Military/Government Protocols Programmer Guide

DC 900-1602I

Protogate, Inc.
12225 World Trade Drive, Suite R
San Diego, CA 92128
February 2003

Protogate, Inc.
12225 World Trade Drive, Suite R
San Diego, CA 92128
(858) 451-0865

Military/Government Protocols Programmer Guide

Printed in the United States of America

Contents

List of Figures 7

List of Tables 9

Preface 11

1 Introduction 19

1.1 Product Overview . . . 19
1.1.1 Freeway Server . . . 19
1.1.2 Embedded ICP . . . 21
1.2 Freeway Client-Server Environment . . . 23
1.2.1 Establishing Freeway Server Internet Addresses . . . 24
1.3 Embedded ICP Environment . . . 24
1.4 Client Operations . . . 24
1.4.1 Defining the DLI and TSI Configuration . . . 24
1.4.2 Opening a Session . . . 25
1.4.3 Exchanging Data with the Remote Application. . . . 25
1.4.4 Closing a Session . . . 25
1.5 Software Description . . . 26
1.6 Hardware Description . . . 26
1.7 Military/Government Protocols Summary . . . 27
1.7.1 Client Configuration of the Military/Government Protocols Environment. 28
1.7.2 Client Control . . . 29

2 DLI Concepts for the Military/Government Protocols Environment 31

2.1 Summary of DLI Concepts . . . 31
2.1.1 Configuration in the Freeway Environment . . . 31
2.1.2 Non-blocking I/O . . . 32

2.1.3 Buffer Management. 33
2.2 Using the DLI in the Military/Government Protocols Environment. 34
2.2.1 Initializing the DLI 34
2.2.2 Military/Government Protocols DLI Session Configuration 34
2.2.3 Opening and Attaching DLI Sessions 35
2.2.4 Detaching and Closing DLI Sessions. 38
2.2.5 Error Reporting. 39

3 Military/Government Protocols DLI Functions 41

3.1 Example Military/Government Protocols Call Sequences 42
3.2 Overview of DLI Functions for Military/Government Protocols 43
3.2.1 DLI Optional Arguments. 45

4 Synchronous Protocol Messages 49

4.1 Synchronous Protocol Message Formats 49
4.1.1 Header Format 50
4.1.1.1 Link Number Field (Byte 0) 50
4.1.1.2 Station Number Field (Byte 1) 50
4.1.1.3 Function Code Field (Byte 2). 50
4.1.1.4 Error Status Field (Byte 3) 50
4.1.1.5 Data Size Field (Word 2) 52
4.1.1.6 Extended Error Status Field (Word 3) 52
4.1.2 Data Area (beginning at Byte 9 / Word 4) 52
4.2 Overview of Requests using Raw dlWrite 53
4.2.1 Configure Segmentation Buffer [1] 54
4.2.2 Configure Communication Buffer [2]. 55
4.2.3 Set Time Stamping [3] 56
4.2.4 Set Signal Monitor Interval [4]. 57
4.2.5 Set Blocking Interval [5] 57
4.2.6 (reserved) [6] 58
4.2.7 Set Link Protocol [7] 58
4.2.8 Configure Link [8] 58
4.2.9 Enable Link [9] 60
4.2.10 Disable Link [10] 60
4.2.11 Request Link Status Report [11] 61
4.2.12 Request Link Statistics Report [12] 61

4.2.13 Request Queue Count Report [13] . . . 61
4.2.14 Output Data Block [14] . . . 62
4.2.15 Set Time Stamp Value [15] . . . 64
4.2.16 Protocol-specific Command [16] . . . 64
4.2.17 Request Version Report [17] . . . 64
4.2.18 Request ICP Links Report [18] . . . 64
4.2.19 Request Buffer Status Report [19] . . . 64
4.3 Overview of Responses using Raw dlRead . . . 66
4.3.1 Configure Segmentation Buffer Response [21] . . . 66
4.3.2 Configure Communication Buffer Response [22] . . . 66
4.3.3 Set Time Stamping Response [23] . . . 66
4.3.4 Set Signal Monitor Interval Response [24] . . . 68
4.3.5 Set Blocking Interval Response [25] . . . 68
4.3.6 (reserved) [26] . . . 68
4.3.7 Set Link Protocol Response [27] . . . 68
4.3.8 Configure Link Response / Link Configuration Report [28] . . . 68
4.3.9 Enable Link Response [29] . . . 69
4.3.10 Disable Link Response [30] . . . 69
4.3.11 Link Status Report [31] . . . 69
4.3.12 Link Statistics Report [32] . . . 72
4.3.13 Queued Output Message Count Report [33] . . . 73
4.3.14 Input Data Block [34] . . . 73
4.3.14.1 Segmentation Header . . . 73
4.3.14.2 System Parameters Area . . . 76
4.3.14.3 Data Messages . . . 76
4.3.15 Link Status Notification [35] . . . 77
4.3.16 Set Time Stamp Value Response [36] . . . 78
4.3.17 Protocol-specific ICP Response [37] . . . 79
4.3.18 (reserved) [38] . . . 79
4.3.19 Transmit Acknowledgment [39] . . . 79
4.3.20 Version Report [40] . . . 80
4.3.21 ICP Links Report [41] . . . 80
4.3.22 Buffer Status Report [42] . . . 80

5 Asynchronous Protocol Messages 83

5.1 Asynchronous Protocol Message Formats . . . 83

5.2 Overview of Requests using Raw dlWrite 85
5.2.1 Output Data Block [51 (Async)] or [14 (Sync)] 85
5.2.2 Configure Link [52 (Async)] or [8 (Sync)] 85
5.2.3 Enable Link [53 (Async)] or [9 (Sync)] 86
5.2.4 Disable Link [54 (Async)] or [10 (Sync)] 86
5.2.5 Set /Clear Modem Signals [55] 86
5.2.6 (reserved) [56] 87
5.2.7 (reserved) [57] 87
5.2.8 Request Link Status Report [58 (Async)] or [31 (Sync)] 87
5.3 Overview of Responses using Raw dlRead 88
5.3.1 Transmit Acknowledgment [71 (Async)] or [39 (Sync)] 89
5.3.2 Configure Link Response [72 (Async)] or [28 (Sync)] 89
5.3.3 Enable Link Response [73 (Async)] or [29 (Sync)] 89
5.3.4 Disable Link Response [74 (Async)] or [30 (Sync)] 89
5.3.5 Link Status Report [78 (Async)] or [31 (Sync)] 89
5.3.6 Data Lost Notification [79] 90
5.3.7 Input Data Block [80 (Async)] or [34 (Sync)] 90
5.3.8 Transmit Negative Acknowledgment [81] 90
5.3.9 Protocol-specific Notification [82] 91

A Military/Government Protocols Loopback Test Program 93
A.1 Overview of the Test Program 94
A.2 Hardware Setup for the Test Program 95
A.3 Running the Test Program 96
A.4 Sample Output from Test Program 98

B DLI and TSI Configuration Process 103

C Military/Government Protocols Input Data Time Stamping 109
C.1 The Concept of Time Stamping 109
C.2 Internal Timing 109
C.3 PCI Host Timing 110

Index 111

List of Figures

Figure 1–1: Freeway Configuration 20
Figure 1–2: Embedded ICP Configuration 21
Figure 1–3: A Typical Freeway Server Environment 23
Figure 1–4: Segmentation and Communication Buffer Usage 28
Figure 2–1: Example DLI Configuration File for Two Freeway Server Links 36
Figure 2–2: Example DLI Configuration File for Two Embedded ICP Links 37
Figure 3–1: "C" Definition of DLI Optional Arguments Structure 45
Figure 4–1: Client Synchronous Protocol Message Format 49
Figure 4–2: Format of Error Status Field 51
Figure 4–3: Link Configuration Block with Two Options 59
Figure 4–4: Output Data Block 63
Figure 4–5: Link Status Report Type II — Word 2 71
Figure 4–6: Input Data Block 74
Figure 4–7: System Parameters Area 75
Figure 5–1: Client Asynchronous Protocol Message Format 83
Figure 5–2: Link Configuration Block with Two Options 86
Figure 5–3: Link Status Report, Type A 90
Figure A–1: Sample Output from Loopback Program 99
Figure B–1: DLI and TSI Configuration Process 107

List of Tables

Table 3–1: DLI Call Sequence for Military/Government Protocols (Non-blocking I/O). 42
Table 3–2: DLI Functions: Syntax and Parameters (Listed in Typical Call Order) 44
Table 3–3: Required dlWrite Optional Arguments Fields 46
Table 3–4: Relevant dlRead Optional Arguments Fields 47
Table 4–1: Meanings of Error Bits in Error Status Field 51
Table 4–2: Synchronous Protocol Messages Originating in the Client 53
Table 4–3: Synchronous Protocols Messages Originating in the ICP 67
Table 4–4: Link Status Report (Type I) 70
Table 4–5: Link Status Report (Type II) 70
Table 4–6: Link Status Report (Type III) 71
Table 4–7: Link Statistics Report 72
Table 4–8: Link Status Notification 78
Table 4–9: Buffer Status Report 81
Table 5–1: Status Word Field Description (ICP-to-Client Message) 84
Table 5–2: Asynchronous-Protocol Function Codes Originating in the Client. 85
Table 5–3: Asynchronous-Protocol Function Codes Originating in the ICP 88
Table B–1: Configuration File Names. 104

Preface

Purpose of Document

This document describes the operation and programming interface for using Protogate's Military/Government Protocols product for the Freeway communications server (or embedded ICP). This product implements several synchronous and asynchronous protocols.

Note

In this document, the term "Freeway" can mean either a Freeway server or an embedded ICP. For the embedded ICP, also refer to the user's guide for your ICP and operating system (for example, the *Freeway Embedded ICP2432 User's Guide for Windows NT*).

Intended Audience

This document should be read by programmers who are interfacing a client application program to Protogate's Military/Government Protocols product running on a Freeway server or an embedded ICP. You should understand the Freeway data link interface (DLI), as explained in the *Freeway Data Link Interface Reference Guide*.

This document supports your protocol-specific *Military/Government Protocols Programmer's Guide* (see the *Protogate References* section on page 13 for a complete list). You must be familiar with the communication message formats detailed in the protocol-specific guide.

Required Equipment

The Military/Government Protocols product requires the following major hardware components to operate:

- a Freeway communications server or an embedded ICP that runs the communications software
- a client computer that runs the following:
  - TCP/IP (for a Freeway server)
  - Freeway DLI
  - the user application program

Organization of Document

Chapter 1 is an overview of Freeway and the Military/Government Protocols product.

Chapter 2 discusses data link interface (DLI) concepts and how they apply specifically to the Military/Government Protocols environment.

Chapter 3 describes how to use the DLI between the client application program and the Military/Government Protocols communications software running on the ICP.

Chapter 4 describes the message formats for Military/Government Protocols synchronous protocol messages.

Chapter 5 describes the message formats for Military/Government Protocols asynchronous protocol messages.

Appendix A describes the Military/Government Protocols loopback test program.

Appendix B is an overview of the configuration process for DLI sessions and TSI connections.

Appendix C is an overview of input data time stamping for Protogate's Military/Government Protocols.

Protogate References

The following general product documentation list is to familiarize you with the available Protogate Freeway and embedded ICP products. The applicable product-specific reference documents are mentioned throughout each document (also refer to the "readme" file shipped with each product). Most documents are available on-line at Protogate's web site, www.protogate.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 1200/1300 Hardware Installation Guide* DC-900-1537
- *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 Electrical Interfaces (Addendum to DC-900-1501)* DC-900-1566
- *ICP2432 Hardware Installation Guide* DC-900-1502

Freeway Software Installation and Configuration Support

- *Freeway Message Switch User Guide* DC-900-1588
- *Freeway Release Addendum: Client Platforms* DC-900-1555
- *Freeway User Guide* DC-900-1333

- *Freeway Loopback Test Procedures* DC-900-1533

Embedded ICP Software Installation and Programming Support

- *ICP2432 User Guide for Digital UNIX* DC-900-1513
- *ICP2432 User Guide for OpenVMS Alpha* DC-900-1511
- *ICP2432 User Guide for OpenVMS Alpha (DLITE Interface)* DC-900-1516
- *ICP2432 User Guide for Solaris STREAMS* DC-900-1512
- *ICP2432 User Guide for Windows NT* DC-900-1510
- *ICP2432 User 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 OS/Protogate Programmer's Guide* DC-900-2008
- *Freeway Protocol Software Toolkit Programmer's Guide* [with OS/Protogate] DC-900-2008
- *Freeway Server-Resident Application and Server Toolkit Programmer Guide* DC-900-1325
- *OS/Impact Programmer Guide* DC-900-1030
- *Protocol Software Toolkit Programmer Guide* [with OS/Impact] DC-900-1338

Military/Government Protocols Support (Synchronous)

- *Military/Government Protocols Programmer's Guide* DC-900-1602
- *AIRCAT 500 Military/Government Protocol Programmer's Guide* DC-908-1605
- *ATDL-1/UDL Military/Government Protocol Programmer's Guide* DC-908-1606
- *CD2 Military/Government Protocol Programmer's Guide* DC-900-1607
- *Common Format Military/Government Protocol Programmer's Guide* DC-908-1608
- *IDL Military/Government Protocol Programmer's Guide* DC-908-1609
- *Lateral Tell Military/Government Protocol Programmer's Guide* DC-908-1610

- *Link 1 Military/Government Protocol Programmer's Guide* DC-908-1604
- *Link 11B (TADIL-B) Military/Government Protocol Programmer's Guide* DC-908-1601
- *LRR Military/Government Protocol Programmer's Guide* DC-908-1611
- *MBDL Military/Government Protocol Programmer's Guide* DC-908-1612
- *MBDL II Military/Government Protocol Programmer's Guide* DC-908-1613
- *MCC Military/Government Protocol Programmer's Guide* DC-908-1614
- *Nine-Bit Radar Military/Government Protocol Programmer's Guide* DC-908-1615
- *RAT-31S Military/Government Protocol Programmer's Guide* DC-908-1616
- *TACC Military/Government Protocol Programmer's Guide* DC-908-1617
- *Transparent Bisync Radar Military/Government Protocol Programmer's Guide* DC-900-1618

Military/Government Protocols Support (Asynchronous)

- *Asynchronous Military/Government Protocol Programmer's Guide* DC-900-1621
- *ICAO Military/Government Protocol Programmer's Guide* DC-900-1622
- *GDL Military/Government Protocol Programmer's Guide* DC-908-1626
- *Link 14 Military/Government Protocol Programmer's Guide* DC-908-1623
- *MTDL Military/Government Protocol Programmer's Guide* DC-908-1627
- *NOAH Military/Government Protocol Programmer's Guide* DC-908-1624
- *SR162 Military/Government Protocol Programmer's Guide* DC-900-1625

Document Conventions

This document follows the most significant byte first (MSB) and most significant word first (MSW) conventions for bit-numbering and byte-ordering. In all packet transfers between the client applications and the ICPs, the ordering of the byte stream is preserved. However, Military/Government Protocols packed data contains word values that are not byte-swapped.

The term "Freeway" refers to any of the Freeway server models (for example, Freeway 500/3100/3200/3400 PCI-bus servers, Freeway 1000 ISA-bus servers, or Freeway 2000/4000/8800 VME-bus servers). References to "Freeway" also may apply to an

embedded ICP product using DLITE (for example, the embedded ICP2432 using DLITE on a Windows NT system).

The serial ports on the ICPs are logically referred to as "links." This document uses the term "link." The links are logically numbered from 1–n, where n is the number of physical ports on the ICP.

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

Revision History

The revision history of the *Military/Government Protocols Programmer Guide*, Protogate document DC 900-1602I, is recorded below:

Revision	Release Date	Description
DC 900-1602A	October 1998	Original release.
DC 900-1602B	November 1998	Use "Link Status Notification" terminology.
DC 900-1602C	February 1999	Modify Appendix A to provide loopback test output. Modify Appendix B for embedded ICPs. Modify Figure 2–1 on page 36 for embedded ICPs.
DC 900-1602D	June 1999	Add Configure Link Acknowledgment for asynchronous protocols (Section 5.3.2 on page 89)
DC 900-1602E	July 1999	Minor modifications throughout for clarity. Add iICPStatus field information (Table 3–4 on page 47). Enhance buffer sizing information (Section 4.2.1 on page 54 and Section 4.2.2 on page 55). Add protocol-specific command (Section 4.2.16 on page 64) and response (Section 4.3.17 on page 79).
DC 900-1602F	November 1999	Add common (synchronous and asynchronous) function codes to Table 5–2 on page 85 and Table 5–3 on page 88.
DC 900-1602G	February 2003	Update contact information for Protogate, Inc. Add the Link Configuration Report for synchronous protocols (Section 4.2.8 on page 58). Minor modifications throughout for clarity.

Revision	Release Date	Description
DC 900-1602H	December 2002	Update the description of time-stamping in Appendix C. Change the Set Time Stamp Interval command to Set Time Stamping (Section 4.2.3 on page 56); update its description and the description of the Set Time Stamp Value command (Section 4.2.15 on page 64).
DC 900-1602I	February 2003	Add negative acknowledgements of transmit expirations to the Link Statistics report (Table 4–7 on page 72) and to the Transmit Acknowledgement notification (Section 4.3.19 on page 79). Minor modifications throughout for clarity.

Customer Support

If you are having trouble with any Protogate product, call us at (858) 451-0865 Monday through Friday between 8 a.m. and 5 p.m. Pacific time.

You can also fax your questions to us at (877) 473-0190 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

1.1 Product Overview

Protogate provides a variety of wide-area network (WAN) connectivity solutions for real-time financial, defense, telecommunications, and process-control applications. Protogate's Freeway server offers flexibility and ease of programming using a variety of LAN-based server hardware platforms. Now a consistent and compatible embedded intelligent communications processor (ICP) product offers the same functionality as the Freeway server, allowing individual client computers to connect directly to the WAN.

Both Freeway and the embedded ICP use the same data link interface (DLI). Therefore, migration between the two environments simply requires linking your client application with the proper library. Various client operating systems are supported (for example, UNIX, VMS, and Windows NT).

Protogate protocols that run on the ICPs are independent of the client operating system and the hardware platform (Freeway or embedded ICP).

1.1.1 Freeway Server

Protogate's Freeway communications servers enable client applications on a local-area network (LAN) to access specialized WANs through the DLI. The Freeway server can be any of several models (for example, Freeway 1100, Freeway 2000/4000, or Freeway 8000/8800). The Freeway server is user programmable and communicates in real time. It provides multiple data links and a variety of network services to LAN-based clients. Figure 1–1 shows the Freeway configuration.

To maintain high data throughput, Freeway uses a multi-processor architecture to support the LAN and WAN services. The LAN interface is managed by a single-board computer, called the server processor. It uses the commercially available VxWorks operating system to provide a full-featured base for the LAN interface and layered services needed by Freeway.

Freeway can be configured with multiple WAN interface processor boards, each of which is a Protogate ICP. Each ICP runs the communication protocol software using Protogate's real-time operating system.

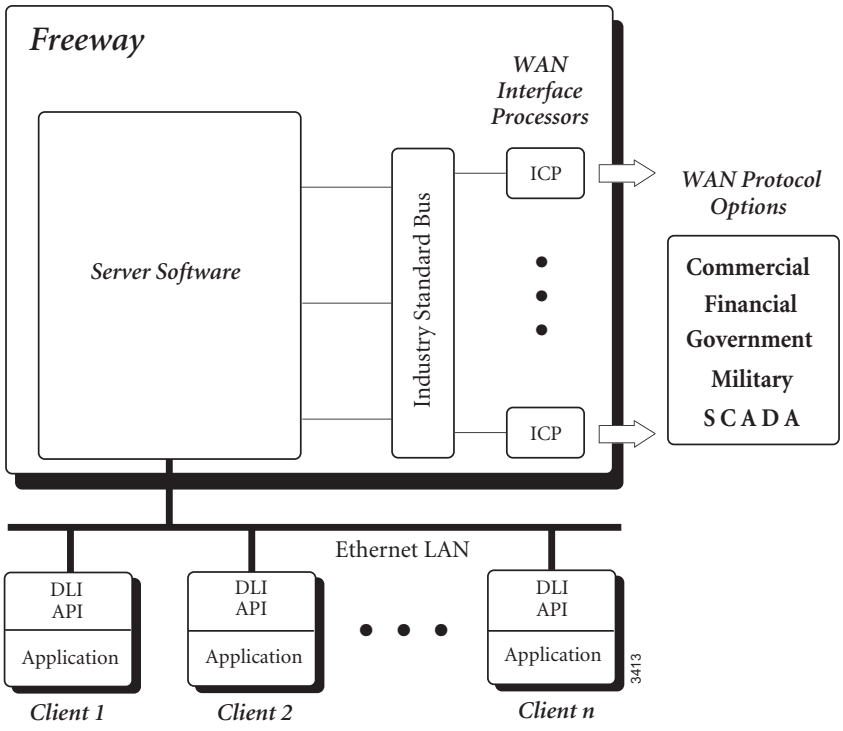

Figure 1–1: Freeway Configuration

1.1.2 Embedded ICP

The embedded ICP connects your client computer directly to the WAN (for example, using Protogate's ICP2432 PCIbus board). The embedded ICP provides client applications with the same WAN connectivity as the Freeway server, using the same data link interface (via the DLITE embedded interface). The ICP runs the communication protocol software using Protogate's real-time operating system. Figure 1–2 shows the embedded ICP configuration.

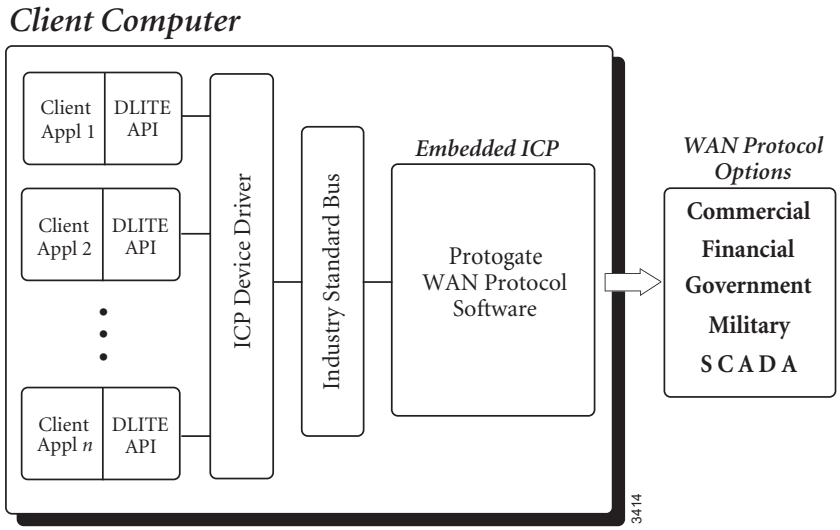

Figure 1–2: Embedded ICP Configuration

Summary of product features:

- Provision of WAN connectivity either through a LAN-based Freeway server or directly using an embedded ICP
- Elimination of difficult LAN and WAN programming and systems integration by providing a powerful and consistent data link interface
- Variety of off-the-shelf communication protocols available from Protogate which are independent of the client operating system and hardware platform
- Support for multiple WAN communication protocols simultaneously
- Support for multiple ICPs (two, four, eight, or sixteen communication lines per ICP)
- Wide selection of electrical interfaces including EIA-232, EIA-449, EIA-530, and V.35
- Creation of customized server-resident and ICP-resident software, using Protogate's software development toolkits
- Freeway server standard support for Ethernet and Fast Ethernet LANs running the transmission control protocol/internet protocol (TCP/IP)
- Freeway server standard support for FDDI LANs running the transmission control protocol/internet protocol (TCP/IP)
- Freeway server management and performance monitoring with the simple network management protocol (SNMP), as well as interactive menus available through a local console, telnet, or rlogin

1.2 Freeway Client-Server Environment

The Freeway server acts as a gateway that connects a client on a local-area network to a wide-area network. Through Freeway, a client application can exchange data with a remote data link application. Your client application must interact with the Freeway server and its resident ICPs before exchanging data with the remote data link application.

One of the major Freeway server components is the message multiplexor (MsgMux) that manages the data traffic between the LAN and the WAN environments. The client application typically interacts with the Freeway MsgMux through a TCP/IP BSD-style socket interface (or a shared-memory interface if it is a server-resident application (SRA)). The ICPs interact with the MsgMux through the DMA and/or shared-memory interface of the industry-standard bus to exchange WAN data. From the client application's point of view, these complexities are handled through a simple and consistent data link interface (DLI), which provides dlOpen, dlWrite, dlRead, and dlClose functions.

Figure 1–3 shows a typical Freeway connected to a locally attached client by a TCP/IP network across an Ethernet LAN interface. Running a client application in the Freeway client-server environment requires the basic steps described in Section 1.4.

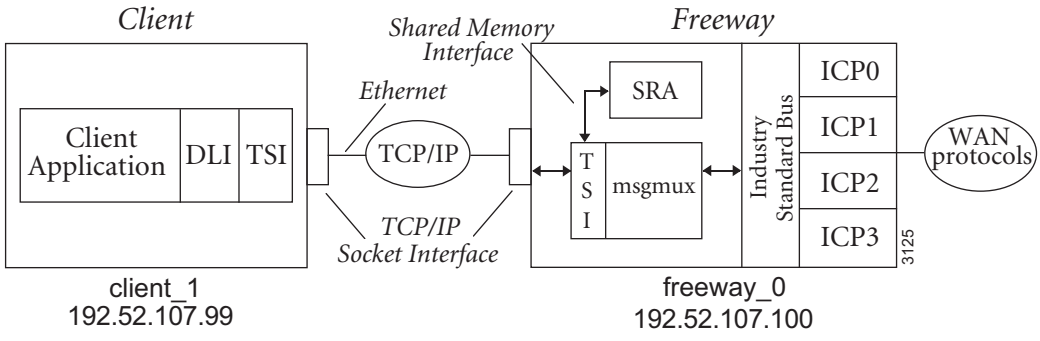

Figure 1–3: A Typical Freeway Server Environment

1.2.1 Establishing Freeway Server Internet Addresses

The Freeway server must be addressable in order for a client application to communicate with it. In the Figure 1–3 example, the TCP/IP Freeway server name is freeway_0, and its unique Internet address is 192.52.107.100. The client machine where the client application resides is client_1, and its unique Internet address is 192.52.107.99. Refer to the *Freeway User Guide* to initially set up your Freeway and download the operating system, server, and protocol software to Freeway.

1.3 Embedded ICP Environment

Refer to the user's guide for your embedded ICP and operating system (for example, the *ICP2432 User Guide for Windows NT*) for software installation and setup instructions. The user's guide also gives additional information regarding the data link interface (DLI) and embedded programming interface descriptions for your specific embedded environment. Refer back to Figure 1–2 on page 21 for a diagram of the embedded ICP environment. Running a client application in the embedded ICP environment requires the basic steps described in Section 1.4

1.4 Client Operations

1.4.1 Defining the DLI and TSI Configuration

You must define the DLI sessions and the transport subsystem interface (TSI) connections between your client application and Freeway (or an embedded ICP). To accomplish this, you first define the configuration parameters in DLI and TSI ASCII configuration files, and then you run two preprocessor programs, dlicfg and tsicfg, to create binary configuration files (see Appendix B). The dlInit function uses the binary configuration files to initialize the DLI environment.

1.4.2 Opening a Session

After the DLI and TSI configurations are properly defined, your client application uses the dlOpen function to establish a DLI session with an ICP link. As part of the session establishment process, the DLI establishes a TSI connection with the Freeway MsgMux through the TCP/IP BSD-style socket interface for the Freeway server, or directly to the client driver for the embedded ICP environment.

1.4.3 Exchanging Data with the Remote Application

After the link is enabled, the client application can exchange data with the remote application using the dlWrite and dlRead functions.

1.4.4 Closing a Session

When your application finishes exchanging data with the remote application, it calls the dlClose function to disable the ICP link, close the session with the ICP, and disconnect from Freeway (or the embedded ICP).

1.5 Software Description

Protogate's Military/Government Protocols product includes the following major software components:

- A group of communications software downloadable images:
    1. Freeway server or embedded ICP software
    2. Real-time operating system (OS/Impact)
    3. Military/Government Protocols communications software
- DLI library for linking with client applications
- Test program (milalp.c) to check product installation (see Appendix A)

The *Freeway User Guide* or the user's guide for your particular embedded ICP and operating system (for example, the *ICP2432 User Guide for Windows NT*) describes the software installation procedures. The DLI provides an interface by which data is exchanged between the client application and Freeway; refer to the *Freeway Data Link Interface Reference Guide*.

1.6 Hardware Description

A typical Freeway configuration of Protogate's Military/Government Protocols product requires the following hardware:

- Freeway communications server (for example, Freeway 1100, Freeway 2000 or Freeway 4000) or an embedded ICP (for example the PCIbus ICP2432)
- Ethernet connection to a client running TCP/IP (for a Freeway server)

1.7 Military/Government Protocols Summary

Protogate's Freeway Military/Government Protocols product implements various synchronous and asynchronous communications protocols. These protocols are typically for tactical data-link or radar interfaces, such as Link 11B for tactical data and CD2 for radar. They may operate simultaneously and cooperatively in an arbitrary combination across the ports of a single ICP. In doing so, they make use of a common interface to the client.

The hardware is Protogate's Freeway communications server (or an embedded ICP) that interfaces a client application with communication data links to local or remote terminals. The software consists of general-purpose Protogate software that has been modified to support the specialized Military/Government Protocols. The combination of this hardware and software off-loads the communication processing from the client computer.

Each link on the Freeway ICPs operates independently and can be assigned any of the implemented protocols, with the result that any combination of the implemented protocols can operate simultaneously on the ICP.

The ICP software transfers individual protocol data messages at the links using relatively small blocks called communication buffers. The ICP/client interface transfers protocol data messages using relatively large blocks called segmentation buffers. Message segmentation reduces the number of client I/O requests, often for a substantial gain in efficiency. The ICP blocks individual data messages received from a serial link, supplying them to the client in a single buffer. The ICP deblocks individual data messages supplied by the client in a single buffer, for transmission to the links. The current incoming-data segmentation buffer for a link is flushed to the client when it becomes full, when its client-configured segmentation timer expires, or when a reportable link status change (such as loss of modem signal) occurs.

See Figure 1–4 for the usage of the segmentation and communication buffers. Section 4.2.1 on page 54 and Section 4.2.2 on page 55 describe the configuration of the respective buffer sizes by the client.

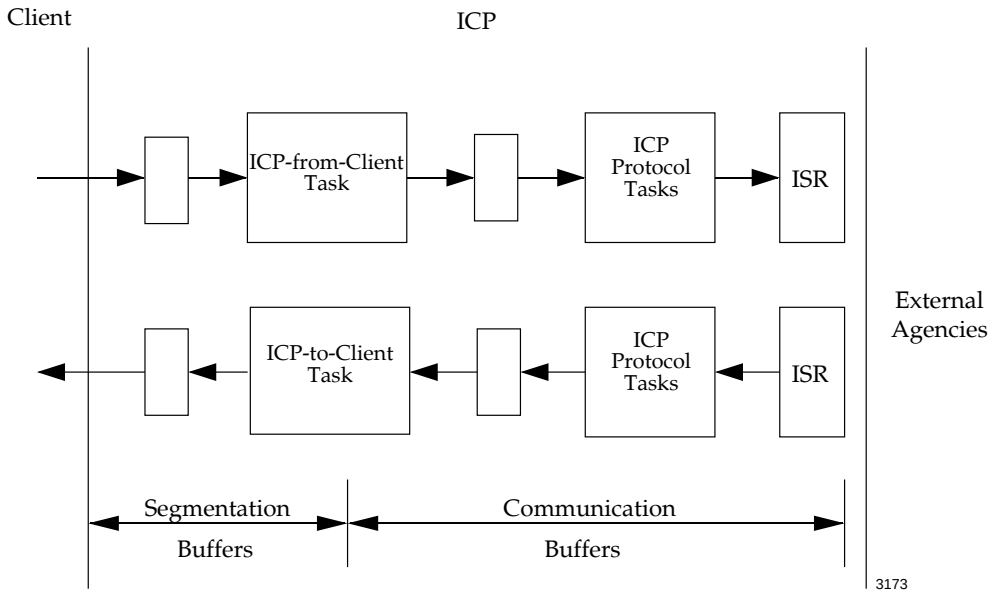

Figure 1–4: Segmentation and Communication Buffer Usage

1.7.1 Client Configuration of the Military/Government Protocols Environment

After the Military/Government Protocols software image is downloaded to the ICP (as described in the *Freeway User Guide* or the appropriate user's guide for your embedded ICP), the client application can configure the Military/Government Protocols software environment with the following features:

- One required General Session for the ICP as a whole, plus optional link-specific sessions.
- Segmentation buffers that contain up to 8128 bytes of data

- Communication buffers that contain up to 2048 bytes of data
- Modem signal monitoring
- Time stamping of received data messages
- Periodicity of flushing to client of accumulated receive data
- Link-by-link protocol selection
- Custom link configuration options appropriate to the selected protocol (data rate, clocking source, receive message size, etc.)

For protocol-specific operational and link-configuration details, refer to the individual protocol programmer's guides (listed in the *Protogate References* section on page 13).

1.7.2 Client Control

After client configuration of the environment is performed, the client can control the link activity. The client application uses messages to perform actions such as the following:

- Enable or disable a link for data exchange
- Provide transmit data messages
- Request link statistics, link status, and other reports

The ICP generates messages in response to client commands, such as the following:

- Confirmations of client commands
- Requested reports

The ICP generates messages to notify the client application of changes in the ICP environment. These messages contain information such as the following:

- Received data
- Modem signal changes
- Loss of receive data
- Statistics counter overflow
- Transmit and receive clocking signal status
- Receive data bit-stream anomalies (such as idle pattern loss)

The client application uses the DLI sessions to exchange messages with the ICP. Military/Government Protocols general message formats are described in Chapter 4 (for synchronous protocols) and in Chapter 5 (for asynchronous protocols). For protocol-specific message details, refer to the individual protocol programmer's guides (listed in the *Protogate References* section on page 13).

Chapter

2 DLI Concepts for the Military/Government Protocols Environment

Note

The term "Freeway" can mean either a Freeway server or an embedded ICP. For the embedded ICP, also refer to the user's guide for your ICP and operating system (for example, the *ICP2432 User Guide for Windows NT*).

This chapter first considers general data link interface (DLI) concepts and then addresses the specific DLI details affecting Military/Government Protocols.

2.1 Summary of DLI Concepts

The DLI presents a consistent, high-level, common interface across multiple clients, operating systems, and transport services. It implements functions that permit your application to use data link services to access, configure, establish and terminate sessions, and transfer data across multiple data link protocols. The DLI concepts are described in detail in the *Freeway Data Link Interface Reference Guide*. This section summarizes the basic information.

2.1.1 Configuration in the Freeway Environment

Several types of configuration affect how a client application runs in the Freeway environment:

- Freeway server configuration
- data link interface (DLI) session configuration

- transport subsystem interface (TSI) connection configuration
- protocol-specific ICP link configuration

The Freeway server is normally configured only once, during the installation procedures described in the *Freeway User Guide*. DLI session and TSI connection configurations are defined by specifying parameters in DLI and TSI ASCII configuration files (Section 2.2.1 gives an example DLI ASCII configuration file). You must then run two preprocessor programs, dlicfg and tsicfg, to create binary configuration files. For a description of the configuration process, refer to Appendix B of this document, as well as the *Freeway Data Link Interface Reference Guide* and the *Freeway Transport Subsystem Interface Reference Guide*. You must perform ICP link configuration within the client application (described in Section 4.2.9 on page 60 or Section 5.2.2 on page 85).

2.1.2 Non-blocking I/O

Note

Earlier Freeway releases used the term "asynchronous" for non-blocking I/O. Some parameter names (for example, the asyncIO DLI configuration parameter) reflect the previous terminology.

Non-blocking I/O applications are useful when doing I/O to multiple channels with a single process where it is not possible to "block" on any one channel waiting for I/O completion.

In the Freeway environment, the term non-blocking I/O indicates that the dlOpen, dlClose, dlRead and dlWrite functions might return after the I/O has been queued at the client, but before the transfer to Freeway is complete. The client must handle I/O completions at the software interrupt level in the completion handler established by the dlInit or dlOpen function, or by periodic use of dlPoll to query the I/O completion status.

For an application session to use non-blocking I/O, the asyncIO DLI configuration parameter must be set to "yes" (the default is "no"). The alwaysQIO DLI configuration

parameter further qualifies the operation of non-blocking I/O activity. Refer to the *Freeway Data Link Interface Reference Guide* for more information.

2.1.3 Buffer Management

The default interrelated Freeway, DLI, and TSI buffer sizes correspond to the default ICP segmentation buffer data area size of 2048 bytes. This translates into the requirement of a maxBufSize TSI configuration parameter value of at least 2160 bytes in the client and server TSI configuration files. This is because room must also be provided for the segmentation buffer header (8 bytes; see Section 4.1.1 on page 50), the segmentation buffer system parameters area (28 bytes; see Section 4.3.14 on page 73), and a Freeway DLI header (76 bytes). The delivered Military/Government Protocols configuration files specify 2200 bytes.

Caution

If you need to change a buffer size for your application, refer to the *Freeway Data Link Interface Reference Guide* for explanations of the factors that you must consider.

2.2 Using the DLI in the Military/Government Protocols Environment

In the Freeway system, the client addresses Freeway sessions through the DLI. All DLI requests in the Military/Government Protocols client application must use the DLI *Raw* operation, which is discussed in detail in the *Freeway Data Link Interface Reference Guide*.

2.2.1 Initializing the DLI

The client application calls dlInit to initialize its interface to Freeway. This call specifies a DLI binary configuration file, which is generated off-line from a text file (see Appendix B for details of the configuration process). The text file contains definitions of the sessions that can be opened, as described in the following Section 2.2.2.

Since *Raw* operation does not perform automatic link configuration, no protocol-specific link configuration parameters are specified in the DLI configuration file.

2.2.2 Military/Government Protocols DLI Session Configuration

The DLI text configuration file consists of the following sections (the parameters are described in the *Freeway Data Link Interface Reference Guide*):

- A "main" section which specifies the DLI configuration for non-session-specific operations
- One section for each of the ICPs, defining its General Session.
- One or more additional sections, each specifying a session associated with a particular serial communication link (port). Each link can be configured independently of the other links.

For a Freeway server, each session has an associated TSI connection name (the transport parameter) which you also must specify in your TSI configuration file, though multiple

sessions can use the same TSI connection. Embedded ICPs do not use a TSI configuration file.

Figure 2–1 (for a Freeway server) and Figure 2–2 (for an embedded ICP) show an example DLI configuration file defining the "main" section, the General Session, and two Military/Government Protocols sessions. You need to include only those session parameters whose values differ from the defaults. For Military/Government Protocols sessions, the required values which are different from the default values are:

- alwaysQIO = "yes"
- asyncIO = "yes"
- cfgLink = "no"
- enable = "no"
- localAck = "no"
- protocol = "raw"

Although the Military/Government Protocols link numbering is 1-based (employing the range of 1–*n*, where *n* is the number of physical ports), the DLI link designations are 0-based, employing the range 0–(*n*-1). For example, the DLI configuration for a session must specify "portNo=0" for link 1. DLI configuration of ICP numbers also is 0-based.

2.2.3 Opening and Attaching DLI Sessions

The client application calls dlOpen to open a required session to an ICP, called the General Session, in order to communicate with that ICP. The General Session is not link-specific. It is intended as the means of configuring and controlling the ICP as a whole. It is also the default session for sending all messages to the client. However, the client normally opens additional sessions to specific links. A link-specific session is mapped to a single link. Each link is allowed to have only one link-specific session mapped to it, although messages for any link can continue to be transferred either way through the General Session. The General Session is indicated in client commands to the ICP by specifying 0 in place of a link number.

```
main                                    // DLI "main" section:                  //
{
      asyncIO = "yes";                  // Use non-blocking I/O                 //
      tsiCfgName = "miltcfg.bin";       // TSI binary config file               //
}
ICP0General                             // General Session name:                //
{                                       // Client-related parameters:           //
      alwaysQIO = "yes";                //      Queue I/Os even if complete     //
      asyncIO = "yes";                  //      Use non-blocking I/O            //
      cfgLink = "no";                   //      Client configures links         //
      enable = "no";                    //      Client enables links            //
      localAck = "no";                  //      Client processes transmit ack   //
      boardNo = 0;                      //      First ICP is zero               //
      portNo = 0;                       //      First ICP link is zero          //
      protocol = "raw";                 //      Use raw operation               //
      transport = "ConnG";              //      TSI connection name specified   //
                                        //         in TSI configuration file    //
}
ICP0link0                               // First session name:                  //
{                                       // Client-related parameters:           //
      alwaysQIO = "yes";                //      Queue I/Os even if complete     //
      asyncIO = "yes";                  //      Use non-blocking I/O            //
      cfgLink = "no";                   //      Client configures links         //
      enable = "no";                    //      Client enables links            //
      localAck = "no";                  //      Client processes transmit ack   //
      boardNo = 0;                      //      First ICP is zero               //
      portNo = 0;                       //      First ICP link is zero          //
      protocol = "raw";                 //      Use raw operation               //
      transport = "Conn0";              //      TSI connection name specified   //
                                        //         in TSI configuration file    //
}
ICP0link1                               // Second session name:                 //
{                                       // Client-related parameters:           //
      alwaysQIO = "yes";                //      Queue I/Os even if complete     //
      asyncIO = "yes";                  //      Use non-blocking I/O            //
      cfgLink = "no";                   //      Client configures links         //
      enable = "no";                    //      Client enables links            //
      localAck = "no";                  //      Client processes transmit ack   //
      boardNo = 0;                      //      First ICP is zero               //
      portNo = 1;                       //      Second ICP link is one          //
      protocol = "raw";                 //      Use raw operation               //
      transport = "Conn0";              //      TSI connection name specified   //
                                        //         in TSI configuration file    //
}
```

Figure 2–1: Example DLI Configuration File for Two Freeway Server Links

```
main                                    // DLI "main" section:                  //
{
    asyncIO = "yes";                    // Use non-blocking I/O                 //
    tsiCfgName = "."                    // Location of NT log/trace svc    //
                                        //     (tsiCfgName unused for VMS)  //
    maxBuffers = 1024;                  // Maximum number of buffers       //
    maxBufSize = 2200;      // Allows 2048 ICP data area, plus 8 extra  //
}
ICP0General                             // General Session name:                //
{                                       // Client-related parameters:           //
    alwaysQIO = "yes";                  //     Queue I/Os even if complete      //
    asyncIO = "yes";                    //     Use non-blocking I/O             //
    cfgLink = "no";                     //     Client configures links          //
    enable = "no";                      //     Client enables links             //
    localAck = "no";                    //     Client processes transmit ack    //
    boardNo = 0;                        //     First ICP is zero                //
    portNo = 0;                         //     First ICP link is zero           //
    protocol = "raw";                   //     Use raw operation                //
    maxBufSize = 2200;      // Allows 2048 ICP data area, plus 8 extra  //
}
ICP0link0                               // First session name:                  //
{                                       // Client-related parameters:           //
    alwaysQIO = "yes";                  //     Queue I/Os even if complete      //
    asyncIO = "yes";                    //     Use non-blocking I/O             //
    cfgLink = "no";                     //     Client configures links          //
    enable = "no";                      //     Client enables links             //
    localAck = "no";                    //     Client processes transmit ack    //
    boardNo = 0;                        //     First ICP is zero                //
    portNo = 0;                         //     First ICP link is zero           //
    protocol = "raw";                   //     Use raw operation                //
    maxBufSize = 2200;      // Allows 2048 ICP data area, plus 8 extra  //
}
ICP0link1                               // Second session name:                 //
{                                       // Client-related parameters:           //
    alwaysQIO = "yes";                  //     Queue I/Os even if complete      //
    asyncIO = "yes";                    //     Use non-blocking I/O             //
    cfgLink = "no";                     //     Client configures links          //
    enable = "no";                      //     Client enables links             //
    localAck = "no";                    //     Client processes transmit ack    //
    boardNo = 0;                        //     First ICP is zero                //
    portNo = 1;                         //     Second ICP link is one           //
    protocol = "raw";                   //     Use raw operation                //
    maxBufSize = 2200;      // Allows 2048 ICP data area, plus 8 extra  //
}
```

Figure 2–2: Example DLI Configuration File for Two Embedded ICP Links

Using *Raw* operation, the client's dlOpen call to open a session allows only the client DLI and Freeway to handle that session. A separate step is required to allow the ICP to handle the session. To accomplish this, the client sends a DLI Attach command in the first dlWrite issued to the session. This informs the ICP's Freeway interface of the session, and returns the ICP session ID to be used to communicate with that session. After a session is opened and attached, writes to the ICP through the session reach the Military/Government Protocols software itself, and the Military/Government Protocols software can write to the client through the session. The ICP's Freeway interface needs only the ICP session ID to route the message to the intended session and to send responses associated with that session to the client.

When using *Raw* operation, the client application must employ the DLI optional arguments data structure (Section 3.2.1 on page 45) to issue dlWrite commands to a session. However, the Military/Government Protocols system requires only a subset of the optional arguments fields (see Table 3–3 on page 46). These are limited to the fields which direct the command through the server and to the correct ICP. The optional arguments structure is also supplied to the client by completed dlRead requests (see Table 3–4 on page 47).

All other data needed by the ICP or by the client is contained in the message buffer itself. The link number, function code, error status, and data size are in the message header, and the data is in the message data area.

2.2.4 Detaching and Closing DLI Sessions

To close a session, the client application first must detach it from the ICP. To accomplish this, the client sends a DLI Detach command using a dlWrite issued to the session. This informs the ICP's Freeway interface that the client is ending the session. When the ICP's Detach response is received in the final dlRead for the session, the client calls dlClose to end the session within the DLI and Freeway. The dlRead optional arguments are shown in Table 3–3 on page 46.

2.2.5 Error Reporting

In the Freeway Military/Government Protocols environment, errors related to the DLI functions are reported in two ways:

- Errors can be returned directly by the DLI function call. Typical errors are those described in the *Freeway Data Link Interface Reference Guide.*

- Errors generated by the Military/Government Protocols software are *not* reported using the DLI optional arguments structure. Instead, there are error status fields in the header associated with each Military/Government Protocols message. These fields are described in detail in Section 4.1.1.4 on page 50 and Section 4.1.1.6 on page 52 for synchronous protocols, or Table 5–1 on page 84 for asynchronous protocols.

Chapter

3 Military/Government Protocols DLI Functions

Note

For convenience, the term "Freeway" here can mean either a Freeway server or an embedded ICP. The client-system interfaces to the two are nearly identical. For the embedded ICP, also refer to the user's guide for your ICP and operating system (for example, the *ICP2432 User Guide for Windows NT*).

This chapter describes how to use the data link interface (DLI) functions to write client applications interfacing to the Freeway Military/Government Protocols software. You should be familiar with the concepts described in the *Freeway Data Link Interface Reference Guide* (summary information was provided in Chapter 2).

The following might be helpful references while reading this chapter:

- Table 3–1 lists a typical sequence of DLI function calls using non-blocking I/O
- The *Freeway Data Link Interface Reference Guide* gives complete DLI error code descriptions.
- The *Freeway Data Link Interface Reference Guide* provides a generic code example which can guide your application program development, along with the program described in Appendix A of this manual.

3.1 Example Military/Government Protocols Call Sequences

Table 3–1 shows the sequence of DLI function calls to send and receive data using non-blocking I/O example. The remainder of this chapter and the *Freeway Data Link Interface Reference Guide* give further information about each function call. Section 2.1.2 on page 32 describes non-blocking I/O.

Table 3–1: DLI Call Sequence for Military/Government Protocols (Non-blocking I/O)

1. Call dlInit to initialize the DLI operating environment.
   The first parameter is your DLI binary configuration file name.
2. Call dlOpen to get a session ID for the Military/Government Protocols General Session.
3. Call dlOpen for link-specific session ID (optional).
4. Call dlPoll to confirm the success of each session ID obtained in Step 2 and Step 3.
5. Call dlBufAlloc for all required input and output buffers.
6. Call dlRead to queue the initial read request.
7. Call dlWrite[a] to attach each session from Step 2 and Step 3 to the ICP. (Section 2.2.3 on page 35)
8. Call dlWrite[a] to configure the Military/Government Protocols environment (Section 1.7.1 on page 28).
9. Call dlWrite[a] to configure and enable the ICP links (Section 4.2.8 on page 58/Section 4.2.9 on page 60 or Section 5.2.2 on page 85/Section 5.2.3 on page 86).
10. Call dlWrite to send requests and data to Freeway (Section 4.2 on page 53 or Section 5.2 on page 85).
11. Call dlRead to queue reads to receive responses and data from Freeway (Section 4.3 on page 66 or Section 5.3 on page 88).
12. As I/Os complete and the I/O completion handler is invoked, call dlPoll to confirm the success of each dlWrite and to accept the data from each dlRead.
13. Repeat Step 10 through Step 12 until you are finished writing and reading.
14. Call dlWrite[a] to disable the ICP links (Section 4.2.10 on page 60 or Section 5.2.4 on page 86).
15. Call dlWrite[a] to detach each session from Step 7 (Section 2.2.4 on page 38).
16. Call dlBufFree for all buffers allocated in Step 5.
17. Call dlClose for each session ID obtained in Step 2 and Step 3.
18. Call dlPoll to confirm that each session was closed in Step 17.
19. Call dlTerm to terminate your application's access to Freeway.

[a] After this dlWrite call, wait for the proper response to arrive using calls to dlRead/dlPoll before continuing (see Step 12).

3.2 Overview of DLI Functions for Military/Government Protocols

After the Military/Government Protocols software is downloaded to the Freeway ICP, the client and Freeway can communicate by exchanging messages. These messages configure and activate each ICP link and transfer data. The client application issues reads and writes to transfer messages to and from the ICP.

This section summarizes the DLI functions used in writing a client application. The simplest view of using the DLI functions is:

- Start up communications (dlInit, dlOpen, dlBufAlloc)
- Send requests and data using dlWrite
- Receive responses using dlRead
- For blocking I/O, use dlSyncSelect to query read availability status for multiple sessions
- For non-blocking I/O, handle I/O completions at the software interrupt level in the completion handler established by the dlInit or dlOpen function, or by periodic use of dlPoll to query the I/O completion status
- If necessary, reset and download the protocol software to the ICP using dlControl
- Shut down communications (dlBufFree, dlClose, dlTerm)

Table 3–2 summarizes the DLI function syntax and parameters, listed in the most likely calling order. Refer to the *Freeway Data Link Interface Reference Guide* for details.

Table 3–2: DLI Functions: Syntax and Parameters (Listed in Typical Call Order)

DLI Function	Parameter(s)	Parameter Usage
int dlInit	(char *cfgFile, char *pUsrCb, int (*fUsrIOCH)(char *pUsrCb));	DLI binary configuration file name Optional I/O complete control block Optional IOCH and parameter
int dlOpen[a]	(char *cSessionName, int (*fUsrIOCH) (char *pUsrCB, int iSessionID));	Session name in DLI config file Optional I/O completion handler Parameters for IOCH
int dlPoll	(int iSessionID, int iPollType, char **ppBuf, int *piBufLen, char *pStat, DLI_OPT_ARGS **ppOptArgs);	Session ID from dlOpen Request type Poll type dependent buffer Size of I/O buffer (bytes) Status or configuration buffer Optional arguments
int dlpErrString	(int dlErrNo);	DLI error number (global variable dlerrno)
char *dlBufAlloc	(int iBufLen);	Minimum buffer size
int dlRead	(int iSessionID, char **ppBuf, int iBufLen, DLI_OPT_ARGS *pOptArgs);	Session ID from dlOpen Buffer to receive data Maximum bytes to be returned Optional arguments structure
int dlWrite	(int iSessionID, char *pBuf, int iBufLen, int iWritePriority, DLI_OPT_ARGS *pOptArgs);	Session ID from dlOpen Source buffer for write Number of bytes to write Write priority (normal or expedite) Optional arguments structure
int dlSyncSelect	(int iNbrSessID, int sessIDArray[], int readStatArray[]);	Number of session IDs Packed array of session IDs Array containing read status for IDs
char *dlBufFree	(char *pBuf);	Buffer to return to pool
int dlClose	(int iSessionID, int iCloseMode);	Session ID from dlOpen Mode (normal or force)
int dlTerm	(void);	
int dlControl	(char *cSessionName, int iCommand, int (*fUsrIOCH) (char *pUsrCB, int iSessionID));	Session name in DLI config file Command (*e.g.* reset/download) Optional I/O completion handler Parameters for IOCH

[a] It is critical for the client application to receive the dlOpen completion status before making any other DLI requests; otherwise, subsequent requests will fail. After the dlOpen completion, however, you do not have to maintain a one-to-one correspondence between DLI requests and dlRead requests.

3.2.1 DLI Optional Arguments

Section 4.2 and Section 4.3 describe the dlWrite and dlRead functions for a Military/Government Protocols application. Both functions use the optional arguments parameter to provide the protocol-specific information required for *Raw* operation. The "C" definition of the optional arguments structure is shown in Figure 3–1.

```
typedef struct          _DLI_OPT_ARGS
{
    unsigned short      usFWPacketType;
    unsigned short      usFWCommand;
    unsigned short      usFWStatus;
    unsigned short      usICPClientID;
    unsigned short      usICPServerID;
    unsigned short      usICPCommand;
    short               iICPStatus;
    unsigned short      usICPParms[3];
    unsigned short      usProtCommand;
    short               iProtModifier;
    unsigned short      usProtLinkID;
    unsigned short      usProtCircuitID;
    unsigned short      usProtSessionID;
    unsigned short      usProtSequence;
    unsigned short      usProtXParms[2];
}   DLI_OPT_ARGS;
```

Figure 3–1: "C" Definition of DLI Optional Arguments Structure

Section 2.2 on page 34 describes using the DLI in the Military/Government Protocols environment. The required dlWrite optional arguments fields for a Military/Government Protocols client application are shown in Table 3–3. The relevant Military/Government Protocols dlRead optional arguments are shown in Table 3–4.

Table 3–3: Required dlWrite Optional Arguments Fields

dlWrite Optional Arguments Field	Value	Usage
usFWPacketType	FW_DATA	
usFWCommand	FW_ICP_WRITE	
usICPCommand	DLI_ICP_CMD_ATTACH DLI_ICP_CMD_WRITE DLI_ICP_CMD_DETACH	Send an Attach request for a session Write a message buffer to a session Send a Detach request for a session
usProtLinkID	Attach: ICP link number associated with session (1–8 or 1–16 for physical links, 0 for General Session)	
	Write and Detach: (unused)	
usProtSessionID	Attach: (unused) Write and Detach: ICP session ID	

Table 3–4: Relevant dlRead Optional Arguments Fields

dlRead Optional Arguments Field	Value	Usage
usFWPacketType	FW_DATA	
usFWCommand	FW_ICP_READ	
usICPCommand	DLI_ICP_CMD_ATTACH DLI_ICP_CMD_READ DLI_ICP_CMD_DETACH	Acknowledgment of Attach Message buffer from session Acknowledgment of Detach
iICPStatus	Attach: DLI_ICP_ERR_NO_ERR DLI_ICP_ERR_BAD_NODE DLI_ICP_ERR_BAD_LINK DLI_ICP_ERR_BAD_SESSID DLI_ICP_ERR_ALREADY_ATTACHED Read: Always DLI_ICP_ERR_NO_ERR Detach: DLI_ICP_ERR_NO_ERR DLI_ICP_ERR_BAD_SESSID	Status return for Attach Status return for Read Status return for Detach
usProtSessionID	Attach: ICP session ID Read and Detach: (unused)	

Chapter

4 Synchronous Protocol Messages

4.1 Synchronous Protocol Message Formats

After the Military/Government Protocols software is downloaded to the ICP, the client and ICP can communicate by exchanging messages. Military/Government Protocols messages are transmitted and received using *Raw* dlWrite and dlRead requests, as described in Section 3.2 on page 43. The message data area might contain link configuration options, status information, or actual data to or from the serial link. Figure 4–1 shows the format of a synchronous protocol message.

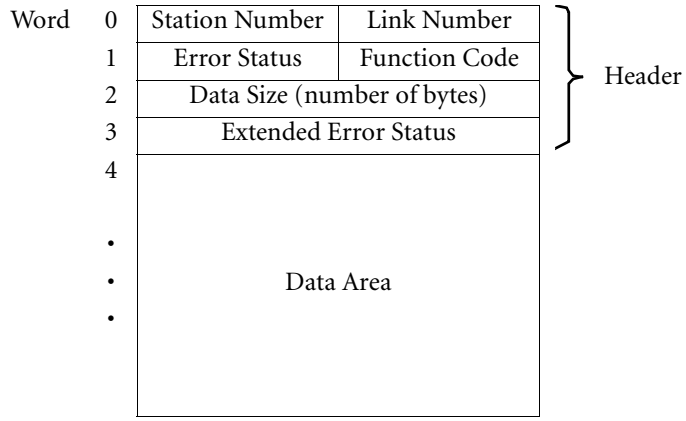

Figure 4–1: Client Synchronous Protocol Message Format

Note

In Figure 4–1, the lower byte addresses are depicted on the right, and the higher byte addresses on the left.

4.1.1 Header Format

All messages sent to and received from the ICP begin with a standard header. The output data block and input data block messages can contain multiple data messages within the data area. Each data message begins with a header as shown in Figure 4–1.

4.1.1.1 Link Number Field (Byte 0)

This 8-bit field contains the ICP serial link number associated with the message. The Request Number of Links command (Section 4.2.18 on page 64) can be used to determine the number of serial links on the ICP. The links are numbered from 1–*n*. (Note, however, that the DLI link configuration file port designations are 0-based. For example, the DLI configuration for a session must specify "portNo=0" for link 1.)

4.1.1.2 Station Number Field (Byte 1)

This 8-bit field contains a station number, for those protocols that define stations that share a link. Station numbering conventions are protocol-dependent.

4.1.1.3 Function Code Field (Byte 2)

This 8-bit field contains the code for the type of message that the client is sending to or receiving from the ICP.

4.1.1.4 Error Status Field (Byte 3)

This 8-bit field contains the error status returned to the client when the ICP detects an error in a received message on a link, or rejects a command received from the client. Figure 4–2 shows the errors reported in this field and Table 4–1 describes the error bits.

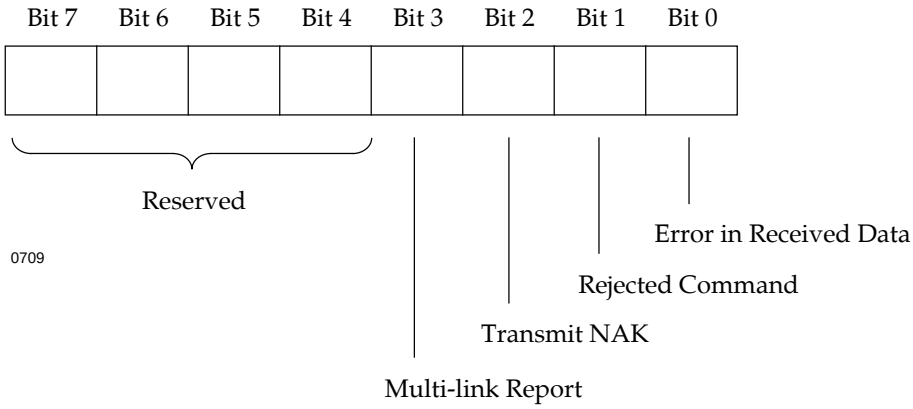

Figure 4–2: Format of Error Status Field

Table 4–1: Meanings of Error Bits in Error Status Field

Bit	Error Type	Meaning
Bit 0	Error in Received Data	This error is reported when there is an error in a message received from the ICP. Bits set in the Extended Error Status field denote the type of error (or errors) in the message.
Bit 1	Rejected Command	This error signifies that the command received from the client was invalid. When the client program sends an illegal command (such as an invalid function code or something similar), the ICP returns the illegal command to the client with the rejected bit set in the error status field.
Bit 2	Transmit NAK	If the Transmit Acknowledge configuration option is selected for a link and an unrecoverable transmit error occurs (such as transmit expiration), the Transmit NAK bit is set in the error status field of the corresponding Transmit Acknowledge response (Section 4.3.19 on page 79).

4.1.1.5 Data Size Field (Word 2)

This 16-bit field contains the number of bytes of data that follow the eight bytes of the message header. On input data messages, this number includes the four bytes of the Time Stamp field.

4.1.1.6 Extended Error Status Field (Word 3)

This 16-bit field contains extended definitions of the "Error in Received Data" bit (bit 0) in the Error Status field. Receive errors are indicated by the bits of this word. The meanings of the bits are protocol-dependent. Refer to your individual protocol programmer's guides for details (page 14 shows a list of guides).

4.1.2 Data Area (beginning at Byte 9 / Word 4)

When the client sends a message to the ICP, the data area can contain link configuration data, data values for commands, or data to be transmitted on a serial link. When the ICP sends a message to the client, the data area can contain data values for responses, link status, link statistics, or data received from a serial link. The ICP protocol tasks convert data back and forth between the serial link format and the client format, as described in your individual protocol programmer's guides (page 14 shows a list of guides). The data is padded with trailing zeroes to reach the final word boundary.

4.2 Overview of Requests using Raw dlWrite

Section 4.2.1 through Section 4.2.19 describe the dlWrite requests (commands, report requests, and data to be transmitted) that the client application can send to the ICP. The dlWrite requests can be transferred via the General Session or via an optional link-specific session, as described in Section 2.2.3 on page 35. Table 4–2 lists the requests by function code number, as found in the Function Code field of the message header.

Table 4–2: Synchronous Protocol Messages Originating in the Client

Code	Name	Reference
1	Configure Segmentation Buffer	Section 4.2.1 on page 54
2	Configure Communication Buffer	Section 4.2.2 on page 55
3	Set Time Stamping	Section 4.2.3 on page 56
4	Set Signal Monitor Interval	Section 4.2.4 on page 57
5	Set Blocking Interval	Section 4.2.5 on page 57
6	(reserved)	
7	Set Link Protocol	Section 4.2.7 on page 58
8	Configure Link	Section 4.2.8 on page 58
9	Enable Link	Section 4.2.9 on page 60
10	Disable Link	Section 4.2.10 on page 60
11	Request Link Status Report	Section 4.2.11 on page 61
12	Request Link Statistics Report	Section 4.2.12 on page 61
13	Request Queued Output Message Count	Section 4.2.13 on page 61
14	Output Data Block	Section 4.2.14 on page 62
15	Set Time Stamp Value	Section 4.2.15 on page 64
16	(protocol-specific command)	Section 4.2.16 on page 64
17	Request Version ID	Section 4.2.17 on page 64
18	Request ICP Links Report	Section 4.2.18 on page 64
19	Request Buffer Status Report	Section 4.2.19 on page 64
20	(reserved)	

4.2.1 Configure Segmentation Buffer [1]

This command initializes the ICP segmentation buffer space. Set the message header data size field to 2. Set word zero of the message data area to the size (in bytes) for the segmentation buffer data area. Section 4.3.1 on page 66 describes the confirmation response.

A default value of 2048 is set at ICP startup. Typically, however, a much smaller value is more convenient, since it results in the availability of many more segmentation buffers. 512 bytes is safely larger than the maximum needed for anything aside from blocked input or output data, so unless the user wishes to block input or output messages at greater length, it is suggested that 512 bytes be specified.

If this command is issued, it must be one of the first eight DLI_ICP_CMD_WRITE commands following the downloading of the ICP. Thereafter the segmentation buffer sizing in force at that point cannot be changed without re-downloading. Thus any buffer resizing must be done within that initial window of opportunity. If the attempt is made later, the command is rejected by the ICP. The segmentation buffers are used for all messages (client commands, ICP responses, and ICP notifications) conveyed on the client/ICP interface. The ICP creates as many segmentation buffers as will fit in the ICP's segmentation buffer memory partition. The segmentation buffer size should always be an even value, with a maximum of 8128.

Caution

If the ICP segmentation buffer sizes are to be raised above the default value of 2048 bytes, the client must take care that the maxBufSize parameter in both the client TSI configuration file and the server TSI configuration file is adequate. See Section 2.1.3 on page 33.

The data area of the segmentation buffers must be large enough to accommodate the desired maximum number of blocked data messages. For example, if the protocol spec-

ifies 14 information bytes for both the client-to-ICP and ICP-to-client formats, and the segmentation buffer size is set to 512, twenty-three messages can be blocked in the Output Data Block command:

(8 header bytes + 14 data bytes) * 23 = 506 bytes

Similarly, a maximum of nineteen messages can be blocked in the Input Data Block ICP response:

(8 header bytes + 4 time stamp bytes + 14 data bytes) * 19 = 494 bytes

The ICP will reject this message if:

- the initial window of opportunity to process this command has closed;
- the specified data size is too small to hold at least one communication buffer as they are currently configured;
- the specified data size is greater than 8192 bytes;
- periodic or threshold Buffer Status Reports have been requested;
- any ports are enabled.

4.2.2 Configure Communication Buffer [2]

This command initializes the ICP communication buffer space. Set the message header data size field to 2. Set word zero of the message data area to the size (in bytes) for the communication buffer data area. Section 4.3.2 on page 66 describes the confirmation response. A default value of 64 is set at ICP startup.

If this command is issued, it must be one of the first eight DLI_ICP_CMD_WRITE commands following the downloading of the ICP. Thereafter the communication buffer sizing in force at that point cannot be changed without re-downloading. Thus any buffer resizing must be done within that initial window of opportunity. If the attempt is made later, the command is rejected by the ICP.

The communication buffers are used for individual messages being processed on the communication interface and transmitted/received on the serial ports. The ICP creates as many communication buffers as will fit in the ICP's communication buffer memory partition. The communication buffer size should always be an even value, with a minimum 32 and a maximum of 2048.

The ICP will reject this message if:

- the initial window of opportunity to process this command has closed;
- the specified data size is less than 32 bytes;
- the specified data size is greater than 2048 bytes;
- the specified data size would result in a communication buffer size (counting header and time stamp) too large to fit into segmentation buffers as they are currently configured;
- any ports are enabled.

4.2.3 Set Time Stamping [3]

This command defines the mode of the ICP's time-stamping (see Appendix C). It specifies a 16-bit data value. Set the message header data size field to 2. There are two modes of time-stamping: Internal and PCI Host. Any non-negative data value in word zero of the message data area sets the ICP timing to Internal. A value of -2 sets it to PCI Host. Section 4.3.3 on page 66 describes the confirmation response. At startup the mode is set to Internal timing.

The ICP will reject this message if PCI Host timing is specified when the host-system ICP driver does not support this feature, or when a negative value other than -2 is specified.

4.2.4 Set Signal Monitor Interval [4]

This command sets the interval of the signal monitoring function. Set the message header data size field to 2. Set word zero of the message data area to the monitor interval value. The monitor interval is specified in seconds. The range allowed is 0 to 60 seconds. An interval of zero disables the monitoring process. Section 4.3.4 on page 68 describes the confirmation response. A default value of 0 is set at ICP startup.

The interface signals monitored are Clear to Send (CTS) and Data Carrier Detect (DCD) for most protocols. Data Set Ready (DSR) is monitored for some protocols. (Refer to the individual protocol programmer's guides.) Signal changes are reported to the client when the change has been present for a period greater than one monitor interval but less than two, using the Link Status Notification response (Section 4.3.15 on page 77).

The ICP will reject this message if the specified interval is greater than 60.

4.2.5 Set Blocking Interval [5]

This command sets the blocking interval for one or all links on the ICP. The blocking interval is the length of time a less-than-full input data segmentation buffer will collect received messages from a link before sending the buffer to the client. (This buffer will be sent earlier if it fills.) Set the message header data size field to 2. Set word zero of the message data area to the blocking interval value. Section 4.3.5 on page 68 describes the confirmation response.

If the link number field specifies zero, the specified value is assigned to all links. Otherwise the specified value is assigned only to the specified link. The blocking interval is specified in milliseconds and can be set only in increments of 10 milliseconds. The allowed range is from 10 to 10000. A default value of 1000 (one second) is set at ICP startup, for each link.

The ICP will reject this message if the specified interval is less than 10 or greater than 10000 (i.e., ten seconds).

4.2.6 (reserved) [6]

Command 6 is reserved.

4.2.7 Set Link Protocol [7]

This command assigns a protocol to a specified link. Set the message header data size field to 2. Set word zero of the message data area to the desired protocol selection code as specified in the individual protocol programmer's guide. Section 4.3.7 on page 68 describes the confirmation response.

If the link number field specifies zero, all links are assigned the specified protocol. Otherwise the specified protocol is assigned to the specified link. The assigning of a protocol to a link includes setting that link with the default link configuration parameters for that protocol. To assign non-default parameter values, use the Configure Link command (described next).

The ICP will reject this message if:

- the specified protocol code does not identify a protocol that is built into the Military/Protocols product being run;
- the specified link number is invalid;
- the specified link is not in a disabled state.

4.2.8 Configure Link [8]

This command sets the link configuration options for an ICP link. The data field contains a pair of 16-bit words for each specified link configuration option. The first word of the pair contains the configuration option number, and the second word contains the value for that option. The list of option/value pairs is terminated by a 16-bit terminator

word containing zero. Set the message header data size field to the number of bytes in the configuration data including the zero terminator word. Section 4.3.8 on page 68 describes the confirmation response. Figure 4–3 gives an example of the link configuration command, with two options specified. All Military/Government Protocols require that the link be in the disabled state (see Section 4.2.10 on page 60) when this command is issued. Refer to your individual protocol programmer's guide for descriptions of the options and their values.

Station Number	Link Number	0
Error Status	Function Code = 8	1
Data Size = 10		2
Extended Error Status		3
Configuration Option = 1 (Data Rate)		4
Configuration Value = 9600		5
Configuration Option = 3 (Error Screening)		6
Configuration Value = 2 (Disabled)		7
0		8

Figure 4–3: Link Configuration Block with Two Options

A successful Configure Link command solicits a Configure Link response containing a Link Configuration Report (Section 4.3.8 on page 68) as the response. This consists of a list of the configuration values for the link as they now are set (including any the parameters that were not addressed in the Configure Link command). This data is in the contains a series of option/value data pairs terminated by a 0 option code, in the format used by the Configure Link command.

It is valid to omit the data area from the Configure Link command (i.e., to specify 0 in the data size header field). In this case, no change to the link configuration is made, but the Link Configuration Report is still returned in a series of option/value data pairs. In this manner, the client program can always query the ICP for the current configuration of any link that has been assigned a protocol.

The ICP will reject this message if:

- the specified link number is invalid;
- the specified link has not been assigned a protocol;
- the specified link is enabled;
- an option code is specified that is invalid for the protocol assigned to the specified link;
- an option value is specified that is invalid for the valid option.

4.2.9 Enable Link [9]

This command enables a link. Set the message header data size field to zero. Set the link number field to the desired link number. Upon receiving this command, the ICP activates the link for transmission and reception of data according to the current configuration settings for all protocols. Certain protocols have a more complex use for the Enable Link command, as noted in certain protocol programmer's guides. Section 4.3.9 on page 69 describes the Enable Link response.

The ICP will reject this message if:

- the specified link number is invalid;
- the specified link has not been assigned a protocol;
- the specified link is already enabled.

4.2.10 Disable Link [10]

This command disables a link. Set the message header data size field to zero. Set the link number field to the desired link number. Upon receiving this command, the ICP deactivates the link for transmission and reception of data. Section 4.3.10 on page 69 describes the Disable Link response.

If any transmission acknowledgements of already-completed data transmissions are pending, they are sent to the client before the Disable Link confirmation response is sent. (Refer to Section 4.3.19 on page 79.)

The ICP will reject this message if:

- the specified link number is invalid;
- the link has not been assigned a protocol;
- the link is already disabled.

4.2.11 Request Link Status Report [11]

This command requests a Link Status report, consisting of information about the status of a specified link or all links, including the serial port modem control and clocking signal states. Set the message header data size field to zero. Set the link number field to the desired link number for a single-link report, or to zero for a separate report for each link. Section 4.3.11 on page 69 presents the format of the Link Status report. The ICP will reject this message if the specified link number is invalid.

4.2.12 Request Link Statistics Report [12]

This command requests a Link Statistics report for the specified link. Set the message header data size field to zero. Set the message header link number field to the desired link number. Section 4.3.12 on page 72 discusses the format of the Link Statistics report returned to the client. The ICP will reject this message if the specified link number is invalid.

4.2.13 Request Queue Count Report [13]

This command requests a Queue Count report, consisting of the count of messages queued for transmission to the specified link, or to all links. Set the message header data size field to zero. Set the link number field to the desired link number, or to zero to

request a Queue Count report on all links. Section 4.3.13 on page 73 discusses the format of the Queue Count report returned to the client. The ICP will reject this message if the specified link number is invalid.

Note

Protogate recommends the use of Transmit Acknowledgment notifications from the ICP (refer to Section 4.3.19 on page 79) for outgoing message flow control, rather than using Queue Count reports. The Queue Count method is considered archaic and less useful, although it is still supported for the sake of remaining compatible with older customer programs. (One advantage of Transmit Acknowledgment notifications is that they indicate expired transmit messages.)

4.2.14 Output Data Block [14]

This command supplies a data block that contains one or more embedded Output Data messages, each of which contains data to be transmitted in an individual message for the protocol assigned to the specified link. Set the message header data size field to the total number of bytes in the data area.

Each embedded, or blocked, message in the overall data area consists of a header and the data associated with it. Set its data size field to the actual number of data bytes in that data area. Set its message header link number field to the desired link number. Each blocked message must start on a word boundary. Thus for protocols that supply an odd number of data bytes to the ICP, an unused one-byte pad must be inserted; nonetheless, the data size field must specify the odd number.

The data for a blocked message is formatted by the ICP protocol software to produce the proper transmit bit-stream. This typically involves such steps as the insertion of a message start sequence, mark bits at regular intervals, a data-check value, last-byte bit padding for purposes of seamlessly merging with a possible time-fill bit pattern, etc.

Figure 4–4 illustrates the structure of an output data block. The segmentation and communication header formats are shown in Figure 4–1 on page 49 and described in Section 4.1 on page 49.

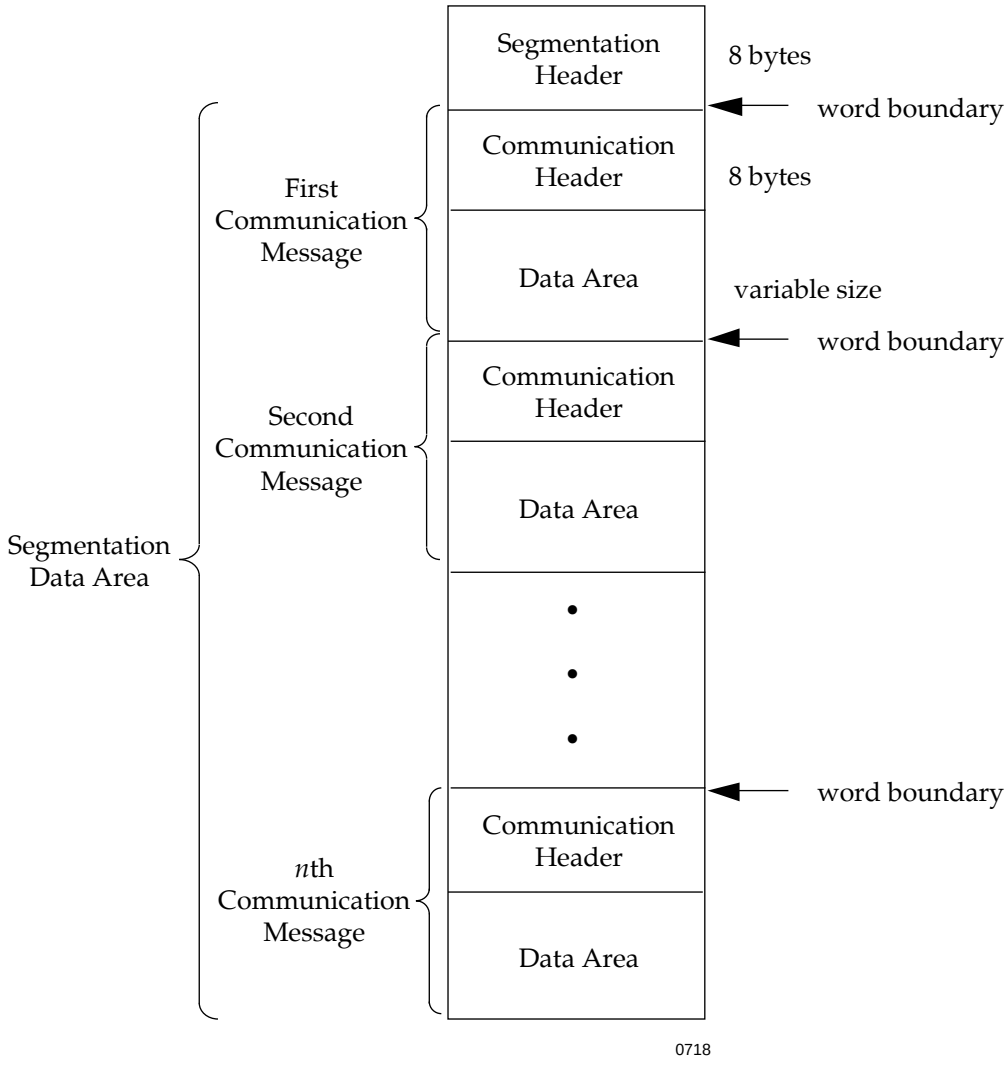

Figure 4–4: Output Data Block

4.2.15 Set Time Stamp Value [15]

This command sets the ICP's time-stamp (see Appendix C) to the specified value. Set the message header data size field to 4. Set the first longword of the data area to the 32-bit time-stamp value that is to replace the current value on the ICP. Section 4.3.16 on page 78 describes the confirmation response. This command is rejected by the ICP if a Set Time Stamping command has specified PCI Host timing.

4.2.16 Protocol-specific Command [16]

This client command (function code 16) is protocol-specific. It is described in the individual programmer's guides for those Military/Government Protocols that use it. If the individual guide does not describe a command 16, then that protocol does not use it. This command is rejected by the ICP if sent for a link that is not assigned a protocol that uses it.

4.2.17 Request Version Report [17]

This command requests a Version report for the Military/Government Protocols software that is running on the ICP. Set the message header data size field to zero. The link number field is unused. Section 4.3.20 on page 80 discusses the format of the Software Version report returned to the client.

4.2.18 Request ICP Links Report [18]

This command requests an ICP Links report, which provides the number of serial ports on the ICP. Set the message header data size field to zero. The link number field is unused. Section 4.3.21 on page 80 discusses the format of the ICP Links report returned to the client.

4.2.19 Request Buffer Status Report [19]

This command requests a Buffer Status report. Set the message header data size field to 2 or 4, depending on the number of parameters (as explained below). The link number

field is unused. Section 4.3.22 on page 80 discusses the format of the Buffer Status report returned to the client.

The Buffer Status report is a diagnostic tool available for monitoring buffer usage on the ICP and diagnosing the abnormal accumulation of buffers in particular ICP queues. The receipt of this command by the ICP causes a Buffer Status report, called a direct-response report, to be sent immediately to the client. In addition, the command data can also request that subsequent reports, called periodic reports, be generated on a specified timed basis. The command data can also request that an event-driven Buffer Status report, called an unsolicited report, be generated when either the communication buffer usage or the segmentation buffer usage rises above or falls below a specified threshold.

The command contains one or two 16-bit words of data. The first word is the requested Buffer Status report interval, in units of seconds. It controls the generation of the periodic reports. A value of zero disables periodic reports. A value of negative one asks for an immediate report without affecting the current periodicity. A positive value specifies the period to be used. The default is zero (periodic reports disabled).

The second word, if included, specifies the buffer usage threshold for the generation of unsolicited reports. It is in units of percent, and can range from zero to 99. This percentage value is applied to both the communication buffer pool and the segmentation buffer pool, and an unsolicited report is issued when this threshold is reached for either pool. After the threshold is reached and the unsolicited report is issued, another unsolicited report is issued when the usage of each pool is again below the threshold. Thus the client is informed both when buffer usage reaches the threshold and when it falls below the threshold. A zero threshold disables unsolicited reports, and a positive threshold enables them. The default is zero (unsolicited reports disabled).

4.3 Overview of Responses using Raw dlRead

Section 4.3.1 through Section 4.3.22 describe the dlRead responses (confirmations, reports, notifications, and received data) that the client application can receive from the ICP. The dlRead responses to link-specific commands are transferred via the appropriate link session, if any. Other responses are transferred via the General Session. Table 4–3 on page 67 lists the responses by function code number.

Commands sent by the client that are unable to be successfully processed by the ICP are returned as-is to the client, with the addition of the Rejected Command bit set in the message's Error Status field (refer back to Figure 4–2 on page 51). This means that the rejected command retains its original function code.

4.3.1 Configure Segmentation Buffer Response [21]

The Configure Segmentation Buffer response signifies that the Configure Segmentation Buffer command (Section 4.2.1 on page 54) was processed successfully. The message header data size field is set to 2. Word zero of the data area contains the number of buffers created in the segmentation buffer pool.

4.3.2 Configure Communication Buffer Response [22]

The Configure Communication Buffer response signifies that the Configure Communication Buffer command (Section 4.2.2 on page 55) was processed successfully. The message header data size field contains 2. Word zero of the data area contains the number of buffers created in the communication buffer pool.

4.3.3 Set Time Stamping Response [23]

The Set Time Stamp Interval response signifies that the Set Time Stamping command (Section 4.2.3 on page 56) was processed successfully. The message header data size field contains 2. Word zero of the data area contains the value specified in the Set Time Stamp Interval command.

Table 4–3: Synchronous Protocols Messages Originating in the ICP

Code	Name	Reference
21	Configure Segmentation Buffer Response	Section 4.3.1 on page 66
22	Configure communication Buffer Response	Section 4.3.2 on page 66
23	Set Time Stamping Response	Section 4.3.3 on page 66
24	Set Signal Monitor Interval Response	Section 4.3.4 on page 68
25	Set Blocking Interval Response	Section 4.3.5 on page 68
26	(reserved)	
27	Set Link Protocol Response	Section 4.3.7 on page 68
28	Configure Link Response / Link Configuration Report	Section 4.3.8 on page 68
29	Enable Link Response	Section 4.3.9 on page 69
30	Disable Link Response	Section 4.3.10 on page 69
31	Link Status Report	Section 4.3.11 on page 69
32	Link Statistics Report	Section 4.3.12 on page 72
33	Queued Output Message Count Report	Section 4.3.13 on page 73
34	Input Data Block	Section 4.3.14 on page 73
35	Link Status Notification	Section 4.3.15 on page 77
36	Set Time Stamp Value Response	Section 4.3.16 on page 78
37	(protocol-specific response)	Section 4.3.17 on page 79
38	(reserved)	
39	Transmit Acknowledgment	Section 4.3.19 on page 79
40	Version Report	Section 4.3.20 on page 80
41	ICP Links Report	Section 4.3.21 on page 80
42	Buffer Status Report	Section 4.3.22 on page 80

4.3.4 Set Signal Monitor Interval Response [24]

The Set Signal Monitor Interval response signifies that the Set Signal Monitor Interval command (Section 4.2.4 on page 57) was processed successfully. The message header data size field contains 2. Word zero of the data area contains the value specified in the Set Monitor Interval command.

4.3.5 Set Blocking Interval Response [25]

The Set Blocking Interval response signifies that the Set Blocking Interval command (Section 4.2.5 on page 57) was processed successfully. The message header data size field contains 2. Word zero of the data area contains the value specified in the Set Blocking Interval command.

4.3.6 (reserved) [26]

This response code is reserved, in conjunction with the reserving of command 6.

4.3.7 Set Link Protocol Response [27]

The Set Protocol response signifies that the Set Link Protocol command (Section 4.2.7 on page 58) was processed successfully. The message header data size field contains 2. Word zero of the data area contains the value specified in the Set Link Protocol command.

4.3.8 Configure Link Response / Link Configuration Report [28]

The Configure Link response (also called the Link Configuration Report) signifies that the Configure Link command (Section 4.2.8 on page 58) was processed successfully, and provides a Link Configuration Report. The data area consists of a complete list of the configuration values for the link as they now are set (including any parameters that were not addressed in the Configure Link command). This data consists of a series of option/value data pairs terminated by a 0 option code, in the format used by the Configure Link command to specify link configuration changes (refer to Figure 4–3 on

page 59). The message header data size field contains the number of bytes occupied by this link configuration data.

4.3.9 Enable Link Response [29]

The Enable Link response signifies that the Enable Link command (Section 4.2.9 on page 60) was processed successfully. The message header data size field is zero.

4.3.10 Disable Link Response [30]

The Disable Link response signifies that the Disable Link command (Section 4.2.10 on page 60) was processed successfully. The message header data size field is zero.

4.3.11 Link Status Report [31]

The Link Status Report is the response to a Link Status Report request (Section 4.2.11 on page 61). Four Link Status Report formats are defined: Type I, Type II, Type III, and Type A. Types I, II, and III are for synchronous protocols. They are shown in Table 4–4 on page 70, Table 4–5 on page 70, and Table 4–6 on page 71, respectively. Refer to the individual protocol programmer's guide to determine which report type applies, and if any variations on it apply. Type A is for asynchronous protocols. This format is shown in Figure 5–3 on page 90.

Link status is reported in word pairs. The first word of each pair is the code for the parameter being reported. The second word is the value of that parameter. If a Set Signal Monitor Interval command has been issued with a non-zero interval, modem signal statuses will be the value last reported by a Link Status Notification (or, if no changes have occurred, the initial status). If modem signals are not being monitored, the immediate modem signal status is reported.

The message header data size field usually contains 20 (five word pairs). But for Type I reports for certain protocols, it contains 24 (six word pairs), in order to show the Data Set Ready signal.

Table 4–4: Link Status Report (Type I)

Parameter (First word)	Description	Value (Second word)
1	Data Carrier Detect (DCD)	0 = On 1 = Off
2	Clear to Send (CTS)	0 = On 1 = Off
3	Receive clocking	0 = Clocking present 1 = Clocking lost
4	Transmit clocking	0 = Clocking present 1 = Clocking lost
5	Receive data status	0 = No data lost 1 = Receive data lost
6	Data Set Ready (DSR) (for certain protocols)	0 = On 1 = Off

Table 4–5: Link Status Report (Type II)

Parameter (First word)	Description	Value (Second word)
1	Idle Pattern	0 = Present 1 = Lost
2	Modem signals	See Figure 4–5
3	Receive clocking	0 = Clocking present 1 = Clocking lost
4	Transmit clocking	0 = Clocking present 1 = Clocking lost
5	Receive data status	0 = No data lost 1 = Receive data lost

Table 4–6: Link Status Report (Type III)

Parameter (First word)	Description	Value (Second word)
1	Receive synchronization status	Refer to your protocol-specific programmer's guide
2	Modem signals	See Figure 4–5 on page 71
3	Receive clocking	0 = Clocking present 1 = Clocking lost
4	Transmit clocking	0 = Clocking present 1 = Clocking lost
5	Receive data status	0 = No data lost 1 = Receive data lost

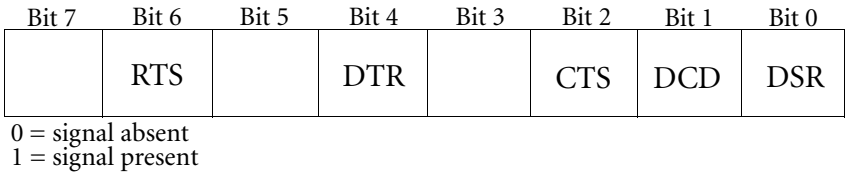

Figure 4–5: Link Status Report Type II — Word 2

4.3.12 Link Statistics Report [32]

The Link Statistics Report is the response to a Link Statistics Report request (Section 4.2.12 on page 61). The message header data size field is set to 40, and the data area contains the link statistics, as shown in Table 4–7 on page 72. If a statistics counter overflows, the client receives a Link Statistics report as an unsolicited notification (showing the overflowing count as the maximum positive 16-bit value (32,767). The statistics counters are cleared after being reported to the client; thus the client must keep running totals.

Table 4–7: Link Statistics Report

Location	Description
Byte 0	Protocol value (as assigned by Set Link Protocol command (Section 4.2.7 on page 58)
Byte 1	(reserved)
Word 1	Data rate (bits per second)
Word 2	Clocking signal source (1 = External; 2 = Internal)
Word 3	Messages received
Word 4	protocol-dependent (refer to the individual protocol programmer's guide)
Word 5	protocol-dependent (refer to the individual protocol programmer's guide)
Word 6	Receive character overruns
Word 7	Transmit character underruns
Word 8	Messages received with no errors
Word 9	Messages received with errors
Word 10	protocol-dependent (refer to the individual protocol programmer's guide)
Word 11	protocol-dependent (refer to the individual protocol programmer's guide)
Word 12	protocol-dependent (refer to the individual protocol programmer's guide)
Word 13	protocol-dependent (refer to the individual protocol programmer's guide)
Word 14	protocol-dependent (refer to the individual protocol programmer's guide)
Word 15	protocol-dependent (refer to the individual protocol programmer's guide)
Word 16	protocol-dependent (refer to the individual protocol programmer's guide)
Word 17	Messages transmitted
Word 18	Lost input messages (due to lack of available buffer for link data)
Word 19	Messages not transmitted because of expiration or other error

4.3.13 Queued Output Message Count Report [33]

The Queued Output Message Count Report is the response to the Queue Count report request (Section 4.2.13 on page 61). The message header data size field is set to 2. If the message header link field specified a specific link, the 16-bit data area of the report contains the number of messages queued for transmission on that link. If the message header link field was set to 0, the data area contains the sum of the messages queued for transmission across all the links on the ICP.

4.3.14 Input Data Block [34]

The Input Data Block notification contains one or more protocol messages formatted for the client application from the bit-streams received at the links. When received via a link session, this notification contains data from only that link. When received via the General Session, this notification may contain data from any or all of the active links. Notification is always made via the link session for a link that has its own session; links with no specific link session defined use the General Session.

An Input Data Block is sent by the ICP upon either of two events. First, when an ICP segmentation buffer allocated for the accumulation of Input Data messages becomes full or is found to have too little remaining room for the next received message). Second, when the Blocking Interval for an ICP segmentation buffer expires and the buffer has accumulated at least one message.

Figure 4–6 on page 74 illustrates the structure of an Input Data Block. The block is composed of a segmentation header, a System Parameters area, and a data area containing one or more Input Data messages.

4.3.14.1 Segmentation Header

The segmentation header format is shown in Figure 4–1 on page 49. Only the function code field and data size field are used. The data size field in the segmentation header is set to the total number of bytes in the data area (i.e., the total length of the embedded individual Input Data messages).

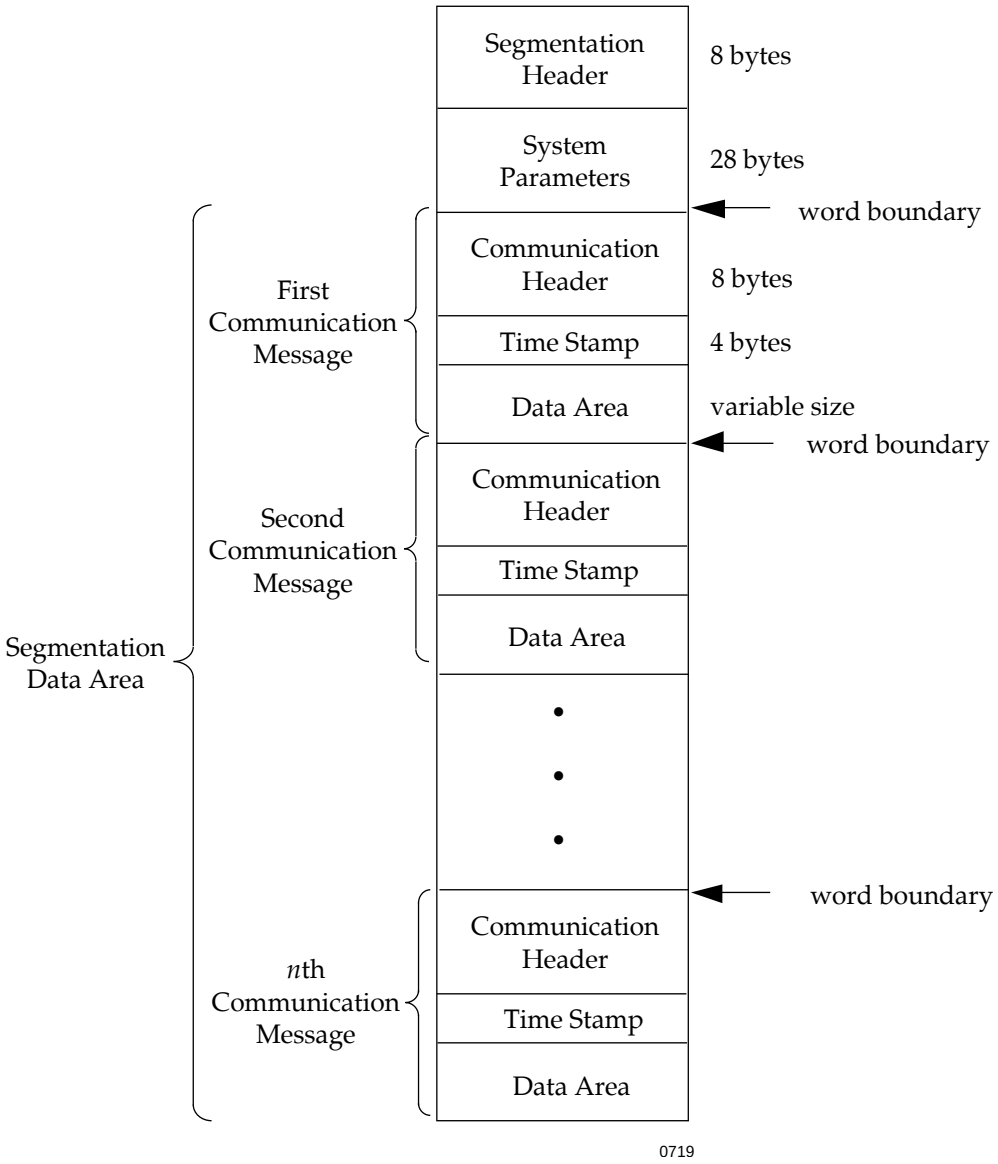

Figure 4–6: Input Data Block

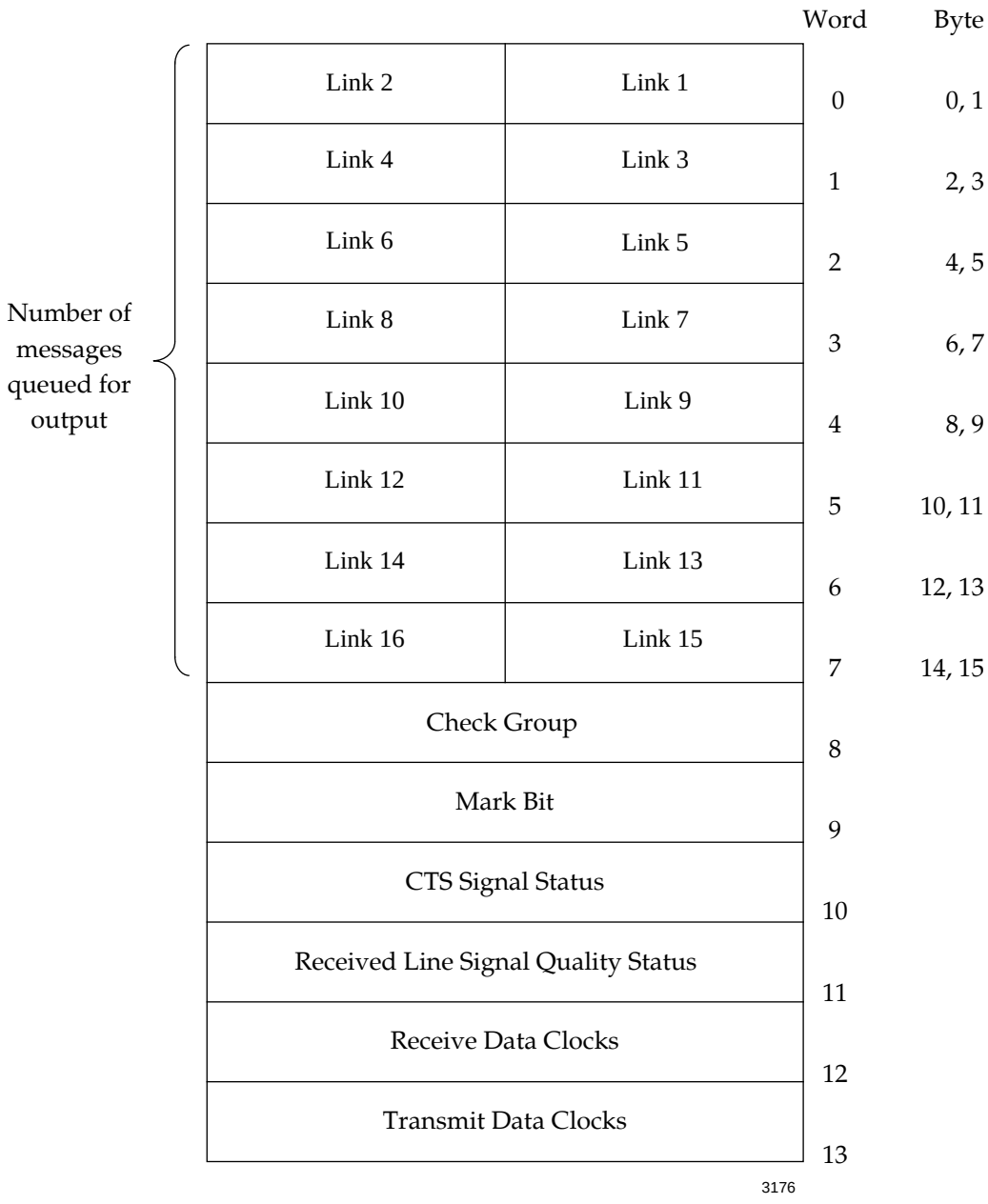

Figure 4–7: System Parameters Area

4.3.14.2 System Parameters Area

The System Parameters immediately follow the segmentation header for the input data block. These parameters report the status of the communication links on the ICP. Figure 4–7 on page 75 shows the structure of the system parameters area. The number of messages queued for output on each link is reported in byte format with a range in values of zero to 255. The following serial port signal status elements are reported using the bits in a word to identify the current status for each link:

- Clear to Send status
- Data Carrier Detect status
- Receive data clocking status
- Transmit data clocking status

These statuses are reported using bit zero to report link 1, bit 1 to report link 2, and so on. A status bit is set if the corresponding signal is off or missing. If modem signals are being monitored (i.e., a Set Signal Monitor Interval command has been issued with a non-zero interval), the modem signal status is the status last reported by a Link Status notification (or, if no changes have occurred, the initial status). If modem signals are not being monitored, the immediate status is reported. These conditions are reported for most protocols. Consult the individual protocol programmer's guides.

4.3.14.3 Data Messages

The Data Messages area contains individual Input Data messages stored in order of reception. The communication header format is shown in Figure 4–1 on page 49. Each Input Data message is composed of a communication header and data area. The link number field contains the number of the link that received the message. The error status and extended error status fields contain any error information pertaining to the associated protocol data.

The data size field contains the number of bytes in the data area, including the four bytes of time stamp; hence the data size value is four greater than the number of actual data bytes supplied for this message. For protocols with an odd number of bytes in the data area, a one-byte pad containing zero is added so that the next Input Data message an start on a word boundary; however, the data size field does not account for that pad.

The first 32-bit word in the data area of each message contains the time stamp of the message. This marks the time of arrival at the link of the first bits of the message. The remainder of the data area contains the message information received on the link. This is the client-format data, as distilled from the full bit-stream that conveyed the message on the link.

4.3.15 Link Status Notification [35]

The Link Status Notification is generated by the ICP when a notable protocol-specific event occurs for a link. The message header data size field contains 4. The first 16-bit word of the data area contains the code for the parameter being reported. The second word contains the value associated with that parameter. Table 4–8 on page 78 lists the parameters that may be reported. Modem signal changes are reported only if modem signals are being monitored (i.e., a Set Signal Monitor Interval command has been issued with a non-zero interval).

Link Status Notifications have three standard formats (called Type I, Type II, and Type III). Each Military/Government protocol uses a particular format, though several use variations not described here. Refer to the individual protocol programmer's guides for details.

Table 4–8: Link Status Notification

Parameter	Error or Status Change		
	Type I	**Type II**	**Type III**
1	Data Carrier Detect (DCD)	Idle Pattern Status	Receive Sync Status
2	Clear to Send (CTS)	Modem Signal	Modem Signal
3	Receive Clocking	Receive Clocking	Receive Clocking
4	Transmit Clocking	Transmit Clocking	Transmit Clocking
5	Receive Data Loss	Receive Data Loss	Receive Data Loss
6 7 8 9	Parameters 6 through 9 are used only in custom formats not described here.		
10	Over Speed Clocks		

Parameters 1 through 5 and their associated values are identical to the Link Status Report parameter/value pairs listed in Table 4–4 on page 70, Table 4–5 on page 70, and Table 4–6 on page 71.

Parameter 10, Over Speed Clocks, has two values. A value of one indicates that the ICP is detected clock signals in excess of the configured data rate. A value of zero indicates that the ICP is not detecting the excessive clock signals. The client may wish to avoid sending Output Data Blocks to the ICP while the over speed clocking condition persists, as the ICP may be unable to reliably transmit. This Link Status Notification response is received only from links which are assigned protocols that define the Over Speed Clock Retry Interval parameter, and which have that parameter configured.

4.3.16 Set Time Stamp Value Response [36]

The Set Time Stamp Value response signifies that the Set Time Stamp Value command (Section 4.2.15 on page 64) was processed successfully. The message header data size field contains 4. The data area contains the value specified in the Set Time Stamp Value command.

4.3.17 Protocol-specific ICP Response [37]

The protocol-specific ICP response (function code 37) signifies that the protocol-specific command 16 (Section 4.2.16 on page 64) was processed successfully. It is described in the programmer's guides for those Military/Government Protocols that use it.

4.3.18 (reserved) [38]

This ICP response/notification code is reserved.

4.3.19 Transmit Acknowledgment [39]

The Transmit Acknowledgment notification provides a flow control feedback mechanism for messages sent from the client to the ICP for transmission on the links (Section 4.2.14 on page 62). The message header data size field contains 2. Word zero of the data area contains the number of messages being acknowledged.

A Transmit Acknowledgment Threshold configuration option is provided for each link. When it is enabled, the ICP maintains a count of messages transmitted to the link but not yet acknowledged to the client. A Transmit Acknowledgement notification is sent to the client when:

- the count of completed transmissions reaches the value of the configuration option;
- a transmission is negatively acknowledged (due to transmit expiration or some other unrecoverable error);
- a Disable Link command is received from the client and any transmit completions have accumulated (sent prior to the Disable Link response).

Negative acknowledgements are indicated by the Transmit NAK flag (bit 2) of the message header Error Status byte (refer to Figure 4–2 on page 51), in which case the data value *N* positively acknowledges the first *N*-1 transmissions and negatively acknowledges the *Nth*.

4.3.20 Version Report [40]

The Version Report is the response to the Version Report Request (Section 4.2.17 on page 64). The message header data size field contains 38 (the sum of 4+33+1, as broken out below).

The first four bytes of the data area consist of a 32-bit mask, with each non-zero bit corresponding to a protocol that is present on the ICP. Protocols are numbered according to the codes used in the Set Protocol command (Section 4.2.7 on page 58). For example, if bit two is set in the mask, the protocol specified by the value 2 is present on the ICP.

The next 33 bytes of the data area contain ASCII text identifying the version of the Military/Government Protocols software source code that was used to make the product that is running on the ICP. This string includes a version ID of the form "x.y-z" and a date. The last byte of the data area contains zero, so that the version text can be treated by the client as a null-terminated string.

4.3.21 ICP Links Report [41]

The ICP Links Report is the response to the ICP Links Report request (Section 4.2.18 on page 64). The message header data size field contains 2. Word zero of the data area contains the number of serial ports on the ICP.

4.3.22 Buffer Status Report [42]

This response is the response to the Buffer Status Report request (Section 4.2.19 on page 64). The report format is shown in Table 4–9 on page 81. The message header data size field is variable, depending on the number of ICP queues being described. Words 0 and 1 of the data area give the current Buffer Status report period and threshold. Words 2–5 describe the pool of communications buffers, and words 6–9 describe the pool of segmentation buffers. Word 10 gives the number ICP queues described in Words 11–*n*, where *n* is the final word index of the report.

Table 4–9: Buffer Status Report

Word	Description
0	Buffer Status Report period (in seconds)
1	Buffer Status Report threshold (in percent)
2	Communication buffer size
3	Total number of communication buffers
4	Number of communication buffers in use
5	Percentage of communication buffers in use
6	Segmentation buffer size
7	Total number of segmentation buffers
8	Number of segmentation buffers in use
9	Percentage of segmentation buffers in use
10	Number of ICP queues listed
11	Queue #1 buffer count
12	Queue #2 buffer count
…	
10 + *n*	Queue #*n* buffer count

The Buffer Status report is generated under three conditions:

- A *direct-response report* is generated when any Buffer Status request is received from the client.
- A *periodic report* is generated when the defined Buffer Status report interval (if any) has elapsed since the issuance of the previous Buffer Status report.
- An *unsolicited report* is generated when the communication or segmentation buffer usage rises above or falls below the defined Buffer Status report threshold (if any).

Direct-response reports are sent via the session used by the request (normally the General Session). Periodic and unsolicited reports are sent through the General Session.

Chapter

5 Asynchronous Protocol Messages

5.1 Asynchronous Protocol Message Formats

Figure 5–1 shows the asynchronous protocol message format. The message headers for asynchronous protocols have the same basic format as those for synchronous protocols (four 8-bit fields followed by two 16-bit fields) as described in Chapter 4.

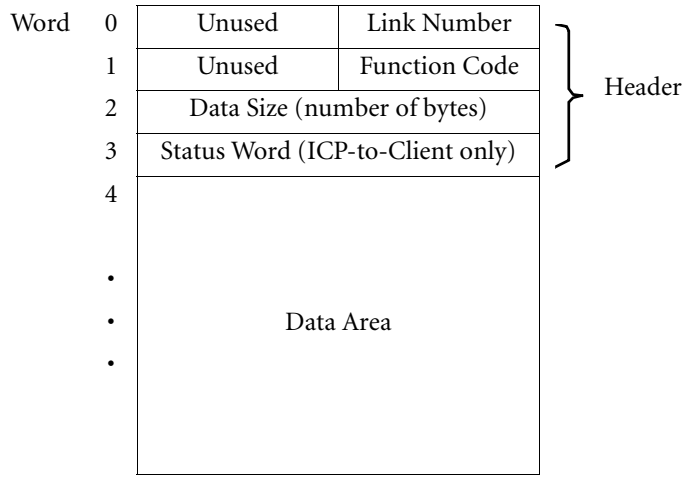

Figure 5–1: Client Asynchronous Protocol Message Format

Note

The protocol-specific description of the message data area is given in each individual protocol programmer's guide.

However there are several differences from the case of synchronous protocol headers:

- The Station Number byte field (Byte 1) is not used.
- The Error Status byte field (Byte 3) is not used.
- The Extended Error Status word field (Word 3) is called the Status word field and is used for all status information.

Table 5–1 lists the Status word bits and their meanings. These bits are numbered from zero to 15. The Status word is used only with the ICP-to-client messages.

Table 5–1: Status Word Field Description (ICP-to-Client Message)

Bit Number	Description
0	(reserved)
1	Framing error
2	Parity error
3	Receive overrun error
4	End-of-message framed
5	Buffer full
6	Receive timeout
7	Loss of DCD
8	Broken message
9	Secure Mode violation
10	CTS timeout
11-14	protocol-specific meanings
15	Rejected message

5.2 Overview of Requests using Raw dlWrite

The following sections describe the link-related commands the client application can send to the ICP for asynchronous protocols. See Table 5–2. (The commands described in Section 4.2 on page 53 are always available, no matter what protocols are being run.)

As noted in Table 5–2, all but one of these commands have corresponding synchronous protocol function codes (the asynchronous codes having been retained for historical reasons when the asynchronous protocols were merged into the Military/Government Protocols set). Asynchronous protocols may use either code; for example, either code 53 or code 9 may be used to enable a link that is assigned an asynchronous protocol.

Table 5–2: Asynchronous-Protocol Function Codes Originating in the Client

Async Code	Equivalent Sync Code	Usage	Reference Section and Page
51	14	Output Data Block	Section 5.2.1 on page 85
52	8	Configure Link	Section 5.2.2 on page 85
53	9	Enable Link	Section 5.2.3 on page 86
54	10	Disable Link	Section 5.2.4 on page 86
55	n/a	Set/Clear Modem Signals	Section 5.2.5 on page 86
56		(reserved)	
57		(reserved)	
58	31	Request Link Status Report	Section 5.2.8 on page 87

5.2.1 Output Data Block [51 (Async)] or [14 (Sync)]

Command 51is identical to command 14 in Section 4.2.14 on page 62.

5.2.2 Configure Link [52 (Async)] or [8 (Sync)]

Command 8 is similar but not identical to command 8 in Section 4.2.8 on page 58. The difference is in the format of the configuration option/value data pairs. In the asynchro-

nous protocol case, the option code is a 16-bit word, as with the synchronous case; but the option value is a 32-bit word, instead of 16 bits. A configuration option code of zero ends the transmission of data, as with the synchronous case.

Figure 5–2 gives an example of the Configure Link command for an asynchronous protocol (whether using function code 52 or 8), using two options. Refer to the individual protocol programmer's guide for configuration option descriptions.

Unused	Link = 1	0
Unused	Function = 52	2
Data Size = 14		4
Reserved for ICP Status		6
Configuration Option		8
Configuration Value (longword)		10
Configuration Option		14
Configuration Value (longword)		16
0		20

Figure 5–2: Link Configuration Block with Two Options

5.2.3 Enable Link [53 (Async)] or [9 (Sync)]

Command 53 is identical to command 9 in Section 4.2.9 on page 60.

5.2.4 Disable Link [54 (Async)] or [10 (Sync)]

Command 54 is identical to command 10 in Section 4.2.10 on page 60.

5.2.5 Set /Clear Modem Signals [55]

The Set/Clear Modem Signals command asserts or deasserts the Data Terminal Ready and Request To Send modem signals. The data field is set with a 16-bit word describing

the action or actions to be taken. Set the appropriate bits in the lower byte of the command word to assert the DTR and/or RTS signals. Set the corresponding bits in the upper byte of the command word to deassert them.

Within each byte, DTR is represented by bit one and RTS by bit four. In the 16-bit data word, turn on DTR by setting bit 1, and turn on RTS by setting bit 4. Turn off DTR by setting bit 9, and turn off RTS by setting bit 12.

5.2.6 (reserved) [56]

Command 56 is reserved.

5.2.7 (reserved) [57]

Command 57 is reserved.

5.2.8 Request Link Status Report [58 (Async)] or [31 (Sync)]

Command 58 is similar to command 31 in Section 4.2.11 on page 61. The difference is that it requests a Type A Link Status Report instead of a Type I, II, or III.

5.3 Overview of Responses using Raw dlRead

The following sections describe the link-related messages that the ICP can send to the client for asynchronous protocols. See Table 5–3.

As noted in Table 5–3, most of these messages have corresponding synchronous protocol function codes (the asynchronous codes having been retained for historical reasons when the asynchronous protocols were merged into the Military/Government Protocols set). Section 4.3 on page 66. Asynchronous protocols may use either code; for example, for a duplicated ICP response message.

Table 5–3: Asynchronous-Protocol Function Codes Originating in the ICP

Async Code	Equivalent Sync Code	Description	Reference Section and Page
71	39	Transmit Acknowledgment	Section 5.3.1 on page 89
72	28	Configure Link Response	Section 5.3.2 on page 89
73	29	Enable Link Response	Section 5.3.3 on page 89
74	30	Disable Link Response	Section 5.3.4 on page 89
75		(reserved)	
76		(reserved)	
77		(reserved)	
78	31	Link Status Report	Section 5.3.5 on page 89
79	n/a	Data Lost Notification	Section 5.3.6 on page 90
80	34	Input Data Block	Section 5.3.7 on page 90
81	n/a	Transmit Negative Acknowledge	Section 5.3.8 on page 90
82	n/a	Protocol-specific Notification	Section 5.3.9 on page 91

5.3.1 Transmit Acknowledgment [71 (Async)] or [39 (Sync)]

Command 71 is similar but not identical to command 39 in Section 4.3.19 on page 79. The difference is in the data area. In the asynchronous protocol case, only one transmission at a time is acknowledged, and the message header data size field contains 0.

5.3.2 Configure Link Response [72 (Async)] or [28 (Sync)]

Command 72 is similar to command 28 in Section 4.3.8 on page 68. There are two differences. The first is in the format of the Link Configuration Report in the data area. For asynchronous protocols, this report has the format shown in Figure 5–2 (but with all configuration items for the assigned protocol).

The second difference from the synchronous Configure Link response is that it is optional. This response by the ICP became operative with version 7.5-6 of the Military/Government protocols. It is sent only if the link has been explicitly configured by the client for the generation of these responses (refer to the individual protocol programmer's guides). Unless stated otherwise in that document, the default is to *not* send acknowledgments, so that previously existing client applications can run without change.

5.3.3 Enable Link Response [73 (Async)] or [29 (Sync)]

Command 73 is identical to command 29 in Section 4.3.9 on page 69.

5.3.4 Disable Link Response [74 (Async)] or [30 (Sync)]

Command 74 is identical to command 30 in Section 4.3.10 on page 69.

5.3.5 Link Status Report [78 (Async)] or [31 (Sync)]

Command 58 is similar to command 11 in Section 4.3.11 on page 69. The difference is that a Type A Link Status Report is generated, instead of a Type I, II, or III. See Figure 5–3. The message header data size field contains 2. Word 0 contains the report in the lower byte.

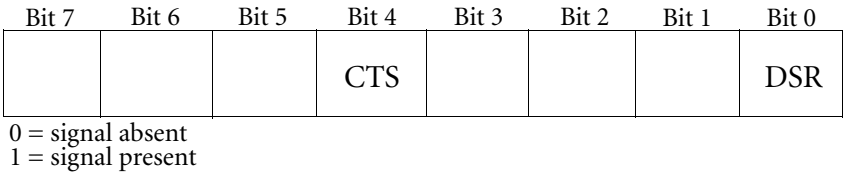

Figure 5–3: Link Status Report, Type A

5.3.6 Data Lost Notification [79]

This message informs the client that incoming link data has been discarded for lack of available buffers. When there are no communication buffers left on the ICP to assign to the reception of the next message on a link, the ICP discards all messages received on that link until a buffer becomes available. This condition is caused by the client failing to read messages from the ICP at a rate compatible with the incoming data stream. The message header data size field contains zero.

5.3.7 Input Data Block [80 (Async)] or [34 (Sync)]

Command 80 is identical to command 34 in Section 4.3.14 on page 73. Function code 80 is generated when the link was enabled with function code 73. Function code 34 is generated when the link was enabled with function code 9.

5.3.8 Transmit Negative Acknowledgment [81]

The Transmit Negative Acknowledgment notification signifies that the asynchronous transmission was aborted or refused. The reason for aborting is loss of CTS. The reason for refusal is a Secure-Mode violation. Each is specified by a bit in the message header

Status field (Table 5–1). The message header data size field contains 2. For loss of CTS, word 0 of the data area contains the number of characters (possibly zero) sent out before the transmission was aborted for loss of CTS. For a Secure-Mode Violation, word 0 does not contain a defined value.

5.3.9 Protocol-specific Notification [82]

This is a protocol-specific ICP notification. For those protocols to which is pertains, it is described in the individual protocol programmer's guides.

Appendix

A Military/Government Protocols Loopback Test Program

Note

The term "Freeway" can mean either a Freeway server or an embedded ICP. For the embedded ICP, also refer to the user's guide for your ICP and operating system (for example, the *ICP2432 User Guide for Windows NT*).

This appendix describes the Military/Government Protocols loopback test procedure, including the following:

- an overview of the test
- a description of how to install the hardware needed for the test
- instructions on how to run the test
- sample screen display from the test

Note

Before running the loopback test, you must install the Freeway software and boot Freeway to download the software as described in the *Freeway User Guide* (for a Freeway server) or the user's guide for your embedded ICP and operating system (for example, the *ICP2432 User Guide for Windows NT*).

Note

When the loopback test is run under VMS, Protogate recommends a minimum Buffered I/O Byte Count process quota of 30,000 bytes. The AST Limit and Open File Limit process quotas must provide a unit for each session that will be opened. The number of sessions is equal to the number of selected ports, plus one.

A.1 Overview of the Test Program

The loopback test uses the data link interface (DLI) available with Freeway. The DLI library is in the freeway/client/*op-sys*/lib directory (where *op-sys* is the identifier for the operating system you are using) and can be used with any data link protocol on Freeway.

The loopback test program is placed in the freeway/client/*op-sys*/bin directory during the installation procedures.

Note

Earlier Freeway terminology used the term "synchronous" for blocking I/O and "asynchronous" for non-blocking I/O. Some parameter names reflect the previous terminology.

One high-level test program written in C is supplied with the Military/Government Protocols: milalp, which uses non-blocking I/O, meaning that the asyncIO DLI configuration parameter (described in the *Freeway Data Link Interface Reference Guide*) must be set to "yes" (the default is "no" for blocking I/O). The test is interactive; it prompts you for all the information needed to run the test.

The loopback test performs the following functions:

- Configures ICP buffer sizes

- Configures the link-level control parameters such as baud rates, clocking, and protocol
- Enables and disables links
- Initiates the transmission and reception of data on the serial lines
- Obtains link statistics from Freeway

You can use the loopback test as a template for designing client applications that interface with the DLI layer. You can also use it to verify that the installed Freeway devices and cables are functioning correctly.

A.2 Hardware Setup for the Test Program

The test program runs in loopback mode. Before running the test, perform the following procedure to install the loopback cabling:

Step 1: Provide a synchronous modem. Configure the modem to supply continuous clocking at a data rate between 50 and 19,200 bits per second. The Freeway ICPs are default configured for external clocking, and the modem supplies the clock signal for loopback testing.

Step 2: Select pairs of adjacent ports to loopback. Ports are looped back in the following pairs: (1,2), (3,4), (5,6), and so on. Install the special three-headed loopback cable between the ports you selected and the synchronous modem. For information on port numbering[1] and instructions on how to install the loopback cable, refer to the appropriate "Port Numbering and Cabling" information in your hardware installation guide.

1. The "Port Numbering and Cabling" information in your hardware installation guide reflects the standard numbering convention of (0,1), (2,3), etc. The loopback test program uses (1,2), (3,4), and so on.

Note

The loopback cable is used only during testing, not during normal operation.

A.3 Running the Test Program

Caution

To run the test program successfully, you must have write privileges in the bin directory on the boot server.

Step 1: **To start the test program in a UNIX or Windows NT system**, change to the directory that contains the test program. For example, if you performed the default installation on a UNIX system, this directory is called /usr/local/freeway/client/*op-sys*/bin where *op-sys* is the identifier for the operating system you are using. On a Windows NT system the default directory is c:\freeway\client*op-sys*\bin. Enter the following command at the prompt:

milalp

To start the test in a VMS system, change to the directory that contains the test program. If you performed the default installation, this directory is called SYS$SYSDEVICE:[FREEWAY.CLIENT.*OP-SYS*.BIN], where *OP-SYS* is the identifier for the hardware platform model and TCP/IP software you are using, for example AXP_TCPWARE. Enter the following command at the prompt:

RUN MILALP

The test may also be invoked with any combination of four options, using a bit-encoded value as a program argument:

bit 0: Suppress the user help query
bit 1: Allow for multiple ICPs in the test

bit 2: Assume all port clocking is internal

bit 3: Prompt user for ICP buffer sizing

Examples of program invocations are (hexadecimal values required):

milalp	(no options)
milalp 2	(multiple ICPs in test)
milalp 0xd	(no help screens, internal clocking, buffer sizing)

Other bit values in the argument are reserved for Protogate use. These uses are documented in the "Module Description" comments near the beginning of the freeway/client/test/mil/milalp.c source file.

Step 2: The following prompts are displayed. Defaults are shown in parentheses:

Select ICP device number (range [0,5]; default 0):

Enter the number of the ICP to be tested. This is the ICP that you cabled for testing in Step 2 on page 95.

Select ICP <i> Link <j> protocol (range [-1,2]; default 1):

Specify which protocol is to be tested on the given link.

Select ICP <i> Link <j> baud-rate [default=2400] (range [1,16]; default 1):

Specify which baud rate is to be used on the given link.

Select ICP <i> Segmentation buffer data area size (range [512,2048]; default 512):

Enter the ICP segmentation buffer data area size, in bytes.

Select ICP <i> Communication buffer data area size (range [<m>,<n>]; default <k>):

Enter the ICP communication buffer data area size, in bytes (minimum is dependent on protocols selected; maximum is dependent on selected segmentation buffer size).

Select ICP <i> transmit window size (range [1,<n>]; default 1):

Enter the ICP transmit acknowledge configuration parameter (maximum is dependent on the specified buffer sizes).

Select test length in minutes (range [1,100000]; default 1):

Enter the number of minutes you want the test to run.

Step 3: After you answer the last prompt, data transfer starts. The test displays a series of numbers, greater than (>) symbols, and less than (<) symbols to indicate that it is running. Numbers represent the links, greater than symbols appended to numbers represent data sent, and less than symbols prepended to numbers represent data received. A link number and its symbol are displayed upon the accumulation of a number of messages equal to three times the window size.

Step 4: Remove the loopback cable and configure the cables for normal operation as described in the appropriate "Port Numbering and Cabling" appendix in your hardware installation guide. Your Freeway is now configured to communicate with its clients.

A.4 Sample Output from Test Program

Figure A–1 shows the screen display from a sample Military/Government Protocols non-blocking loopback test program. Output displayed by the program is shown in typewriter type and your responses are shown in **bold type**. Each entry is followed by a carriage return.

```
% milalp 0xd

NO HELP selected
INTERNAL CLOCKING selected
SPECIFY BUFFER SIZES selected

PROTOGATE MILITARY/GOVERNMENT PROTOCOLS CLIENT TEST PROGRAM 5.0-0 09-FEB-00

Select ICP device number (range [0,5]; default 0): 1

ICP selected: 1

Session icp1general opened

ICP Protocol Software Version: Protogate MIL/GOV  7.5-4  07-JAN-00
Number of Links on ICP 1: 4

AVAILABLE PROTOCOLS FOR ICP 1:
  ( 0  ==> Omit this link from test)
  (-1  ==> Omit remaining links from test)
    1. PROT-A
    2. PROT-B

Select ICP 1 Link 1 protocol (range [-1,2]; default 1): 1
Select ICP 1 Link 2 protocol (range [-1,2]; default 1): 1
Select ICP 1 Link 3 protocol (range [-1,2]; default 1): 2
Select ICP 1 Link 4 protocol (range [-1,2]; default 1): 2
```

Figure A–1: Sample Output from Loopback Program

```
      BAUD     INTERNAL
      RATE     CLOCKING
     Default       1
     50 Baud (a)   2
   56.8 Baud (a)   3
   74.2 Baud (a)   4
    300 Baud       5
    600 Baud (s)   6
   1050 Baud (a)   7
   1200 Baud       8
   2400 Baud       9
   4800 Baud      10
   9600 Baud      11
  19200 Baud      12
  38400 Baud      13
 (a) = Asynchronous protocols
  (s) = Synchronous protocols

Select ICP 1 Link 1 baud-rate [default=2400] (range [1,16]; default 1): 1
Select ICP 1 Link 2 baud-rate [default=2400] (range [1,16]; default 1): 1
Select ICP 1 Link 3 baud-rate [default=2400] (range [1,16]; default 1): 1
Select ICP 1 Link 4 baud-rate [default=2400] (range [1,16]; default 1): 1

Select ICP 1 Segmentation buffer data area size (range [512,2048]; default 512): 512
Select ICP 1 Communication buffer data area size (range [38,500]; default 38): 38

Size   of ICP 1 segmentation  buffers:   512
Size   of ICP 1 communication buffers:    38
Number of ICP 1 segmentation  buffers:  1734
Number of ICP 1 communication buffers:  10627
```

Figure A–1: Sample Output from Loopback Program *(Cont'd)*

```
Select ICP 1 transmit window size (range [1,11]; default 1): 10

Opening sessions for selected links....

Session icp1port0 opened
Session icp1port1 opened
Session icp1port2 opened
Session icp1port3 opened

Configuring selected links....

Selected links configured

Enabling selected links....

Selected links enabled

Select test length in minutes (range [1,100000]; default 1): 1

Starting data transfer

1>2>3>4>1>2>3>4><2<11>2><4<3<1<21>2>3>4><1<21>2><3<4<2<11>3>2>4><3<4<2<11>2><2<
13>4>1>2><4<2<3<11>2>3>4><1<2<3<41>2><13>1>4><1<4<3<2<22>1><23>4>2>1><2<3<4<12>
1><23>4><4<1<32>1><2<12>3>4>1><1<4<2<31>2><13>4><2<41>2><3<1<22>1>3>4><1<21>2><
4<3<1<22>3>1>4><3<4<1<22>1><1<24>2>1><4<1<23><32>1><1<24>2>1><1<3<23><11>2><44>
<2<32>1><23>4><11>2><2<4<3<13>2>1><2<44><3<12>1>3><1<24>1>2><4<1<3<22>1>3><4<14
><3<21>2><1<23>2>1><1<44><2<32>1><13><22>4>1><1<2<4<31>2>3><1<2<44>1>2><3<1<23>
1>2><14><2<42>1><1<3<23><41>2>4><3<2<11>2><23>4><1<42>1><2<3<13><42>1>4><1<22>1
><33><1<21>2><44><3<1<22>1>3><14><22>1><4<1<3<23>2>1><44><1<2<31>2><1<23>2>4>1>
<1<4<2<32>1>3><1<2<44><32>1><1<23>2>1><1<24><4<32>1><23><4<14>2>1><2<1<31>2><23
>4><4<11>2><33><22>4><4<22><3<1<11>3>1><1<44>2><2<3<11>2>3><2<11>4><42><2

Shutting down selected links....

<3<1
```

Figure A–1: Sample Output from Loopback Program *(Cont'd)*

```
Host Statistics for ICP 1
-------------------------
Data Rate          2400   2400   2400   2400
Link ID number        1      2      3      4
RCV % efficient      61     61     87     85
XMT % efficient      62     62     87     88
RCV data packets   1837   1850   1122   1100
XMT data packets   1870   1860   1120   1140
XMT data unacked     20     19     20     16
XMT data NAKed        0      0      0      0
            73      Average percent efficiency (receive)
            74      Average percent efficiency (transmit)
            252     Data Segmentation Buffers received
            599     Data Segmentation Buffers sent

ICP 1 Synchronous Protocol Statistics
-------------------------------------
Protocol         PROT-A PROT-A PROT-B PROT-B
Link ID number        1      2      3      4
Data Rate          2400   2400   2400   2400
Msgs received      1837   1850   1122   1100
Mark/Parity errs      0      0      0      0
Chkgrp/CRC errs       0      0      0      0
Rcv overruns          0      0      0      0
Xmt underruns         0      0      0      0
Megs recd wo err   1837   1850   1122   1100
Msgs recd w errs      0      0      0      0
EOM errs              0      0      0      0
Messages sent      1850   1841   1100   1124
Lost messages         0      0      0      0

Selected links shut down

Session icp1general closed
Session icp1port0 closed
Session icp1port1 closed
Session icp1port2 closed
Session icp1port3 closed

Loopback test complete
```

Figure A–1: Sample Output from Loopback Program *(Cont'd)*

Appendix

B DLI and TSI Configuration Process

Note

The term "Freeway" can mean either a Freeway server or an embedded ICP. For the embedded ICP, also refer to the user's guide for your ICP and operating system (for example, the *ICP2432 User Guide for Windows NT*).

This appendix summarizes the process for configuring DLI sessions and TSI connections. DLI and TSI text configuration files are used as input to the dlicfg and tsicfg preprocessor programs to produce binary configuration files which are used by the dlInit and dlOpen functions. For embedded ICPs, only a DLI configuration file is used (not a TSI configuration file).

During your client application development and testing, you might need to perform DLI configuration repeatedly (as well as TSI configuration for a Freeway server).

Note

Some of the specifics regarding Military/Government Protocols DLI session configuration were described earlier in Section 2.2.1 on page 34.

You should be familiar with the protocol-independent configuration procedures described in the *Freeway Data Link Interface Reference Guide* and the *Freeway Transport Subsystem Interface Reference Guide*.

The DLI and TSI configuration files provided with the product are listed in Table B–1.

Table B–1: Configuration File Names

	Freeway Server	**Embedded ICP**
DLI:	mildcfg	milembcfg
TSI:	miltcfg	TSI not applicable for embedded ICP

The DLI and TSI configuration procedures are summarized as follows. Keep in mind that TSI configuration does not apply to an embedded ICP environment.

1. For a Freeway server, create or modify a TSI text configuration file specifying the configuration of the TSI connections (for example, miltcfg in the freeway/client/test/mil directory).

2. Create or modify a DLI text configuration file specifying the DLI session configuration for all ICPs and serial communication links in your system (for example, mildcfg in the freeway/client/test/mil directory).

3. If you have a UNIX or Windows NT system, skip this step. If you have a VMS system, run the makefc.com command file from the [FREEWAY.CLIENT.TEST.MIL] directory to create the foreign commands used for dlicfg and tsicfg.

   @MAKEFC <*tcp-sys*>

   where <*tcp-sys*> is your TCP/IP package:

MULTINET	(for a Multinet system)
TCPWARE	(for TCPware system)
UCX	(for a UCX system)

   VMS example: @MAKEFC UCX

4. For a Freeway server, go to the freeway/client/test/mil directory and execute tsicfg with the text file from Step 1 as input. This creates the TSI binary configuration

file in the same directory as the location of the text file (unless a different path is supplied with the optional filename). If the optional filename is not supplied, the binary file is given the same name as your TSI text configuration file plus a .bin extension.

tsicfg *TSI-text-configuration-filename [TSI-binary-configuration-filename]*

VMS example:	tsicfg miltcfg
UNIX example:	freeway/client/*op-sys*/bin/tsicfg miltcfg
NT example:	freeway\client*op-sys*\bin\tsicfg miltcfg

5. From the freeway/client/test/mil directory, execute dlicfg with the text file from Step 2 as input. This creates the DLI binary configuration file in the same directory as the location of the text file (unless a different path is supplied with the optional filename). If the optional filename is not supplied, the binary file is given the same name as your DLI text configuration file plus a .bin extension.

   dlicfg *DLI-text-configuration-filename [DLI-binary-configuration-filename]*

VMS example:	dlicfg mildcfg
UNIX example:	freeway/client/*op-sys*/bin/dlicfg mildcfg
NT example:	freeway\client*op-sys*\bin\dlicfg mildcfg

Note

You must rerun dlicfg or tsicfg whenever you modify the text configuration file so that the DLI or TSI functions can apply the changes. On all but VMS systems, if a binary file already exists with the same name in the directory, the existing file is renamed by appending the .BAK extension. If the renamed file duplicates an existing file in the directory, the existing file is removed by the configuration preprocessor program.

6. If you have a UNIX system, move the binary configuration files that you created in Step 4 and Step 5 into the appropriate freeway/client/*op-sys*/bin directory where *op-sys* indicates the operating system: dec, hpux, sgi, solaris, or sunos.

   UNIX example: mv mildcfg.bin /usr/local/freeway/client/hpux/bin
   mv miltcfg.bin /usr/local/freeway/client/hpux/bin

7. If you have a VMS system, run the move.com command file from the [FREEWAY.CLIENT.TEST.MIL] directory. This moves the binary configuration files you created in Step 4 and Step 5 into the bin directory for your particular TCP/IP package.

   @MOVE *filename* <*tcp-sys*>

   where *filename* is the name of the binary configuration file and <*tcp-sys*> is the TCP/IP package:

MULTINET	(for a Multinet system)
TCPWARE	(for TCPware system)
UCX	(for a UCX system)

   VMS example: @MOVE MILDCFG.BIN UCX

8. If you have a Windows NT system, move the binary configuration files that you created in Step 4 and Step 5 into the appropriate freeway\client*op-sys*\bin directory where *op-sys* indicates the operating system: axp_nt or int_nt (for Freeway server); axp_nt_emb or int_nt_emb (for an embedded ICP).

   NT example: copy mildcfg.bin \freeway\client\axp_nt\bin
   copy miltcfg.bin \freeway\client\axp_nt\bin

When your application calls the dlInit function, the DLI and TSI binary configuration files generated in Step 4 and Step 5 are used to configure the DLI sessions and TSI connections. Figure B–1 shows the configuration process.

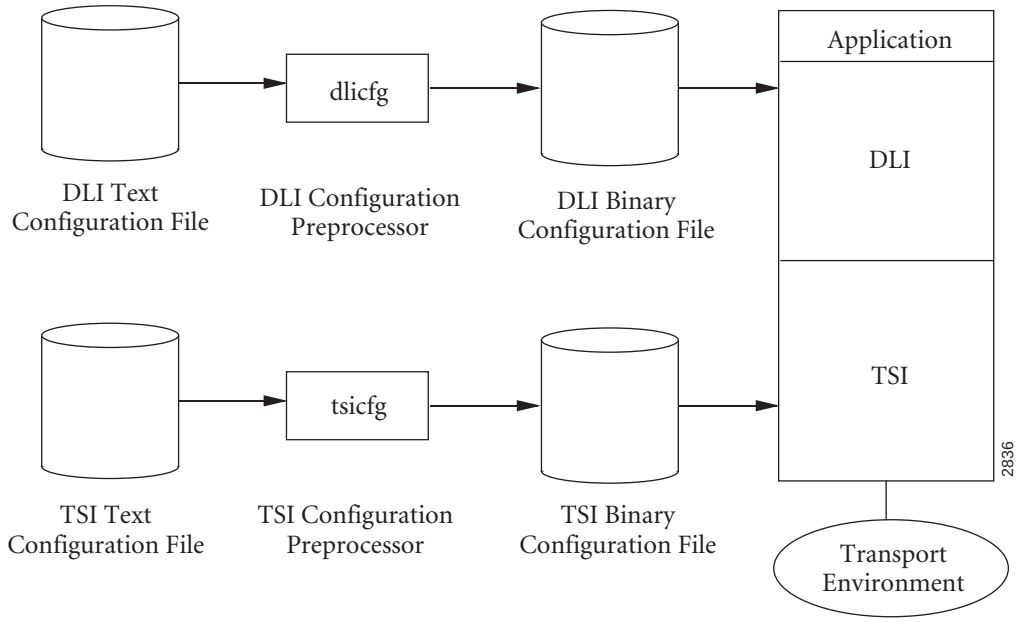

Figure B–1: DLI and TSI Configuration Process

Appendix

C Military/Government Protocols Input Data Time Stamping

C.1 The Concept of Time Stamping

Protogate's Military/Government Protocols include a 32-bit time stamp with each input data message that is received from a link and sent to the client. This value is inserted as the first four bytes following the individual Input Data message header and preceding the protocol message data itself. The time-stamp value represents some number of milliseconds. To the ICP, this is an arbitrary number. To the client program, it is an offset from a base time determined by the client program.

The time stamp for a given input data message is assigned upon the arrival of the first bit of the message data at the serial link. It is taken from a 32-bit time-stamp word maintained by the OS/Protogate operating system. The value is set to zero when the ICP is downloaded, and is then incremented once per millisecond. Two commands that alter this default activity are available to the client: Set Time Stamping and Set Time Stamp Value.

The Set Time Stamping command may specify either of two kinds of time-stamp activity: Internal timing (the default at startup) and PCI Host timing. The Set Time Stamp Value command applies only when Internal timing is active.

C.2 Internal Timing

When doing Internal timing, the ICP simply updates its 32-bit time-stamp value once per millisecond. This is the only process of maintaining the time-stamp value, apart from interventions by the Set Time Stamp Value command. This command enables the

client program to periodically correct any timing drift that may be occurring on the ICP with respect to client-system timekeeping, by specifying a replacement value for the current time-stamp contents. Other than obeying this client command, the ICP does not instigate any timing correction on its own. The only wraparound to 0 with Internal timing is when 32-bit overflow occurs.

C.3 PCI Host Timing

When doing PCI Host timing, the ICP regularly updates its time-stamp contents via PCIbus transactions with the ICP driver that resides in whatever system hosts the ICP. PCI Host timing can be specified only when the ICP driver has the PCI Host timing capability. Freeway servers with the FreeBSD operating system support PCI Host timing. ICP drivers in systems with the ICPs embedded in the host PCIbus may or may not support PCI Host timing, depending on the host system and the driver release level.

As a 32-bit count of milliseconds, the PCI Host timing time-stamp represents an offset from the most recent midnight. It wraps around to zero at the next midnight (86,400,000 milliseconds). PCI Host timing values are kept to within a millisecond of the host system time via PCIbus transactions. Note that with Freeway servers or client systems that tune their system times according to external time sources (such as Network Time Protocol), the PCI Host timing thereby reflects that external source.

PCI Host timing is selected by the Set Time Stamping client command. This command selects PCI Host timing by specifying a 16-bit data value of -2.

A data value of 0 or greater selects Internal timing, if the client system ever needs to return to that timing mode. Any other negative value is rejected by the ICP. Two explanatory notes: (1) the value -2 instead of -1 is used by this command because -1 was used for a timing mode that pertained only to VMEbus ICPs; and (2) all non-negative values are permitted for establishing Internal timing (not just 0) in order to retain compatibility with the time-stamping methods used by pre-OS/Protogate client programs, which thereby may continue to run unchanged on ICPs with OS/Protogate.

Index

A

Acknowledgments
  attach 47
  detach 47
  transmit 79, 89
  transmit (negative) 90
Addressing
  Internet 24
alwaysQIO DLI parameter 32, 35
asyncIO DLI parameter 32, 35
Attach
  acknowledgment 47
  command 38, 46
Audience 11

B

Binary configuration files 24, 32, 34, 42, 104
Bit numbering 15
Blocking interval
  command 57
Blocking of messages 27
Buffer
  communication 28, 55
  management 33
  segmentation 28, 54
Buffer status report
  request 64
  response 80
Byte ordering 15

C

cfgLink DLI parameter 35
Client control 29
Client-server environment 23
  summary of steps 24
Clocks
  system parameters area 76
Closing DLI sessions 38
Commands
  attach 38, 46
  detach 38, 46
  foreign 104
  see dlWrite categories
Communication buffer 28, 29
  command 55
Configuration
  binary files 32, 34, 42, 104
  communication buffer 55
  custom link configuration 29
  DLI
    alwaysQIO parameter 32, 35
    asyncIO parameter 32, 35
    cfgLink parameter 35
    enable parameter 35
    example 36, 37
    localAck parameter 35
    main section 34
    portNo parameter 35
    protocol parameter 35
    sessions 34
    summary 104
    transport parameter 34
  DLI and TSI process 103
  dlicfg program 32, 105
  Freeway environment 31
  segmentation buffer 54
  TACMIL environment 28
  TSI
    configuration file 34
    maxBufSize parameter 33, 54

summary 104
tsicfg program 32, 104
Configure link
command 58, 85
example 59, 86
Connection
TSI configuration 32, 34, 103
Control (client) 29
Custom link configuration 29
Customer support 17

D

Data 52
input data block 73, 90
segmentation header 73
system parameters 76
messages 76
output data block 62, 85
Data flushing 29
Data link interface (DLI) 23, 24
Data lost report
response 90
Data size field 52
Deblocking of messages 27
Detach
acknowledgment 47
command 38, 46
Direct memory access 23
Disable link
command 60, 86
dlBufAlloc (*see also* Functions) 44
dlBufFree (*see also* Functions) 44
dlClose (*see also* Functions) 44
dlControl (*see also* Functions) 44
dlerrno global variable 44
DLI concepts
configuration 31
see also Configuration, DLI
configuration process 103
initialization 34
non-blocking I/O 32
session configuration 34
summary 31
DLI functions 41
example call sequence 42
overview 43
see also Functions
summary table 44
syntax synopsis 44
dlicfg preprocessor program 32, 105
dlInit (*see also* Functions) 44
dlOpen (*see also* Functions) 44
dlpErrString (*see also* Functions) 44
dlPoll (*see also* Functions) 44
dlRead categories
acknowledgments
attach 47
detach 47
transmit 79, 89
transmit (negative) 90
confirmations
protocol-specific ICP response 79
data reception 73, 90
reports
buffer status 80
data lost 90
link status 69
link status notification 77
number of links 80
queue count 73
software version 80
statistics 72
dlRead (*see also* Functions) 44
dlTerm (*see also* Functions) 44
dlWrite categories
commands
attach 38, 46
configure communication buffer 55
configure link 58, 85
configure segmentation buffer 54
detach 38, 46
disable link 60, 86
enable link 60, 86
set blocking interval 57
set protocol 58
set signal monitor interval 57
set time stamp interval 56
set time stamp value 64
data transfer 62, 85
reports

buffer status 64
link statistics report 61
link status report 61
number of links 64
queue count 61
software version 64
dlWrite (*see also* Functions) 44
Documents
reference 13
Download software 24, 43

E

Embedded ICP
overview 21
enable DLI parameter 35
Enable link
command 60, 86
Error codes
dlerrno global variable 44
Error reporting 39
link statistics report 72
Error status field 39, 50
bit meanings 51
extended 52
format 51
rejected bit 66
Ethernet 22
Example
call sequence 42
configure link command 59, 86
DLI configuration file 36, 37
loopback test program 93
Extended error status field 52

F

FDDI 22
Features
product 22
Files
binary configuration 32, 34, 42, 104
configuration file names 104
example DLI configuration 36, 37
makefc.com 104
move.com 106
mp_verify.c 26
Flushing of data 29
Foreign commands 104
Format of messages 49
asynchronous 83
Freeway
client-server environment 23
environment configuration 31
overview 19
Function code field 50
client codes 85
ICP codes 88
Functions
dlBufAlloc 44
dlBufFree 44
dlClose 44
dlControl 44
dlInit 44
dlOpen 44
dlpErrString 44
dlPoll 44
dlRead 44, 66, 88
optional arguments 45
see also dlRead categories
dlSyncSelect 44
dlTerm 44
dlWrite 44, 53, 85
optional arguments 45
see also dlWrite categories

G

General session 28, 34, 35, 42, 46, 53
example configuration 36, 37
unsolicited reports 81

H

Hardware components 26
Header
format 50
segmentation 73
Header fields
data size 52
error status 50
format 51
extended error status
format 52

function code 50
client codes 85
ICP codes 88
link number 50
station number 50
History of revisions 16

I

Idle pattern
loss 30
Initializing the DLI 34
Input data
time stamping 109
Input data block 73, 90
messages 76
segmentation header 73
system parameters 76
Internet addresses 24
I/O
non-blocking 32

L

LAN interface processor 20
lib subdirectory 94
Link 59, 68
configure
command 58, 85
custom configuration 29
disable
command 60, 86
enable
command 60, 86
link-specific session 28, 35
see also Session
numbering 35, 50
Link number field 50
Link statistics report
request 61
response 72
Link status notification
response 77
Link status report
request 61
response 69
localAck DLI parameter 35
Loopback test
milalp 94
TACMIL protocol
non-blocking sample output 99
Loopback test program 93

M

makefc.com file 104
maxBufSize TSI parameter 33, 54
Message
blocking 27
deblocking 27
header format 50
input data block 73, 74, 76, 90
output data block 62, 63, 85
segmentation 27
Message format 49
asynchronous 83
input data block 74
output data block 63
Messages
time stamping 29
milalp loopback test 94
Modem signal monitoring 29
move.com file 106
mp_verify test program 26

N

Non-blocking I/O 32
call sequence 42
Number of links report
request 64
response 80

O

Opening DLI sessions 35
Operating system
Protogate's real-time 20, 21
Optional arguments 38
dlRead relevant 47
dlWrite required 46
structure 45
OS/Impact 26
Output data block 62, 63, 85
Overview

DLI and TSI configuration 103
DLI functions 43
dlRead responses 66, 88
dlWrite requests 53, 85
embedded ICP 21
Freeway server 19
product 19

P

portNo DLI parameter 35
Product
features 22
introduction 19
overview 19
support 17
Programs
dlicfg preprocessor 32, 105
loopback test 93
test 26
tsicfg preprocessor 32, 104
Protocol
command 58
summary 27
protocol DLI parameter 35
Protocol selection 29
Protocol-specific ICP response 79

Q

Queue count report
request 61
response 73

R

Raw operation 38, 45
Reference documents 13
Reports
buffer status 64, 80
data lost 90
link statistics 61, 72
link status 61, 69
link status notification 77
number of links 64, 80
queue count 61, 73
software version 64, 80
unsolicited 81
Request 87
Responses
see dlRead categories
Revision history 16
rlogin 22

S

Segmentation
buffer 28
command 54
header 73
of messages 27
Server processor 20
Session
closing 38
DLI configuration 31, 34, 103
general 35
general session 28, 34, 42, 46, 53
general session example 36, 37
link-specific 28, 35, 42, 53
opening 35
Signal monitor interval
command 57
Signals
status 71, 77, 90
SNMP 22
Software
components 26
download 24, 43
Software version report
request 64
response 80
Station number field 50
Statistics report
request 61
response 72
Status report
request 61
response 69
Support, product 17
System parameters area 76

T

TACMIL
DLI functions 41

environment configuration 28
hardware description 26
protocol summary 27
software description 26
TCP/IP 22
package 104
Technical support 17
telnet 22
Test program 93
Test programs
mp_verify 26
Time stamp field 52
Time stamp interval
command 56
Time stamp value
command 64
Time stamping 29, 109
Transmit acknowledgment 79, 89
Transmit negative acknowledgment 90
transport DLI parameter 34
Transport subsystem interface (TSI) 24
TSI configuration
process 103
see Configuration, TSI
tsicfg preprocessor program 32, 104

U

UNIX
configuration process 103

V

Version 80
VMS
configuration process 103
VxWorks 20

W

WAN interface processor 20
Windows NT
configuration process 104

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 Protogate at 12225 World Trade Drive, Suite R, San Diego, CA 92128, or fax it to (877) 473-0190.

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: _____

_____

_____

_____

_____

_____

Protogate, Inc.
Customer Service
12225 World Trade Drive, Suite R
San Diego, CA 92128