)
Freeway Header (network byte-order; used in DLI calls only):		
usFWPacketType	= FW_DATA	= FW_DATA
usFWCommand	= FW_ICP_WRITE	= FW_ICP_READ
usFWStatus	= 0	= 0
ICP Header[a]		
usICPClientID	= 0	= 0
usICPServerID	= 0	= 0
[a]usICPCount	= 16 + data size (non-DLI calls only)	= 16 (non-DLI calls only)
usICPCommand	= DLI_ICP_CMD_WRITE	= DLI_ICP_CMD_READ
[a]iICPStatus	= Client memory order (Table 5–2 on page 41)	= Status code (Table 6–1 on page 54)
usICPParms[0]	= 0	= ICP return node from Attach cmd
usICPParms[1]	= 0	= 0
usICPParms[2]	= 0	= 0
Protocol Header[a]		
usProtCommand	= DLI_PROT_SEND_DATA	= DLI_PROT_RESP_DATA_ACK
iProtModifier	= 0	= Status code (Table 6–2 on page 54)
usProtLinkID	= Link number	= Link number
usProtCircuitID	= 0	= 0
usProtSessionID	= Session ID from Attach response	= Session ID from Attach response
usProtSequence	= 0	= 0
usProtXParms[0]	= 0	= 0
usProtXParms[1]	= 0	= 0

[a] Requirements for non-DLI calls: ICP Header fields must be in network byte-order, Protocol Header fields must be in client byte order, and the ICP Header usICPCount and iICPStatus fields are required.

5.9 Set Buffer Size

Table 5–11 shows the DLI_PROT_SET_BUF_SIZE header format. See Section 4.1.2 on page 30 for a functional description and information regarding the associated data area.

Table 5–11: DLI_PROT_SET_BUF_SIZE Header Format

Field	Command Value (Write)	Response Value (Read)
Freeway Header (network byte-order; used in DLI calls only):		
usFWPacketType	= FW_DATA	= FW_DATA
usFWCommand	= FW_ICP_WRITE	= FW_ICP_READ
usFWStatus	= 0	= 0
ICP Header[a]		
usICPClientID	= 0	= 0
usICPServerID	= 0	= 0
[a]usICPCount	= 18 (non-DLI calls only)	= 18 (non-DLI calls only)
usICPCommand	= DLI_ICP_CMD_WRITE	= DLI_ICP_CMD_READ
[a]iICPStatus	= Client memory order (Table 5–2 on page 41)	= Status code (Table 6–1 on page 54)
usICPParms[0]	= 0	= ICP return node from Attach cmd
usICPParms[1]	= 0	= 0
usICPParms[2]	= 0	= 0
Protocol Header[a]		
usProtCommand	= DLI_PROT_SET_BUF_SIZE	= DLI_PROT_RESP_BUF_SIZE
iProtModifier	= 0	= Status code (Table 6–2 on page 54)
usProtLinkID	= Link number[b]	= Link number
usProtCircuitID	= 0	= 0
usProtSessionID	= Session ID from Attach response	= Session ID from Attach response
usProtSequence	= 0	= 0
usProtXParms[0]	= 0	= 0
usProtXParms[1]	= 0	= 0

[a] Requirements for non-DLI calls: ICP Header fields must be in network byte-order, Protocol Header fields must be in client byte order, and the ICP Header usICPCount and iICPStatus fields are required.
[b] Since this command is not link-specific, the link number of any attached link can be specified.

5.10 Unbind

Table 5–12 shows the DLI_PROT_SEND_UNBIND header format. See Section 4.2.3 on page 33 for a functional description. There is no associated data area.

Table 5–12: DLI_PROT_SEND_UNBIND Header Format

Field	Command Value (Write)	Response Value (Read)
Freeway Header (network byte-order; used in DLI calls only):		
usFWPacketType	= FW_DATA	= FW_DATA
usFWCommand	= FW_ICP_WRITE	= FW_ICP_READ
usFWStatus	= 0	= 0
ICP Header[a]		
usICPClientID	= 0	= 0
usICPServerID	= 0	= 0
[a]usICPCount	= 16 (non-DLI calls only)	= 16 (non-DLI calls only)
usICPCommand	= DLI_ICP_CMD_WRITE	= DLI_ICP_CMD_READ
[a]iICPStatus	= Client memory order (Table 5–2 on page 41)	= Status code (Table 6–1 on page 54)
usICPParms[0]	= 0	= ICP return node from Attach cmd
usICPParms[1]	= 0	= 0
usICPParms[2]	= 0	= 0
Protocol Header[a]		
usProtCommand	= DLI_PROT_SEND_UNBIND	= DLI_PROT_RESP_UNBIND
iProtModifier	= 0	= Status code (Table 6–2 on page 54)
usProtLinkID	= Link number	= Link number
usProtCircuitID	= 0	= 0
usProtSessionID	= Session ID from Attach response	= Session ID from Attach response
usProtSequence	= 0	= 0
usProtXParms[0]	= 0	= 0
usProtXParms[1]	= 0	= 0

[a] Requirements for non-DLI calls: ICP Header fields must be in network byte-order, Protocol Header fields must be in client byte order, and the ICP Header usICPCount and iICPStatus fields are required.

Chapter

6 Error Conditions

The ICP protocol software uses both the `iICPStatus` and `iProtModifier` header fields to report success and error status codes, as described in the following sections.

6.1 iICPStatus Field Codes

The `iICPStatus` field serves two purposes. First, when the client application writes a command, this field identifies the client's memory organization as Big Endian (0x0000) or Little Endian (0x4000). If the client application uses the DLI application program interface, the DLI automatically fills in the correct memory organization indicator when the client application writes a command to the protocol software. If the client application uses any other means to access the protocol software, the application must fill this field in correctly.

The second purpose of the `iICPStatus` field is to return the `DLI_ICP_CMD_STATUS_OK` success code (zero) or a negative error code for the "`DLI_ICP_CMD`" commands (attach and detach). The status codes are listed in Table 6–1. The client application should always use the symbolic name when referencing a specific error within program code, because the assigned values defined within the `bsp_errs.h` file (Section A.3 on page 65) are subject to change.

Table 6–1: Status Codes Reported in the `iICPStatus` Field

Status Code (**iICPStatus field**)	Description
`DLI_ICP_CMD_STATUS_OK`	Successful
`DLI_ICP_ERR_ALREADY_ATTACHED`	Link already attached
`DLI_ICP_ERR_BAD_CMD`	Invalid ICP command
`DLI_ICP_ERR_BAD_LINK`	Invalid ICP link
`DLI_ICP_ERR_BAD_NODE`	Invalid ICP node
`DLI_ICP_ERR_BAD_SESSID`	Invalid ICP session ID

6.2 iProtModifier Field Codes

When the application writes a command, the `iProtModifier` field is not used and should be set to zero. When the application reads a response from any of the "`DLI_PROT`" commands, this field contains the `PROT_CMD_STATUS_OK` success code or a negative error code. The status codes are listed in Table 6–2. The client application should always use the symbolic name when referencing a specific error within program code, because the assigned values defined within the `bsp_errs.h` file (Section A.3 on page 65) are subject to change.

Table 6–2: Status Codes Reported in the `iProtModifier` Field

Status Code (**iProtModifier field**)	Description
`PROT_CMD_STATUS_OK`	Successful
`PROT_CMD_ERR_BUF_FIXED_SIZE`	ICP Buffer size not modifiable
`PROT_CMD_ERR_BUF_INVALID_SIZE`	Invalid ICP buffer size parameter
`PROT_CMD_ERR_CMD_INVALID`	Invalid protocol command
`PROT_CMD_ERR_CONFIG_CLOCK`	Invalid clock source link parameter
`PROT_CMD_ERR_CONFIG_DATA_DIR`	Invalid data direction link parameter
`PROT_CMD_ERR_CONFIG_ELECTRICAL`	Invalid electrical interface link parameter
`PROT_CMD_ERR_CONFIG_ENCODING`	Invalid bit encoding link parameter

Table 6–2: Status Codes Reported in the `iProtModifier` Field *(Cont'd)*

Status Code (iProtModifier field)	Description
PROT_CMD_ERR_CONFIG_RATE	Invalid data rate link parameter
PROT_CMD_ERR_LINK_ACTIVE	Link is enabled/bound
PROT_CMD_ERR_LINK_INACTIVE	Link is disabled/unbound
PROT_CMD_ERR_XMIT_NOXMT	Transmit data supplied for receive-only link
PROT_CMD_ERR_XMIT_SIZE	Transmit data exceeds ICP buffer size

Appendix

A Include Files

Table A–1 summarizes the include files normally required by application programs for inclusion of the definitions of symbolically named values and data structures referenced in this document.

Table A–1: Include Files

Include File	Description	Reference Section
`bsp_blks.h`	Protocol block structure and field definitions	Section A.1 on page 58
`bsp_defs.h`	Data symbol definitions	Section A.2 on page 62
`bsp_errs.h`	Error code symbol definitions	Section A.3 on page 65

A.1 bsp_blks.h

```
/*****************************************************************************
*
*               CONFIDENTIAL & PROPRIETARY INFORMATION
*              Distribution to Authorized Personnel Only
*              Unpublished/Copyright 1998 Simpact, Inc.
*                        All Rights Reserved
*
* This document contains confidential and proprietary information of
* Simpact, Inc and is protected by copyright, trade secret and state
* and federal laws. The possession or receipt of this information
* does not convey any right to disclose its contents, reproduce it, or use, or
* license the use, for manufacture or sale, the information or anything
* described therein. Any use, disclosure, or reproduction without Simpact's
* prior written permission is strictly prohibited.
*
* Software and Technical Data Rights
*
* Simpact software products and related documentation will be furnished
* hereunder with "Restricted Rights" in accordance with:
*
*      A. Subparagraph (c)(1)(ii) of the clause entitled Rights in Technical
*      Data and Computer Software (OCT 1988) located at DFARS 252.227-7013; or
*
*      B. Subparagraph (c)(2) of the clause entitled Commercial Computer
*      Software - Restricted Rights (JUN 1987) located at FAR 52.227.19.
*
*****************************************************************************/

/*****************************************************************************
*
*   bsp_blks.h
*
*   This file supplies client/ICP interface data structure definitions.
*
*****************************************************************************
*
* REVISION HISTORY
*
*  Date       Initials  Description
*  ----       --------  -----------
*  16-SEP-98  DMS       Original version.
*
*****************************************************************************/

#ifndef DLI
```

```
/*****************************************************************************
 *
 *  ICP message definition for non-DLI client calls.
 *  The following is declared at the end, following declarations
 *  for each of its three parts:
 *
 *    typedef struct _ICP_PACKET
 *    {
 *       ICP_HDR   icp_hdr;
 *       PROT_HDR  prot_hdr;
 *       DATA      data;
 *    } ICP_PACKET;
 *
 */

typedef struct _ICP_HDR             /* ICP message header */
{
   unsigned short usICPClientID;    /* Old su_id (unused) */
   unsigned short usICPServerID;    /* Old sp_id (unused) */
   unsigned short usICPCount;       /* Size of PROT_HDR plus data */
   unsigned short usICPCommand;     /* ICP command */
            short usICPStatus;      /* ICP command status */
   unsigned short usICPParms[3];    /* ICP extra parameters */
} ICP_HDR;
#define ICP_HDR_SIZE  sizeof(ICP_HDR)

typedef struct _PROT_HDR            /* PROTOCOL message header */
{
   unsigned short usProtCommand;    /* Protocol command */
            short iProtModifier;    /* Protocol command status */
   unsigned short usProtLinkID;     /* Protocol link ID */
   unsigned short usProtCircuitID;  /* Protocol circuit ID (unused) */
   unsigned short usProtSessionID;  /* Protocol session ID */
   unsigned short usProtSequence;   /* Protocol sequence (unused) */
   unsigned short usProtXParms[2];  /* Protocol extra parameters */
} PROT_HDR;
#define PROT_HDR_SIZE  sizeof(PROT_HDR)

typedef union _DATA                 /* Message data area -- variable length */
{
   unsigned int   lwords[1];        /* Data array as 32-bit units */
   unsigned short words [2];        /* Data array as 16-bit units */
   unsigned char  bytes [4];        /* Data array as  8-bit units */
} DATA;

typedef struct _ICP_PACKET          /* ICP full message structure */
{
    ICP_HDR       icp_hdr;          /* ICP header      (network byte-order) */
    PROT_HDR      prot_hdr;         /* Protocol header (client  byte-order) */
    DATA          data;             /* Data area       (client  byte-order) */
} ICP_PACKET;
#define PKT_HDR_SIZE  (ICP_HDR_SIZE+PROT_HDR_SIZE)

#endif /* not DLI */
```

```
/*****************************************************************************
 *
 *  Typical non-DLI data message packet access:
 *
 *    ICP_PACKET    *pMsg;
 *    unsigned char *pData;
 *    pMsg  = (ICP_PACKET *) malloc ( PKT_HDR_SIZE + MAXDATA );
 *    pData = pMsg->data.bytes;
 *
 *  Typical DLI data message packet access:
 *
 *    unsigned char *pData;
 *    pData = (unsigned char *) malloc ( MAXDATA );
 *
 */

/*****************************************************************************
 *
 *  Configure Link command data structure definition.
 *  This structure is supplied as the data area for DLI calls
 *  and as the DATA substructure for non-DLI calls.
 *
 *  Typical non-DLI message packet access:
 *
 *    ICP_PACKET *pMsg;
 *    CONF_TYPE  *pConf;
 *    pMsg  = (ICP_PACKET *) malloc ( PKT_HDR_SIZE + CONF_SIZE );
 *    pConf = (CONF_TYPE  *) &pMsg->data;
 *
 *  Typical DLI message packet access:
 *
 *    CONF_TYPE  *pConf;
 *    pConf = (CONF_TYPE *) malloc ( CONF_SIZE );
 *
 */

typedef struct _CONF_TYPE
{
    unsigned int   rate;           /* baud rate                  */
    unsigned char  clock;          /* 0 = external, 1 = internal */
    unsigned char  electrical;     /* electrical interface       */
    unsigned char  encoding;       /* 0 = NRZ, 1 = NRZB          */
    unsigned char  xmtidle;        /* transmit idle character    */
    unsigned char  datadir;        /* data direction(s)          */
    unsigned char  reserved[3];    /* LWORD ALIGNMENT            */
} CONF_TYPE;
#define CONF_SIZE  sizeof(CONF_TYPE)
```

```
/*****************************************************************************
 *
 *  Link Statistics Report response data structure definition.
 *  This structure is supplied as the data area for DLI calls
 *  and as the DATA substructure for non-DLI calls.
 *
 *  Typical non-DLI message packet access:
 *
 *    ICP_PACKET  *pMsg;
 *    STATS_TYPE  *pStats;
 *    pMsg   = (ICP_PACKET  *) malloc ( PKT_HDR_SIZE + STATS_SIZE );
 *    pStats = (STATS_TYPE  *) &pMsg->data;
 *
 *  Typical DLI message packet access:
 *
 *    STATS_TYPE  *pStats;
 *    pStats = (STATS_TYPE  *) malloc ( STATS_SIZE );
 *
 */

typedef struct _STATS_TYPE
{
    unsigned int   frame_rcvd;     /* # of frames received       */
    unsigned int   frame_sent;     /* # of frames transmitted    */
    unsigned int   rcv_ovrrun;     /* # of receiver overruns     */
    unsigned int   xmt_undrun;     /* # of transmitter underruns */
} STATS_TYPE;
#define STATS_SIZE  sizeof(STATS_TYPE)
#define NUM_STATS   (STATS_SIZE/sizeof(unsigned int))

/*  end bsp_blks.h */
```

A.2 bsp_defs.h

```
/*****************************************************************************
*
*               CONFIDENTIAL & PROPRIETARY INFORMATION
*              Distribution to Authorized Personnel Only
*              Unpublished/Copyright 1998 Simpact, Inc.
*                        All Rights Reserved
*
* This document contains confidential and proprietary information of
* Simpact, Inc and is protected by copyright, trade secret and state
* and federal laws. The possession or receipt of this information
* does not convey any right to disclose its contents, reproduce it, or use, or
* license the use, for manufacture or sale, the information or anything
* described therein. Any use, disclosure, or reproduction without Simpact's
* prior written permission is strictly prohibited.
*
* Software and Technical Data Rights
*
* Simpact software products and related documentation will be furnished
* hereunder with "Restricted Rights" in accordance with:
*
*      A. Subparagraph (c)(1)(ii) of the clause entitled Rights in Technical
*      Data and Computer Software (OCT 1988) located at DFARS 252.227-7013; or
*
*      B. Subparagraph (c)(2) of the clause entitled Commercial Computer
*      Software - Restricted Rights (JUN 1987) located at FAR 52.227.19.
*
*****************************************************************************/

/*****************************************************************************
*
*   bsp_defs.h
*
*   This file supplies client/ICP interface data symbol definitions.
*
*****************************************************************************
*
* REVISION HISTORY
*
*  Date       Initials  Description
*  ----       --------  -----------
*  16-SEP-98  DMS       Original version.
*
*****************************************************************************/
```

```
/*
 * Define "DLI" if DLITE is defined, since client perspective
 * treats DLITE as a flavor of DLI
 */
#ifdef DLITE
#ifndef DLI
#define DLI
#endif
#endif

#ifndef DLI
/*
* Client byte-order indicators, used in non-DLI calls only (iICPStatus field)
*
* Big-Endian    means MSB of word value is lower byte address in memory word
* Little-Endian means LSB of word value is lower byte address in memory word
*/
#define BIG_ENDIAN     0x0000 /* Motorola, SUN, .... */
#define LITTLE_ENDIAN  0x4000 /* DEC, Intel, .... */
#endif /* not DLI */

/*
 * Link configuration data rate index values, named by bits/sec
 *
 * valid indexes are 0-24
 * index 0 means default rate (9600 baud)
 * valid direct rates are 75 and above
 * thus values in range 25-74 are invalid
 */
#define BAUD_DEFAULT         0        /* implied 9600 baud */
#define BAUD_300_INDEX       1
#define BAUD_600_INDEX       2
#define BAUD_1200_INDEX      3
#define BAUD_2400_INDEX      4
#define BAUD_4800_INDEX      5
#define BAUD_9600_INDEX      6
#define BAUD_19200_INDEX     7
#define BAUD_38400_INDEX     8
#define BAUD_56000_INDEX     9
#define BAUD_57600_INDEX    10
#define BAUD_64000_INDEX    11
#define BAUD_76800_INDEX    12
#define BAUD_115200_INDEX   13
#define BAUD_122880_INDEX   14
#define BAUD_153600_INDEX   15
#define BAUD_184320_INDEX   16
#define BAUD_230400_INDEX   17
#define BAUD_307200_INDEX   18
#define BAUD_368640_INDEX   19
#define BAUD_460800_INDEX   20
#define BAUD_614400_INDEX   21
#define BAUD_737280_INDEX   22
#define BAUD_921600_INDEX   23
#define BAUD_1228800_INDEX  24
```

```
/*
 * Link configuration clocking values
 */
#define CLOCK_DEFAULT   0       /* implies external */
#define CLOCK_EXTERNAL  0
#define CLOCK_INTERNAL  1

/*
 * Link configuration electrical interface values
 *
 * Must be EIA_DEFAULT if ICP does not program the electrical interface
 */
#define EIA_DEFAULT  0    /*  implies EIA-232        */
#define EIA_188      1    /*  MIL 188C               */
#define EIA_232      2    /*  EIA 232                */
#define EIA_562      3    /*  EIA 562                */
#define EIA_422      4    /*  EIA 422                */
#define EIA_485      5    /*  EIA 485                */
#define EIA_114B     6    /*  MIL 188-114 balanced   */
                          /*  7 unused               */
#define EIA_423      8    /*  EIA 423                */
#define EIA_114U     9    /*  MIL 188-114 unbalanced */
                          /* 10 unused               */
                          /* 11 unused               */
#define EIA_449     12    /*  EIA 449                */
#define EIA_530     13    /*  EIA 530                */
#define EIA_V35     14    /*  V.35                   */
                          /* 15 and above unused     */
/*
 * Link configuration bit-encoding values
 */
#define ENCODE_DEFAULT  0 /* implies NRZ */
#define ENCODE_NRZ      0
#define ENCODE_NRZB     1

/*
 * Transmit idle values
 */
#define XMTIDLE_DEFAULT  0xff   /* all 1s */

/*
 * Data direction values
 */
#define DATA_DIR_DEFAULT  0x0   /* implies both directions */
#define DATA_DIR_BOTH     0x0
#define DATA_DIR_NOXMT    0x1   /* bit 0 set     ==> no transmit */
#define DATA_DIR_NORCV    0x2   /* bit 1 set     ==> no receive  */
#define DATA_DIR_NONE     0x3   /* both bits set ==> no xmt, no rcv */
```

A.3 bsp_errs.h

```
/*****************************************************************************
*
*               CONFIDENTIAL & PROPRIETARY INFORMATION
*              Distribution to Authorized Personnel Only
*              Unpublished/Copyright 1998 Simpact, Inc.
*                        All Rights Reserved
*
* This document contains confidential and proprietary information of
* Simpact, Inc and is protected by copyright, trade secret and state
* and federal laws. The possession or receipt of this information
* does not convey any right to disclose its contents, reproduce it, or use, or
* license the use, for manufacture or sale, the information or anything
* described therein. Any use, disclosure, or reproduction without Simpact's
* prior written permission is strictly prohibited.
*
* Software and Technical Data Rights
*
* Simpact software products and related documentation will be furnished
* hereunder with "Restricted Rights" in accordance with:
*
*      A. Subparagraph (c)(1)(ii) of the clause entitled Rights in Technical
*      Data and Computer Software (OCT 1988) located at DFARS 252.227-7013; or
*
*      B. Subparagraph (c)(2) of the clause entitled Commercial Computer
*      Software - Restricted Rights (JUN 1987) located at FAR 52.227.19.
*
*****************************************************************************/

/*****************************************************************************
*
*   bsp_errs.h
*
*   This file supplies client/ICP interface protocol command error definitions.
*
*****************************************************************************
*
* REVISION HISTORY
*
*  Date       Initials  Description
*  ----       --------  -----------
*  16-SEP-98  DMS       Original version.
*
*****************************************************************************/


/*****
 *****  These are the values that may be assigned to the Protocol Header
 *****  signed short iProtModifier field in responses from the ICP.
 *****/
```

```
/* The 0 value is the standard success status */

#define PROT_CMD_STATUS_OK              0


/* Negative values are error statuses */

#define PROT_CMD_ERR_CMD_INVALID        -  1
#define PROT_CMD_ERR_LINK_INACTIVE      -  2
#define PROT_CMD_ERR_LINK_ACTIVE        -  3
#define PROT_CMD_ERR_BUF_INVALID_SIZE   -101
#define PROT_CMD_ERR_BUF_FIXED_SIZE     -102
#define PROT_CMD_ERR_XMIT_SIZE          -201
#define PROT_CMD_ERR_XMIT_NOXMT         -202
#define PROT_CMD_ERR_CONFIG_RATE        -301
#define PROT_CMD_ERR_CONFIG_CLOCK       -302
#define PROT_CMD_ERR_CONFIG_ELECTRICAL  -303
#define PROT_CMD_ERR_CONFIG_ENCODING    -304
#define PROT_CMD_ERR_CONFIG_DATA_DIR    -305


/*
 *  NOTE:  Positive values
 *
 *  Instead of zero, a positive value is returned with the
 *  successful DLI_PROT_RESP_SOFTWARE_VER response.
 *  It represents the number of physical ports on the ICP.
 *
 */
```

Index

A

Access category 30
  attach 30
    header format 42
  detach 30
    header format 45
  set ICP buffer size 30
  summary 29, 38
Access modes
  session 30
Acknowledgment 34
Attach 25, 30
  header format 42
  session ID 25, 30, 42
Audience 11

B

Bind 26, 31
  header format 43
bsp_blks.h 57
bsp_defs.h 57
bsp_errs.h 57
Buffer size
  header format 50
Buffer size for ICP 25

C

C data structures
  configure link 31
  DLI optional arguments 39
  ICP packet (driver calls) 40
  statistics report 36
Client memory order
  big endian 41
  little endian 41
Command/response sequence 24
Commands 29
  access category 29, 38
  data category 29, 38
  link category 29, 38
  report category 29, 38
  summary table
    functional description 29
    header format 38
  typical sequence 23
Comparison
  DLI versus driver calls 41
Configuration
  DLI protocol parameter 18
  options 32
Configure link 25, 31
  C data structure 31
  header format 44
  table of options 32
Customer support 14

D

Data
  receive 26, 34
  send 26, 34
Data category 34
  normal data
    header format 48, 49
  receive acknowledgment 34
  receive data 34
  send data 34
  summary 29, 38
Data link interface, *see* DLI
Detach 27, 30
  header format 45

Digital UNIX
  embedded ICP 17
DLI
  comparison with driver 41
  dlInit function 18
  dlOpen function 18
  dlRead function 18
  dlWrite function 18
  optional arguments 18, 41
    versus ICP packet 41
  programming interface 18
  protocol parameter 18
  raw operation 18
    protocol parameter 18
DLI configuration file 19
dlInit function 18
dlOpen function 18
dlRead function 18
dlWrite function 18
Documents
  reference 12
Driver
  comparison with DLI 41
  ICP packet 41
  ICP packet structure 40
  programming interface 19

E

Embedded ICP 16
  Digital UNIX 17
  OpenVMS 17
  Windows NT 16
Errors
  iICPStatus field error codes 53
  iProtModifier field error codes 54

F

Files
  bsp_blks.h 57
  bsp_defs.h 57
  bsp_errs.h 57
  DLI configuration 19
  include files 57
  test program 19
  TSI configuration 19
Format
  command and response headers 37
Freeway server 15
Functions
  dlInit 18
  dlOpen 18
  dlRead 18
  dlWrite 18

H

Hardware supported 15
  embedded ICP 16
  Freeway server 15
Header
  ICP header fields 41
  protocol header fields 41
  server header fields 41
Header format 37
  attach 42
  bind 43
  configure link 44
  detach 45
  normal data 48, 49
  set buffer size 50
  software version 46
  statistics report 47
  unbind 51
History of revisions 14

I

ICP buffer size 25, 30
ICP header
  fields 41
  network byte order 41
ICP packet
  versus DLI optional arguments 41
ICP packet C data structure 40
iICPStatus field error codes 53
Include files 57
Initialization 23
Introduction to product 15
iProtModifier field error codes 54

L

Link category 31

bind 31
header format 43
configure link 31
header format 44
table of options 32
summary 29, 38
unbind 33
header format 51

N

Network byte order
ICP header 41
server header 41
Normal data
header format 48, 49

O

OpenVMS
embedded ICP 17
Optional arguments
C data structure 39
DLI 18, 41
Options
configure link 32
Overview
protocol 15

P

Product
introduction 15
support 14
Programming interfaces 18
DLI 18
driver 19
socket 19
Protocol
overview 15
theory of operation 21
protocol DLI parameter 18
Protocol header
fields 41

R

Raw operation, DLI 18
protocol parameter 18
Receive
acknowledgment 34
data 26
Receive data 34
Reference documents 12
Report category 35
software version report 35
header format 46
statistics report 35
header format 47
summary 29, 38
Reports 27
Response/command sequence 24
Responses 29
access category 29, 38
data category 29, 38
link category 29, 38
report category 29, 38
summary table
functional description 29
header format 38
typical sequence 23
Revision history 14

S

Send
data 26
Send data 34
Sequence of operations 23
attach 25
bind 26
configure link 25
detach 27
diagram 24
initialization 23
receive data 26
reports 27
send data 26
set ICP buffer size 25
termination 28
unbind 27
Server header
fields 41
network byte order 41
Session

access modes 30
attach command 25, 30
DLI application 18
session ID returned by attach 25, 30, 42
Set buffer size
header format 50
Socket
programming interface 19
Software version report 35
header format 46
Statistics report 35
C data structure 36
header format 47
Support, product 14
Supported hardware 15
embedded ICP 16
Freeway server 15

T

Technical support 14
Termination 28
Test program files 19
TSI configuration file 19

U

Unbind 27, 33
header format 51

W

Windows NT
embedded ICP 16

Customer Report Form

We are constantly improving our products. If you have suggestions or problems you would like to report regarding the hardware, software or documentation, please complete this form and mail it to Simpact at 9210 Sky Park Court, San Diego, CA 92123, or fax it to (619) 560-2838.

If you are reporting errors in the documentation, please enter the section and page number.

Your Name: _____

Company: _____

Address: _____

_____

_____

Phone Number: _____

Product: _____

Problem or
Suggestion: _____

_____

_____

_____

_____

_____

Simpact, Inc.
Customer Service
9210 Sky Park Court
San Diego, CA 92123