Bailey® infi 90®

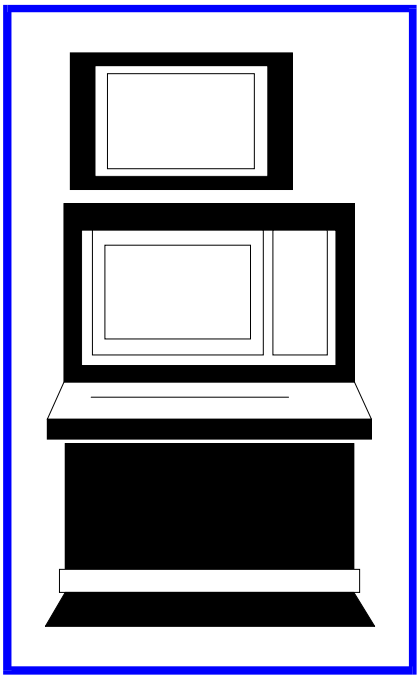
semAPI

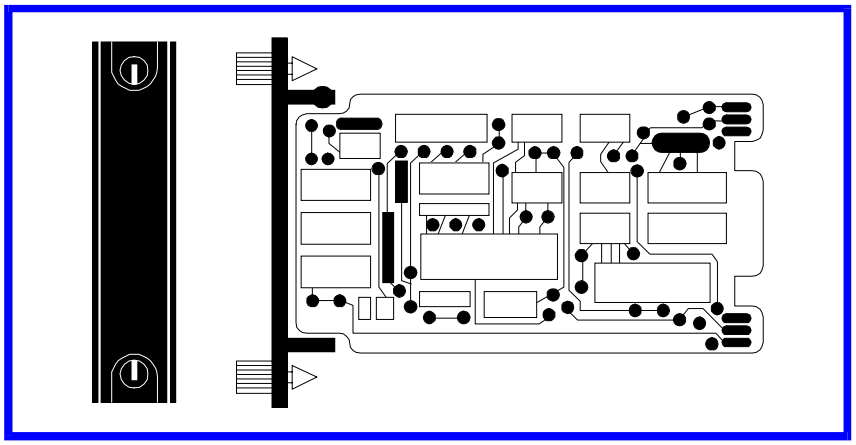
Instruction

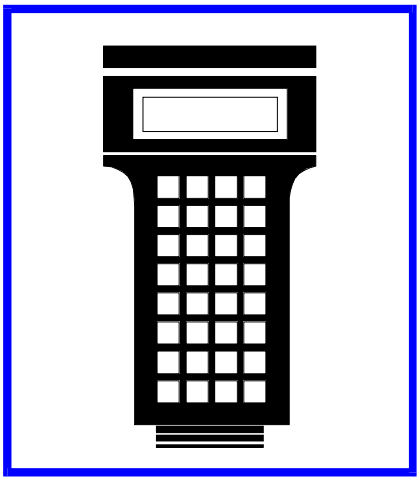
Application Programming Interface

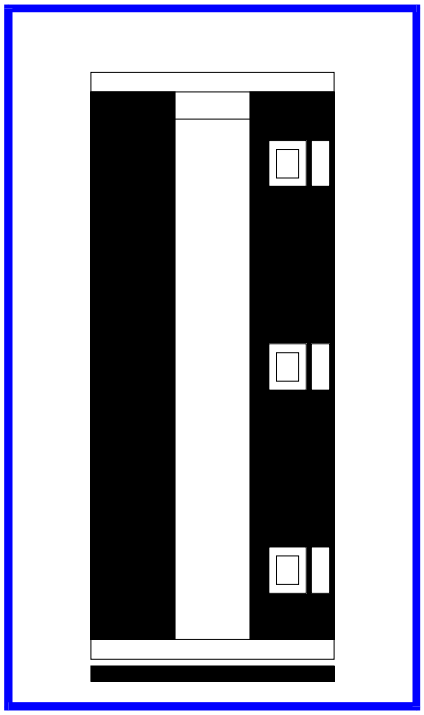
HP-UX Platform

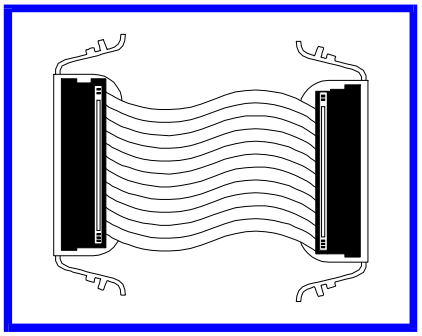
(Release 1.1)

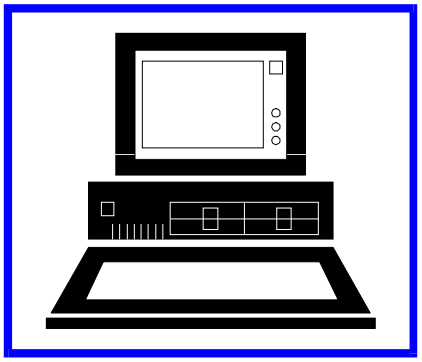












Bailey Process Control and Automation Solutions from Elsag Bailey Group

WARNING notices as used in this instruction apply to hazards or unsafe practices that could result in personal injury or death.

CAUTION notices apply to hazards or unsafe practices that could result in property damage.

NOTES highlight procedures and contain information that assists the operator in understanding the information contained in this instruction.

WARNING

INSTRUCTION MANUALS

DO NOT INSTALL, MAINTAIN, OR OPERATE THIS EQUIPMENT WITHOUT READING, UNDERSTANDING, AND FOLLOWING THE PROPER **Elsag Bailey** INSTRUCTIONS AND MANUALS; OTHERWISE, INJURY OR DAMAGE MAY RESULT.

RADIO FREQUENCY INTERFERENCE

MOST ELECTRONIC EQUIPMENT IS INFLUENCED BY RADIO FREQUENCY INTERFERENCE (RFI). CAUTION SHOULD BE EXERCISED WITH REGARD TO THE USE OF PORTABLE COMMUNICATIONS EQUIPMENT IN THE AREA AROUND SUCH EQUIPMENT. PRUDENT PRACTICE DICTATES THAT SIGNS SHOULD BE POSTED IN THE VICINITY OF THE EQUIPMENT CAUTIONING AGAINST THE USE OF PORTABLE COMMUNICATIONS EQUIPMENT.

POSSIBLE PROCESS UPSETS

MAINTENANCE MUST BE PERFORMED ONLY BY QUALIFIED PERSONNEL AND ONLY AFTER SECURING EQUIPMENT CONTROLLED BY THIS PRODUCT. ADJUSTING OR REMOVING THIS PRODUCT WHILE IT IS IN THE SYSTEM MAY UPSET THE PROCESS BEING CONTROLLED. SOME PROCESS UPSETS MAY CAUSE INJURY OR DAMAGE.

NOTICE

The information contained in this document is subject to change without notice.

Elsag Bailey, its affiliates, employees, and agents, and the authors and contributors to this publication specifically disclaim all liabilities and warranties, express and implied (including warranties of merchantability and fitness for a particular purpose), for the accuracy, currency, completeness, and/or reliability of the information contained herein and/or for the fitness for any particular use and/or for the performance of any material and/or equipment selected in whole or part with the user of/or in reliance upon information contained herein. Selection of materials and/or equipment is at the sole risk of the user of this publication.

Preface

The Strategic Enterprise Management Application Programming Interface (semAPI) software is a library of functions that provides software application programs access to Elsag Bailey INFI 90 OPEN control system information.

NOTE: The semAPI software requires the use of a software key (provided with the software) for operation.

This software describes the semAPI software for the HP-UX platform. The semAPI software is available in two functional levels: data acquisition and supervisory control.

Data Acquisition (DA)

Provides data acquisition and process monitoring capabilities. The DA level allows an application to read data from an INFI 90 OPEN system.

Supervisory Control (SC)

Provides data acquisition, process monitoring and supervisory capabilities. The SC level allows an application to read and write data to and from an INFI 90 OPEN system.

List of Effective Pages

Total number of pages in this instruction is 218, consisting of the following:

Page No.	Change Date
Preface	Original
List of Effective Pages	Original
iii through ix	Original
1-1 through 1-8	Original
2-1 through 2-4	Original
3-1 through 3-7	Original
4-1 through 4-7	Original
5-1 through 5-13	Original
6-1 through 6-91	Original
7-1 through 7-2	Original
8-1 through 8-9	Original
9-1 through 9-9	Original
A-1 through A-2	Original
B-1 through B-2	Original
C-1	Original
D-1 through D-9	Original
E-1 through E-19	Original
F-1 through F-2	Original
G-1 through G-7	Original
H-1	Original
I-1 through I-4	Original
J-1	Original
K-1	Original
L-1 through L-2	Original
M-1 through M-2	Original
N-1 through N-4	Original
Index-1 through Index-2	Original

When an update is received, insert the latest changed pages and dispose of the superseded pages.

NOTE: On an update page, the changed text or table is indicated by a vertical bar in the outer margin of the page adjacent to the changed area. A changed figure is indicated by a vertical bar in the outer margin next to the figure caption. The date the update was prepared will appear beside the page number.

Table of Contents

Table of Contents (continued)

Table of Contents (continued)

Table of Contents (continued)

List of Figures

List of Tables

List of Tables

Trademarks and Registrations

Registrations and trademarks used in this document include:

™ AXP	Trademark of Digital Equipment Corporation.
™ Alpha	Trademark of Digital Equipment Corporation.
™ DEC	Trademark of Digital Equipment Corporation.
™ Ethernet	Trademark of Xerox Corporation.
™ HP-UX	Registered trademark of Hewlett-Packard Company.
™ HP9000	Trademark of Hewlett-Packard Company.
® INFI 90	Registered trademark of Elsag Bailey Process Automation.
® INFI-NET	Registered trademark of Elsag Bailey Process Automation.
® Intel	Registered trademark of Intel Corporation.
® Microsoft	Registered trademark of Microsoft Corporation.
™ MS-DOS	Trademark of Microsoft Corporation.
™ Open VMS	Trademark of Digital Equipment Corporation.
™ VAX	Trademark of Digital Equipment Corporation.
™ VMS	Trademark of Digital Equipment Corporation.
® Windows	Registered trademark of Microsoft Corporation.
™ Windows NT	Trademark of Microsoft Corporation.

SECTION 1 - INTRODUCTION

OVERVIEW

The Strategic Enterprise Management Application Programming Interface (semAPI) software is a library of functions that provides software application programs access to INFI 90 OPEN control system information.

The semAPI functions along with Elsag Bailey computer interface modules form the link between host computer application and control system. This versatile link enables a host computer to perform the following tasks to the INFI 90 OPEN system:

- Process monitoring and control.
- System status monitoring.
- Process control unit tuning (SC level only).

Communication protocols between the host and the computer interface include RS-232-C and Small Computer System Interface (SCSI). The host computer interacts with the computer interface by issuing a command and receiving a reply. The semAPI software consists of a function library and a device driver. The semAPI software along with the customer created application program reside on the host computer (see Figure 1-1).

INTENDED USER

The user of this software should be a process engineer who has thorough knowledge of the INFI 90 OPEN system, Elsag Bailey function codes, C or C++ programming language and the host platform.

FEATURES

- Support for RS-232-C or SCSI communication between the semAPI software and the computer interface modules.
- Multiple applications can communicate concurrently with a single computer interface.
- Remote computer interface capability.
- semAPI functions remain the same regardless of the platform.

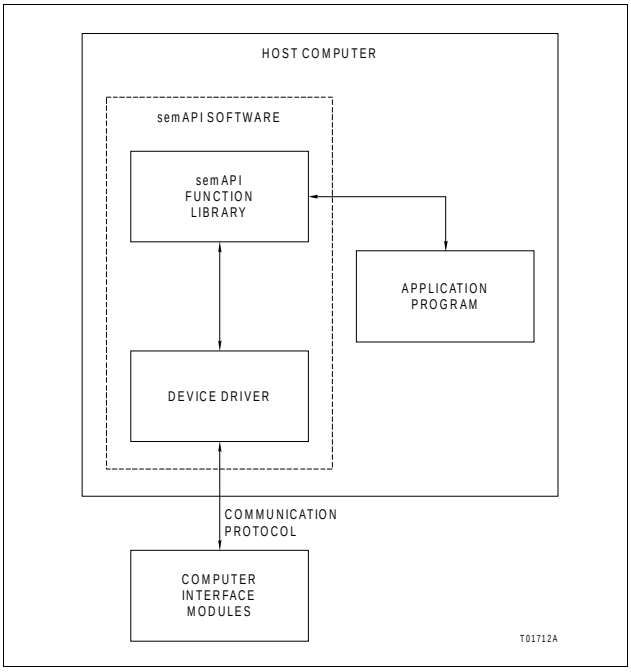


Figure 1-1. Block Diagram

INSTRUCTON CONTENT

This instruction contains information about hardware and software setup for the semAPI software. Detailed information is provided for each function. This instruction contains the following sections:

Introduction Provides an overview of the semAPI software package. Other items include intended user information, software features, glossary of terms and abbreviations (Table 1-1), a list of reference documents (Table 1-2), nomenclature of the software (Table 1-3) and related hardware (Table 1-4), and a control level reference table (Table 1-5).

Description and Operation Contains a functional and hardware description of the semAPI software package. Other items include communication protocol, hardware configuration, host software and hardware requirements, and communication parameters.

Installation Outlines the software installation process.

Software Description and Operation This section includes computer interface and INFI 90 OPEN system overviews, programming basics, general code description, platform variations, and library organization. Other topics include file list, compiling, linking, performance considerations and a software checklist.

Software Details This section details the computer interface architecture and describes the semAPI functions: PCU, read, write, time, manager and miscellaneous functions. This section also provides descriptions on the decoding and error handling information.

Function Library This section details each of the semAPI functions.

Test Program Provides information about the TALK90 test program.

Application Examples Provides a sample program that can be used as an example or as a model.

Troubleshooting This section lists troubleshooting information.

Appendices The appendices describe in detail, the various structures referenced in Section 6.

HOW TO USE THIS INSTRUCTION

1. Read the introduction and the description and operation sections. Reading these sections helps the user understand the semAPI software package and how it fits in with the INFI 90 OPEN system and the application program.

2. Follow the installation steps in Section 3.

3. Read Section 4 for a description of the semAPI software and Section 5 for detailed information about the software.

4. Refer to the function library in Section 6 when programming and use Section 7 when testing functions.

5. Refer to Section 8 for a programming model if difficulties occur or Section 9 for troubleshooting information if communication problems arise.

6. Refer to the appendices as needed while programming.

DOCUMENT CONVENTIONS

This document uses standard text conventions throughout to represent keys, user data input and display items.

KEY Identifies a keyboard key.

Example: Press **Return**.

Display item Any item that displays on the screen appears as italic text in this document.

Example: *$ DEFINE/SYSTEM.EXEC ICI$DISK "drive1$dia1:"*

File name Any file names and file extensions appear as bold-italic text.

Example: ***ici_err.h***

Italic Identifies a variable parameter in a command line.

Example: *source, Username, Password, filename.ex*

USER INPUT Indicates a fixed input that must be entered exactly as shown.

Example: Type **LOGIN**.

[ ] A user input that is optional. Text within the brackets still follow the previously described conventions.

GLOSSARY OF TERMS AND ABBREVIATIONS

Table 1-1 contains terms and abbreviations that are unique to Elsag Bailey or have a definition that is different from standard industry usage.

Table 1-1. Glossary of Terms and Abbreviations

Term	Definition
Block Data	Function block specifications.
Block Value	A digital or analog value of a function block output.
Connect	Connection is the mechanism which enables exception reports to flow between INFI 90 OPEN nodes. Each node maintains records of exception report routing information. Connecting a point enables exception reports, but generally does not cause them to occur. The computer interface point table consists of points that have been established by the host computer.
Configuration	The act of setting up equipment to accomplish specific functions or a list of parameters associated with such a setup.
Control Output	The control system signal that influences the operation of a final control element.
DAANG	Data Acquisition Analog.
Device Driver	A INFI 90 OPEN system function block that provides control for a single input device.
EWS	Engineering work station.
Exception Report	Information update generated when the status or value of a point changes by more than a specified significant amount; abbreviated as XR.
Executive Block	Fixed function block that determines overall module operating characteristics.
Fatal Error	An unexpected failure or error that prevents the module from executing the configuration. An error that causes a device to go into a fail mode and sends its output to a defined value.
Function Block	The occurrence of a function code at a block address of a module.
Function Code	An algorithm which manipulates specific functions. These functions are linked together to form the control strategy.
INFI-NET	Advanced data communication highway.
ICI	INFI 90 OPEN Computer Interface. A standard Elsag Bailey interface for communications between a host computer and an INFI 90 OPEN network (formally called a CIU).
Loop	A data communication network with ring topology.

Table 1-1. Glossary of Terms and Abbreviations (continued)

Term	Definition
Module Address	A unique identifier of a specific device or a communication channel. Refers to Controlway or module bus address.
Multistate Device Driver	An INFI 90 OPEN system function block that provides control for a dual input device.
NC	Network Connect. The low level platform specific communications library that is transparent to a host application program.
Network Interface	Term for all local and remote interfaces, computer interfaces, and console interfaces to the INFI-NET communication system (synonymous with ICI).
Node	A point of interconnection to a network.
Node Address	A unique identifier of a specific device or a communication channel. Refers to Plant Loop or INFI-NET address.
Not-Set-Permissive	A remote switch input to remote control memory function block that disables remote switch set commands.
Override (OVR)	A remote switch input (RCM function block) that controls the block output under certain conditions.
Plant Loop	Network 90 data communication highway.
Point Index	An integer value that specifies a particular entry in the computer interface point table. Indexes are also referred to as points.
Point Table	The computer interface database that contains the status and value of a point.
Process Variable	An input that is used by the control strategy of a control device.
Ratio Index	Analog input to a INFI 90 OPEN station function block. The ratio index variable may be both read and written by the host through the computer interface.
RCM	Remote Control Memory. A function code that implements a logical switch function.
Read Point or Import Point	A point established in the computer interface with **ESTABLISH POINT**. Read points receive exception reports from function blocks configured in other INFI 90 OPEN nodes.
Red Tag	A logical flag used to indicate a point is out of service. Any value reported from this point is not valid until the red tag is removed.
Remote Control Memory Block	A INFI 90 OPEN function block that implements a logical switching function. The computer interface allows the host computer to read and set RCM points in remote process control units through the RCM read point type.
Remote Control Memory Report	An computer interface point type that allows the host to simulate an RCM function block configured in the computer interface.
Remote Manual Set Constant (RMSC)	A function code which generates exception reports consisting of a status and an analog value.
Reply Code	The reply to an interface command indicating whether or not the command was accepted and performed or rejected and not performed.
Report Point, Write Point, or Export Point	A point established in the computer interface with **ESTABLISH REPORT.** These points appear as function blocks configured in the host computer to other INFI 90 OPEN nodes. This enables the host computer to send exception reports to modules which have established and connected exception report routes.
Sample Number	A sequence number used to synchronize trend data with INFI 90 OPEN system time.
semAPI	Strategic Enterprise Management Application Programming Interface.

Table 1-1. Glossary of Terms and Abbreviations (continued)

Term	Definition
Specifications or Exception Report Specs	INFI 90 OPEN function block characteristics are determined by specification values. These are entered into the module when the point is configured, or in the case of tunable specifications, when the function block is tuned. Specifications are sent out to other modules when exception report routes are established, and when the specifications are changed.
Set Point	Target set for a process variable or standard representing desired value of the process variable.
Watchdog Timer	A timer set to check on the progress of an operation in the computer interface. For example, the computer interface uses a watchdog timer to periodically check the status of host computer communications. If this option is enabled in the **RESTART** command and the host computer has not communicated with the computer interface during the watchdog timing period, the computer interface goes off-line.
Work Flag	Two status bytes. The bits of the work flag are encoded to provide the host computer with information about the receipt of time sync messages, specifications, and exception reports by the computer interface.

REFERENCE DOCUMENTS

Table 1-2 lists the documents referenced in this instruction.

Table 1-2. Reference Documents

Document Number	Document Title
I-E96-610	INFI-NET to Computer Interfaces (INICI01/03)
WBPEEUI210004A0	Function Code Application Manual

NOMENCLATURE

Table 1-3 lists the semAPI product nomenclature. Table 1-4 lists the nomenclature that can be associated with the semAPI software package.

Table 1-3. Nomenclature (semAPI)

Position	1	2	3	4	5	6	7	8	
	L	I	A	P	I				**semAPI (Application Programming Interface)**
						1			**Operating System** HP-UX
							0		**Control Level** Data Acquisition
							1		Supervisory Control
								1	**Client Licensing** One User/Client
								5	Five Users/Clients
								T	Ten Users/Clients

Table 1-4. Nomenclature (Associated Products)

Nomenclature	Description
INICI03 (includes):	INFI-NET to Computer Interface
INICT03A	INFI-NET to Computer Transfer Module
INNIS01	Network Interface Slave Module
IMMPI01	Multi-Function Processor Interface Module

semAPI FUNCTION LEVELS

The semAPI software is available in two functional levels: data acquisition and supervisory control.

Data Acquisition (DA) Provides data acquisition and process monitoring capabilities. The DA level allows an application to read data from an INFI 90 OPEN system.

Supervisory Control (SC) Provides data acquisition, process monitoring and supervisory capabilities. The SC level allows an application to read and write data to and from an INFI 90 OPEN system.

Table 1-5 provides a complete list of semAPI functions along with the associated functional level (DA or SC) of the function.

Table 1-5. Function Control Levels

Function	Control Levels	
	Data Acquisition	Supervisory Control
Callup	X	X
Cancel quick message	X	X
Connect point group	X	X
Connect point list	X	X
Connect to logical computer interface	X	X
Demand module status		X
Disconnect from logical computer Interface	X	X
Disconnect point group	X	X
Disconnect point list	X	X
Disestablish point	X	X
Enable management	X	X
Environment	X	X
Establish export point	X	X
Establish import point	X	X
Force restart	X	X
Get command exceptions		X
Get data exceptions	X	X
Hangup	X	X
ICI error text	X	X

Table 1-5. Function Control Levels (continued)

Function	Control Levels	
	Data Acquisition	Supervisory Control
On-Line/Off-Line	X	X
Output control point		X
Output report	X	X
Read block		X
Read command exceptions		X
Read data exceptions	X	X
Read data group	X	X
Read data list	X	X
Read data specs		X
Read enhanced block output		X
Read ICI status	X	X
Read performance data	X	X
Read system date and time	X	X
Read work flag	X	X
Regenerate specs		X
Restart	X	X
Retrieve quick message	X	X
Set system time and date		X
Trend data poll	X	X
Tune block		X

SECTION 2 - DESCRIPTION AND OPERATION

INTRODUCTION

This section contains functional and hardware descriptions of the Strategic Enterprise Management Application Programming Interface (semAPI) software package. Other items include; communication protocol, computer interface setup, software and hardware requirements, and communication parameters.

FUNCTIONAL DESCRIPTION

The semAPI functions link a host computer application with an INFI 90 OPEN control system. The functions allow a host computer application to obtain and use information from the control system.

The semAPI functions can use the Elsag Bailey computer interface module INICI03 INFI-NET to Computer Interface.

Refer to Table 2-1 for more information on the communication interface modules. Figure 2-1 shows an INICI03 INFI-NET Computer Interface connecting a host computer to an INFI 90 OPEN control system. A host application can communicate

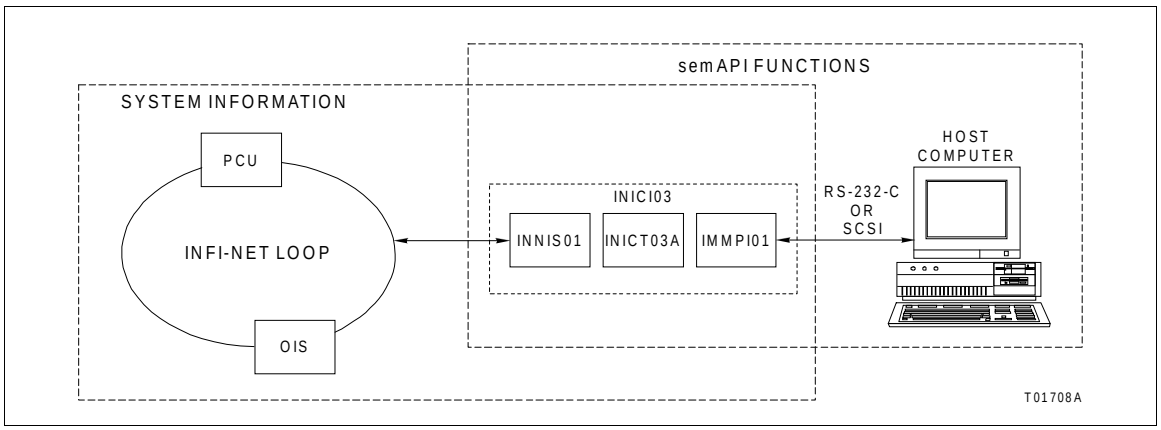


Figure 2-1. Interface Block Diagram

with different types of Elsag Bailey computer interface modules over different communication protocols through a single set of semAPI functions.

HARDWARE DESCRIPTION

The following describes the computer interface hardware module set.

> **NOTE:** Refer to the appropriate product instruction for detailed information about the interface modules. Refer to Table 1-2 for product instruction numbers.

The INICI03 computer interface consists of three modules. These modules are:

- INICT03A INFI-NET to Computer Transfer Module.
- INNIS01 Network Interface Slave Module.
- IMMPI01 Multi-Function Processor Interface Module.

This computer interface supports a maximum of 30,000 points (depending on point type) and supports RS-232-C and SCSI communication protocols. Points are the values of function blocks (stored in the computer interface) from control modules in the INFI 90 system. Values generated by the host computer and stored in the computer interface are also considered points.

COMMUNICATION PROTOCOL

Table 2-1 lists the communication protocols supported by the semAPI software and the communication modules that support those communication protocols.

Table 2-1. Communication Protocol and Module Support

Platform	Communication Protocol	Communication Module
HP-UX	RS-232-C, SCSI	INICI03

TYPICAL INTERFACE HARDWARE CONFIGURATION

This section shows a sample hardware configuration option. It is intended as an example and may not represent the exact connections for an application (refer to the appropriate computer interface instruction for exact connections). Figure 2-2 shows multiple INICI03 interfaces connecting an INFI-NET Loop to a host computer.

> **NOTE:** Multihost configurations require one license for each host computer.

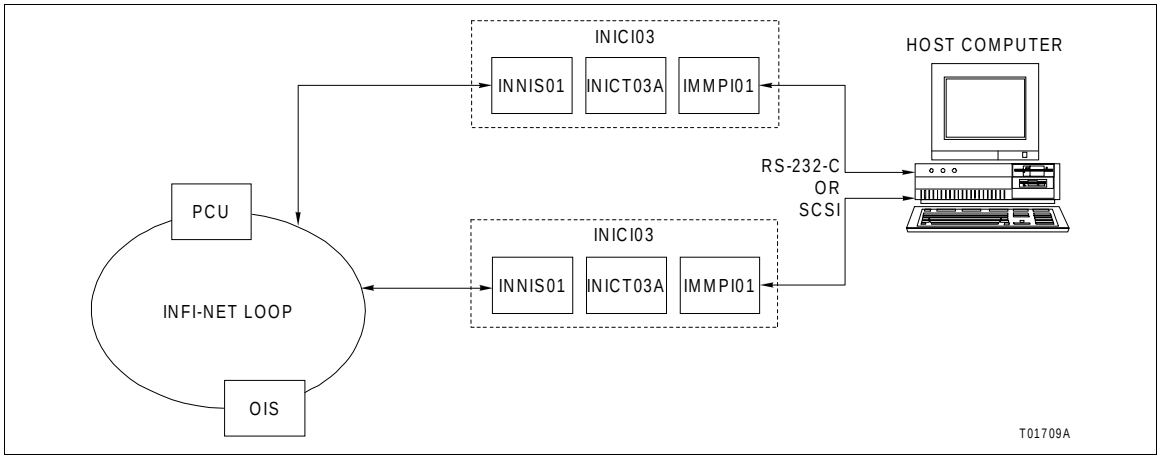


Figure 2-2. Multiple ICI Interfaces

HOST COMPUTER SOFTWARE AND HARDWARE REQUIREMENTS

The following information lists host computer software and hardware requirements when using semAPI functions. General hardware requirements:

- 2.5 megabytes of available disk space. Includes header files, object libraries, linked executables and a configuration file for two computer interfaces.

- Memory requirements are dependent on the number of semAPI functions used within the user application. The TALK90 application (which uses all semAPI functions) executes successfully on machines with 16 Megabytes of Random Access Memory (RAM) (500 kilobytes of conventional memory). Actual memory requirements are directly dependent on the user application.

Specific hardware and software requirements:

- HP 9000 computer, 7200 series microprocessor or less (single CPU).

- HP-UX (A.09.03 or later).

COMMUNICATION PARAMETERS

The application program that is developed by the user is called the client application. The client application talks to the Elsag Bailey computer interface modules (i.e., ICIs) through the device driver program called the server. There are two lines of communication in a typical application:

1. The first line of communication is between the application program (the client) and the driver program (the server). The

user determines whether to use a network or a local protocol during configuration.

2. The second line of communications is between the driver program (the server) and the Elsag Bailey computer interface modules.

3. Both lines of communication are set using ICICONF.EXE.

For detailed information about configuring communication parameters, refer to installation instructions in Section 3.

SECTION 3 - INSTALLATION

INTRODUCTION

This section describes the installation process of the Strategic Enterprise Management Application Programming Interface (semAPI) software.

A software key is included with the installation media. The software key must be installed correctly in order to use the semAPI software. Refer to Appendix N for details.

> **NOTE:** Before installing the software verify that the host computer meets all hardware and software requirements. Refer to ***HOST COMPUTER SOFTWARE AND HARDWARE REQUIREMENTS*** in Section 2.

INSTALLATION

The installation section consists of an installation overview and an installation procedure.

Installing the Software Key

The semAPI software package comes with a software key that plugs into the termination device. The software key is required in order to use the semAPI software. Refer to Appendix N for installation details.

Overview

This section provides an overview of the installation process and lists the requirements in order to perform the installation procedure. For detailed installation instructions refer to ***Installation*** later in this section.

1. Install the software key. Refer to Appendix N for information.

2. Create a root directory for the semAPI software.

3. Issue the command to extract the semAPI software from the provided ***TAR*** file on the distribution media.

4. Invoke the installation procedure to create the directory structure and move the extracted files of the semAPI software into the appropriate directories.

5. Create the serial port devices.

6. Configure the services file.

7. Source one of the included shell scripts depending on the shell being used by the particular user.

8. Create a symbolic link to the server portion of the semAPI software package.

9. Run the ***ICICONF*** program to define the logical link between the computer interface modules (as they are referred to in an application program) and the actual physical port number to be used in the interface module.

Installation

The semAPI software is organized into the following directory hierarchy:

/ICI — Main ICI directory with several subdirectories.

/ICI/LIB — Library subdirectory containing the object libraries.

/ICI/EXE — Executable subdirectory containing the linked executable modules. This directory contains ***TALK90.EXE***, ***ICICONF.EXE*** and ***DEVICE.EXE***.

/ICI/SOURCE — Source subdirectory where user application source code resides. A sample application program is included in this directory called ***SAMPLE1.C***.

/ICI/INCLUDE — Header subdirectory containing the header files for the semAPI software package.

The provided tape contains one ***TAR*** file set. This set will contain the header files (also called include files), the object libraries, executables and an installation shell script. Use the following procedure to install the semAPI software.

To install:

1. Install the software key. Refer to Appendix N for information.

> **NOTE:** You must be logged into the **root** account for the remainder of the installation procedure.

2. Create the root directory where the software is to be loaded.

   a. At the prompt type:

   cd / ENTER

b. Type:

umask 000 ENTER
mkdir ici ENTER

c. Type:

cd ici ENTER

3. Load the semAPI installation procedure from the distribution media. Mount the distribution media in the drive.

At the prompt type:

tar xf /dev/rmt/0m ENTER

4. Invoke the shell script to create the directory structure and move the semAPI files to the appropriate location in the directory structure.

At the prompt type:

sh ./install.sh ENTER

NOTE: The install shell script has installed the semAPI software into the directory hierarchy with the file protections set such that all users have read, write, and execute privileges on all files. The system administrator may decide to change the file protections to restrict access to the semAPI software as needed.

5. Create the serial port devices. The actions taken in this step depend on the actual revision of the HP-UX operating system. You must have **root** privileges to create the serial port devices.

a. Create the ttya serial port. At the prompt type:

mknod /dev/ttya c 1 0x204004 ENTER

b. Determine the version of the HP-UX operating system that is running on the system. At the prompt type:

uname -r ENTER

c. Create the ttyb serial port.

HP-UX version A.09.01 and A.09.03, type:

mknod /dev/ttyb c 1 0x205004 ENTER

HP-UX version A.09.05, type:

mknod /dev/ttyb c 1 0x502004 ENTER

d. Modify the file protections on the serial ports. At the prompt type:

chmod 666 /dev/ttyb ENTER

chmod 666 /dev/ttya ENTER

6. Configure the services file for communications between the client and server portions of the semAPI package. The actual service numbers in the following examples may be changed if these services are already in use on the target host platform. You must have **root** privileges to modify the /etc/services file. Use your favorite editor to add the following lines to the /etc./ services file:

DD_ttya 20000/tcp

DD_ttya 20000/udp

DD_ttyb 20001/tcp

DD_ttyb 20001/udp

NOTE: These lines are case sensitive, always use capital **DD** and lowercase device names.

7. Source one of the included shell scripts depending on the shell being used by the particular user. You may need to copy the shell script files ***icilog.sh*** and ***icilog.csh*** to a location that is accessible by all semAPI users on the system.

a. For users of the ***sh*** shell the following line should be added to the users ***.profile*** file:

. /ici/icilog.sh ENTER

b. For users of the ***csh*** shell the following line should be added to the users ***.cshrc*** file:

source /ici/icilog.csh ENTER

NOTE: The actual path to the shell scripts may be different if they have been copied to another location on the target host platform.

8. Create a symbolic link called ***DEVICE*** to the server portion of the semAPI software. At the prompt type:

cd /ici/exe ENTER

ln -s device DEVICE.EXE ENTER

NOTE: The ***DEVICE.EXE*** portion of the above command is case sensitive and should be in upper case letters.

9. The application program will reference the interface devices (ICIs) using a logical ICI number. The ***ICICONF*** program is a configuration program that is run which creates a configuration file which is a cross-reference table between logical interface devices and physical device names. Refer to Table 3-1 for an example.

Table 3-1. Cross Reference Table

Logical ICI	Device Name
1	ttya
2	ttyb

The following is a sample run of the ***ICICONF*** program that sets up logical ICI number one:

cd /ici/exe ENTER

./iciconf ENTER

The logical ICI configuration defines all interface devices that the application can communicate with. This refers to interface devices that reside on this machine.

Define ICI Logical Configuration (Y/N) ? **Y**

Enter Logical ICI to update/define (0=exit): **1**

Physical ICI : TTYA:

ICI Node Name :

ICI Network type (0-DECNET,1-TCP/IP,2-LOCAL): **2**

Logical ICI : 1

Physical ICI : TTYA:

ICI Node Name :

ICI Network Type : 2

Physical ICI Backup:

ICI Backup Node :

ICI Backup Net Type: **0**

Is this correct(Y/N)? **Y**

Enter Logical ICI to update/define (0=exit): **0**

The physical properties for each interface device that reside on this machine need to be defined. Interface devices that are on different machines need to be defined on the particular machine.

Examples: VAX: TXA2:, TTA2:, LTA15:, B400:
HP: ttya, ttyb
PC: COM1, COM2

Define ICI Physical Properties for a ICI(Y/N) ? **Y**

Enter the Physical ICI : **TTYA:**

Enter the TCP/IP port number that clients should use when communicating with the ICI module. The port number must be between 1024 and 65535 and must be unique amongst all servers running on this processor. However, a value of 0 is allowed and indicates that no clients will use TCP/IP to communicate with the device.

TCP/IP port for this device: **3001**
(3-Serial, 4-SCSI)
Connection Type (3-Serial, 4-SCSI)
RS-232 Baud Rate : **19200**
Data Bits : **8**
(1 - NONE, 2 - EVEN, 3 - ODD)
Parity : **1**
Stop Bits : **1**
Physical ICI : **TTYA:**
RS-232 baud : **19200**
Data Bits : **8**
Parity : **1 (NONE)**
Stop Bits : **1**
Is this correct(Y/N)? **Y**

NOTES:

1. The TCP/IP port number is always entered when configuring the physical properties of the ICI. This port is actually the LISTEN port for the DD software. A value of zero is also valid if no TCP/IP connections are required to the DD software.

2. The TCP/IP port number is only entered if *ICI network Type* is TCP/IP when configuring the logical properties of the ICI. For client to server communications to function, the same TCP/IP port number must be entered for the logical and physical configurations for a particular ICI. If a network type different from TCP/IP is selected then the prompt for the TCP/IP port is not displayed for the logical configuration parameters.

10. When using TCP/IP as the network type (client/server communication), the host machine must have blind rshell capability on the local node. Enter the host name in ***/etc./hosts.equiv*** (for system wide use) or in the ***.rhosts*** file located in the users home directory for a specific user. To execute type:

rlogin *host_name*

where *hostname* is the name of the local node. If the blind rshell is setup correctly, the user will not be prompted for a password.

SECTION 4 - SOFTWARE DESCRIPTION AND OPERATION

INTRODUCTION

This section includes communication interface and INFI 90 OPEN system overviews, host computer programming basics, general code description, platform variations, and library organization. Other topics include file list, compiling, linking, performance considerations and a software checklist.

INFI 90 OPEN SYSTEM OVERVIEW

Configuring an arrangement of function blocks in the control modules of the system implements a process control strategy. The ***Function Code Application Manual*** (refer to Table 1-2 for document number) describes function blocks and codes in detail. Control modules in the process control unit receive data from the process and perform control functions based on this data. INFI 90 OPEN control modules transfer process values from one node/PCU to another using an exception reporting procedure. Using this procedure, a module (the receiver) wanting data from another module (the sender) requests an exception report route be established to the sender. After this route is established, the sender initiates an exception report whenever the data changes significantly. Communication interface modules access the communication loop of the INFI 90 OPEN system. The Strategic Enterprise Management Application Programming Interface (semAPI) software uses the communication interface modules to establish a link between the host computer and the INFI 90 OPEN system.

COMMUNICATION INTERFACE OVERVIEW

The semAPI software is compatible with the following Elsag Bailey communication interface:

- INICI03 INFI-NET to Computer Interface.

Refer to Table 2-1 for more information on these communication interfaces and modules.

The primary functions of the communication interfaces are to allow a host computer to tune control modules, monitor system data, and control the process. The host computer application can access station variables, function block outputs, module status, control stations and output exception reports by using the semAPI functions to communicate with the communication interface.

HOST COMPUTER PROGRAMMING BASICS

The host computers must be programmed to use Elsag Bailey communication interface modules to perform the desired input and output operations. Some of the basic operations performed by the semAPI functions are:

- Clearing or initializing the communication interface.
- Tuning modules in the process control unit.
- Monitoring system status.
- Reading INFI 90 OPEN data.
- Writing data to the INFI 90 OPEN system (supervisory control).

Clearing the Communication Interface

The **RESTART** command clears the communication interface point table of data points and initializes the communication interface (refer to Section 6 for details).

Tuning Control Modules

Tuning of control modules within a process control unit can be accomplished using the semAPI function **TUNE BLOCK.**

Monitoring System Status

The communication interface receives system status information from the various INFI 90 OPEN modules. Using this information, the host computer can monitor the system.

Processing I/O Data

A host computer application program asks the communication interface for data. After receiving the data, the application program processes the data. It can also output data to the INFI 90 OPEN modules controlling the process through the semAPI functions.

GENERAL CODE DESCRIPTION

All of the semAPI functions are written in ANSI C. There is a call for each command available in the semAPI set. Refer to Section 6 for detailed information on each semAPI function.

CODE COMMONALITIES

All commands are called in the same manner:

status = command *(logical_ici_number,*
user_parameters,

user_data,
user_data_length,
error_structure)

Figure 4-1 represents the various types of data elements passed between the user application program and the semAPI function library. The application program calls a function from the semAPI function library.

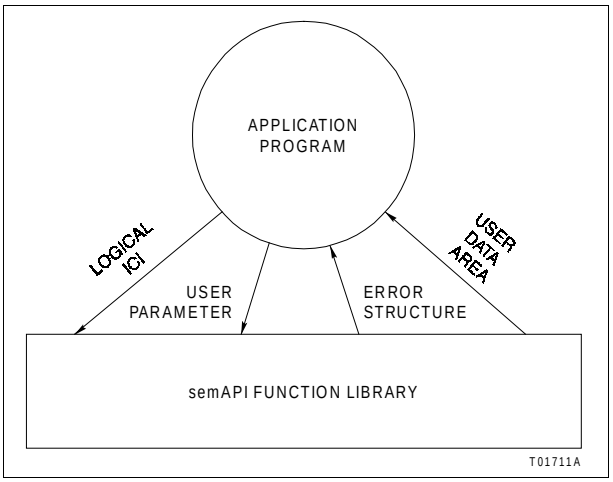


Figure 4-1. Application Program and Function Library Elements

Each function returns a status and an error structure. The status and error structure notifies the application program of problems encountered while attempting to issue the semAPI function. Refer to Appendix I for details about error structures. The user parameter structure is the data that the application program must provide to the semAPI function library in order to issue the command. The user data area structure is the data the semAPI function library returns to the application program. The user must tell the library the size (in bytes) of the user data area. This ensures that the function library has enough room to return the necessary data to the application program. Each function has a logical communication interface input parameter. The logical ICI parameter defines which communication interface the issued command addresses.

Command or Function The semAPI function call (i.e., **READ DATA EXCEPTIONS**).

Return Status Status from the semAPI function call (refer to Appendix I).

Logical ICI Defines which communication interface the semAPI function addresses.

User Parameter The structure containing the input parameters to the semAPI function library. These items must be provided by the user in order for the semAPI function library to issue the command properly.

User Data Area The structure containing the return data for the application program. This is the location for returned data from the semAPI function library.

User Data Length Specifies the size (in bytes) of the user data area. This size is required so that the semAPI function library knows that there is enough room to return the data to the application program.

Error Structure The structure that contains error information. This information assists the users when troubleshooting and debugging the application program. Refer to Appendix I and Section 9 for more information.

PLATFORM VARIATIONS

The semAPI functions are identical across all platforms. Variations in the code occur in lower level code which does not affect user calls to the semAPI library.

LIBRARY ORGANIZATION

The semAPI functions comprise the library. The semAPI function set is composed of the following parts:

- Object module libraries.
- Header files.
- Sample application.
- Link procedure.

FILE LIST

Table 4-1 lists the files that comprise the HP-UX semAPI function set.

NOTE: The user may have selected an alternate organization for the files at installation time

Table 4-1. semAPI Function Files (HP-UX)

Directory	File Name	Description
/ici	icilog.csh	Shell script containing the environment variables that point to the location of the various types of files. This is used for the CSH shell.
	icilog.sh	Shell script containing the environment variables that point to the location of the various types of files. This is used for the SH shell.
	install.sh	Shell script for installing semAPI software. This procedure makes the semAPI directory tree, moves the files to the correct locations, and links the executable programs.
/ici/exe	device	The executable program for the server task.
	iciconf	The executable program for the ICI configuration program.
	minirouter	The executable program for the buffer task.
	talk90	The executable program for the TALK90 program (refer to Section 7).

Table 4-1. semAPI Function Files (HP-UX) (continued)

Directory	File Name	Description
/ici/include	ICI_ERR.H	Definition of error structure (ERR_STRUCT).
	ICI_TIME.H	Definition of time structure (TIME_STRUCT) and valid time units.
	ICI_USER.H	Definition of point types, work flag (WORK_FLAG), time stamp (ICI_TIME_STAMP), and INFI 90 OPEN address (INFI90ADR).
	L3ATABLE.H	Definition of a value table used for analog point values.
	L3BTABLE.H	Definition of a status table used for digital point values.
	L3GMI.H	Definition of the structures and prototypes needed for the GMI commands.
	L3QUAL.H	File containing generic #defines for several semAPI functions.
	L3ICCONF.H	Definition of the structures and prototypes for the communication interface configuration functions.
	L3MANG.H	Definition of the structures and prototypes for the communication interface manager functions.
	L3MISC.H	Definition of the structures and prototypes for the miscellaneous functions.
	L3PCCONF.H	Definition of the structures and prototypes for the PCU configuration functions.
	L3PTABLE.H	Definition of the problem report table.
	L3READ.H	Definition of the structures and prototypes for the read functions.
	L3STABLE.H	Definition of the specification table.
	L3TIME.H	Definition of the structures and prototypes for the time functions.
	L3TTABLE.H	Definition of the trend table.
	L3WRITE.H	Definition of the structures and prototypes for the write functions.
	PLATFORM.H	This header file contains the #define for the platform that the semAPI function set has been built on. If the #define is a 1 then that is the platform that the function library has been built on.
	l3groups.h	Definition of structures and prototypes for the read list and group functions.
/ici/lib	libicida.a libt90da.a ***- or -*** libicisc.a libt90sc.a	Object libraries containing semAPI Data Acquisition (DA) functions. Object libraries containing semAPI Supervisory Control (SC) functions.
ici/source	sample1.c	Sample application program that establishes a connection to the communication interface, establishes a few points and does a read data list on the established points.

COMPILING

The semAPI function set does not come with a compilation command procedure. Use the standard C compiler when compiling an application program.

An application program must be compiled using the HP-UX C compiler. The command line that follows can be used to compile an application program.

```
cc -c -Aa -I/ici/include userapp.c
```

LINKING

The following text provides linking instructions.

The example command line can be used to link an application program to the appropriate libraries.

Data Acquisition (DA) linking:

```
cc -o userapp userapp.o -L/ici/lib -/licida
```

Supervisory Control (SC) linking:

```
cc -o userapp userapp.o -L/ici/lib -/licisc
```

Internal Use (IU) linking:

```
cc -o userapp userapp.o -L/ici/lib -/liciiu
```

PERFORMANCE DATA

There are several factors that determine the performance of the semAPI functions in an application. These include:

- Number of computer notifies.
- Communication interface throughput to the computer.
- INFI 90 OPEN loop loading.
- Computer loading of the communication interface I/O program.
- Host computer I/O throughput.
- Number of I/O channels on host computer.
- Types of exceptions and time stamping.
- Host computer CPU performance.
- Host computer loading caused by other processes.

Because of these factors, it is very difficult to obtain performance data for every possible type of system configuration. Table 4-2 shows performance test numbers. The SCSI andRS-232-C tests were run on a local computer directly connected to an INICI03 module. The SCSI testing used SCSI-I.

The RS-232-C communication parameters were:

Baud rate: 19200
Data bits: 8
Stop bits: 1
Parity: none

Table 4-2. Performance (Exception Reports per Second)

Platform	Protocol			
	RS-232-C	SCSI	TCP/IP	DECnet
HP-UX	130	1000	Future	N/A

SOFTWARE CHECKLIST

1. Verify the version number the disks.

2. Verify that the library received is for the platform specified.

3. Load the library to the platform via the instructions in Section 3.

4. Verify that the directory structures created on the platform correspond to the directory structures listed in the appropriate installation procedure in Section 3.

5. Connect a terminal to the ICI diagnostic port and enable ICI command/reply sequence, if available.

> **NOTE:** To enable the command/reply sequence for INOSM01 module refer to **Open Systems Manager** instruction.

6. Read the instructions and sample program in Section 8 for developing an application. Compile and link the application program (refer to ***LINKING***).

7. Verify any errors returned from the program against the errors listed in Section 9 and Appendix I.

SECTION 5 - SOFTWARE DETAILS

INTRODUCTION

This section describes the computer interface architecture. Other information provides configuration descriptions of the computer interface, PCU, read, write and time functions. This section also provides some error structure information.

COMPUTER INTERFACE ARCHITECTURE

The following text describes the overall Strategic Enterprise Management Application Program Interface (semAPI) software architecture. From the top down, the application resides on the host computer. The application communicates to the computer interface modules by making calls to the semAPI software.

The semAPI calls are functions, such as **RESTART** or **ESTABLISH IMPORT POINT**. The entire function library is detailed in Section 6.

If the application requires security features the application can use the manager functions.

The client coordinates multiple message transfers between itself and one or more computer interfaces. The client is linked in with the application and the semAPI functions.

Collectively the application program and semAPI functions make up the client portion of the computer interface client/server architecture. Refer to Figure 5-1 for graphical representation of the client/server model.

The client communicates to the server. This communication is direct linked or linked across a network via network connect functions. The server coordinates multiple applications communicating to a computer interface module. There is one server per computer interface.

The server connects to the computer interface via a network connect layer. The network connect is platform and communications media specific, such as SCSI,or RS-232-C).

The semAPI software allows for multiple applications to be connected to multiple computer interfaces, or a single computer interface. The semAPI software also allows a single application to be connected to a single or multiple computer interfaces.

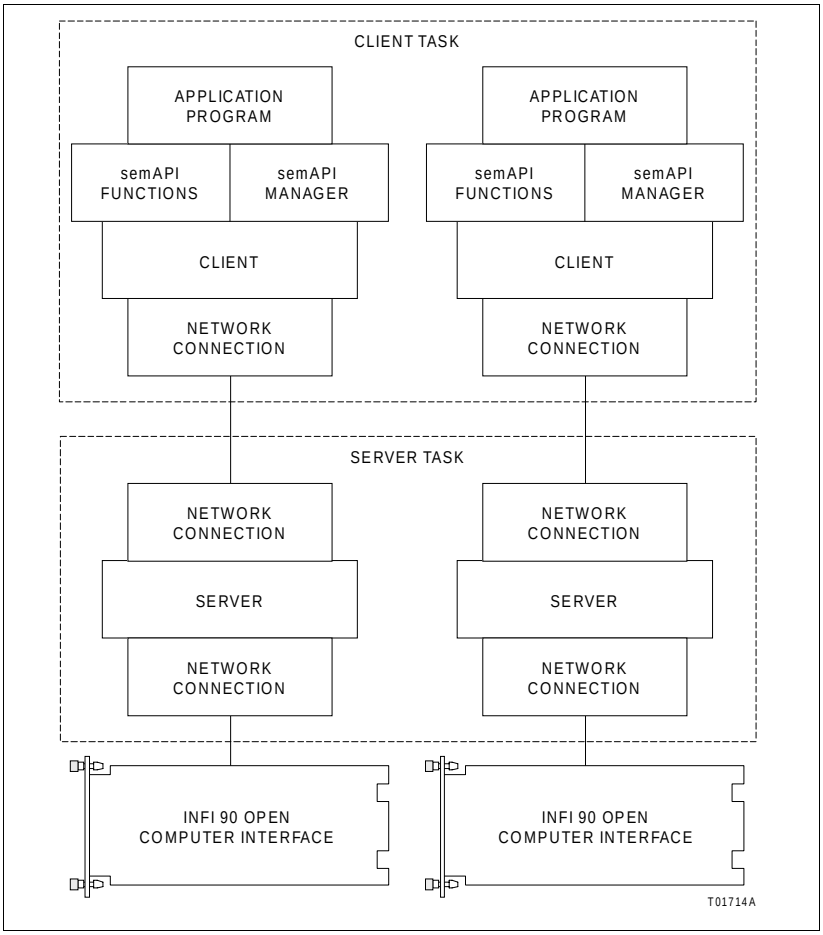


Figure 5-1. Computer Interface Architecture

The client and server layers can be closely coupled to allow direct, high-speed, one-to-one access from the applications to the computer interface or networked to allow multiple connections and flexibility.

The overall multilayer architecture allows maximum flexibility in client/server applications.

FUNCTION DESCRIPTION

The entire function library is described in Section 6. The section is categorized into the following groups:

- Computer interface configuration.
- PCU.
- Read.
- Time.
- Write.
- Manager.
- Miscellaneous.

There are three ways to make semAPI calls:

- Wait access method.
- Quick access method.
- Internal access method.

Using the wait access method, the user issues a semAPI function call (supplying the appropriate parameters). The application ***blocks*** and awaits a response from the computer interface. After a response is available the user data area is returned. Errors encountered are indicated by the return status. If errors occur, detailed information is obtained by examining the error structure.

Using the quick access method, the user issues a semAPI function call (supplying the appropriate parameters). The function passes a message number back to the application. The application is free to process other items. Quick messages are maintained in the system until an application issues **RETRIEVE QUICK**, **CANCEL QUICK** or exits memory. The application program calls **RETRIEVE QUICK** passing the message number and address of the user data area. If a response is ready, the user data area is filled in. If a response is not ready, the ICI_CONTINUE status is returned. If errors occur, detailed information is obtained by examining the error structure.

Calls using the internal access method, only apply to read exception report functions. It is possible that the computer interface will pass back more exceptions than the user data area has room for. If this occurs, a warning message from the function call (**SUB_MORE_EXCEPTIONS**) is returned. Use the ***get*** functions (refer to Section 6) to access the internal buffer of the function library, and clear the buffered exceptions.

1. Issue **READ EXCEPTION REPORT** call.

2. If a *SUB_MORE_EXCEPTIONS* warning is returned, then issue **GET EXCEPTION REPORT** until *no exceptions* are returned.

3. After resolving, issue **READ EXCEPTION REPORT** again.

> **NOTE:** An application may use **READ WORK FLAG** to determine when its time to issue **READ EXCEPTION REPORT**.

A typical calling sequence for an semAPI function is as follows:

Invoke the semAPI function. The parameters that are passed are subject to validation testing. The validation process checks to see if the logical computer interface connection is established. If the function finds an invalid parameter, it immediately returns an error status (*SUB_INVAL_ARGS*) without attempting to send any message to the computer interface. If

the parameters are valid, the message is built and sent to the computer interface module.

If the function was sent using the quick method the user must issue **RETRIEVE QUICK** (s_retrieve_quick_reply) with the message number to receive the appropriate information. The **RETRIEVE QUICK** will pass back a status of *ICI_OK, ICI_WARNING, ICI_FATAL* or *ICI_CONTINUE*. The *ICI_CONTINUE* status indicates that the message is not available yet.

If the wait version of the function is chosen the function waits for replies from the server. When a response is received, it is decoded immediately and returned to the calling application.

In either the wait or quick case, when the response is available, the return status to the semAPI function (wait method) or the return of **RETRIEVE QUICK** (quick method) will indicate the success (*ICI_OK*) or failure (*ICI_WARNING* or *ICI_FATAL*). The user should examine the error structure if either *ICI_FATAL* or *ICI_WARNING* is returned.

Exception reporting is handled in a unique way. Multiple users can sign up on the exception reports list. Every user that signs up will get all of the exception reports requested by any of the users. Therefore getting exceptions becomes an unsolicited message.

For an application to use the exception reporting scheme the application must sign up to receive exceptions. To sign up for exceptions, the user must call one of the read exception reports functions (refer to Section 6). After calling, the user is now registered to receive exception reports via unsolicited messages.

Registration is by type of exception desired. Possible choices are:

- Data exceptions.
- Spec exceptions.

COMPUTER INTERFACE CONFIGURATION

The function library is described in Section 6. The functions contained in the library under ***ICI CONFIGURATION*** in Section 6 perform initialization and set up which include tasks such as:

- Start-up and end.
- Point table.
- Status.

RESTART starts up the computer interface. **CALLUP** and **HANGUP** enables and disables the password feature of the computer interface.

The point table functions are used to establish and disestablish import and export points and to connect the routes to the points already established in the computer interface.

Establish Import Data Points

For an import point to begin acquiring data from function blocks, an exception report route must be established to the remote point. To establish import points use **ESTABLISH IMPORT POINT**. These functions also assign the point an index number in the computer interface point table.

The index number assigned to a point is the means by which the computer interface and host computer associate a point.

There are no restrictions on the index number assignments except that they be unique and in the range of one to 30,000 (for INICI03 computer interface). The module status of the computer interface is automatically established as point index zero. Import points can be disestablished using **DISESTABLISH POINT**. Disestablishing a point removes the exception report route between the control module and the computer interface issuing the function.

> **NOTE:** To build an initial point table, it is recommended that the computer interface be restarted (in the off-line mode), build the host computer data base, and then put the computer interface on-line.

Establish Export Data Points

Use **ESTABLISH EXPORT POINT** to establish an export point in the computer interface. This function assigns the point an index number in the computer interface. The index number assigned to an export point is the means by which the computer interface and host computer associate a point. Export points can be disestablished using **DISESTABLISH POINT**. Disestablishing an export point will remove the exception report route from the point.

Establishing to an Export Data Point (From other INFI 90 OPEN Network Interfaces)

Other network interfaces can establish exception report routes to an export point. Upon establishing the exception report route, specifications will be received by the network interface. Since export points are treated like function block outputs, specify the point index number as the function block when establishing exception report routes to export points. The module number must be two and the loop and node addresses must be that of the computer interface containing the export

point. When accessing computer interface indices in an INICI03 from another node, configure the point to receive data from module two and block number which is directly related to the index (i.e., index 3 = block 3).

Establish Stations

Establish a station as a single index station read point if the host computer must be able to control the station in any station mode or if station control is not required. A single computer interface point index controls the entire control station. The host computer sends control commands with a source level that will allow control of the station no matter what the station mode (local or computer). The application may also establish the various components as individual points using:

PT_CONTROL_OUTPUT	PT_SET_POINT
PT_PROCESS_VARIABLE	PT_SET_POINT_OUTPUT
PT_RATIO_INDEX	PT_STATION_MODE
PT_RATIO_INDEX_WRITTEN	PT_STATION_STATUS

PCU CONFIGURATION FUNCTION DESCRIPTION

This section describes the function category in Section 6 called ***PCU CONFIGURATION FUNCTIONS***. These functions relate to the tuning of the INFI 90 OPEN PCU via the computer interface.

The host computer can issue **READ BLOCK** while the module is in execute mode. To change the tunable specifications within the reply, use **TUNE BLOCK** to send the function code, along with the tuned specifications, back to the control module. The control module remains in execute mode during the tuning process.

This approach to tuning is suggested for a host computer running application programs that use parameter gains determined by a plant engineer. The disadvantage of this approach is that the plant engineer has to determine and manually change the gains as plant conditions change.

Another approach to tuning is to use the output of a computer interface as an input to an adapt function block (function code 24). Use the resulting value as a gain of the advanced PID controller function block (function code 156). In this manner, an application can dynamically tune the gain of an advanced PID controller function block as it controls the process. This approach is suggested for host computer programs that adaptively control by computing new gains to automatically tune the system.

READ FUNCTIONS DESCRIPTION

This section describes the functions under ***READ FUNCTIONS*** in Section 6. These functions read data from the interface module to the host computer.

The computer interface allows the host computer to acquire field and system data, control field processing, and to output field data for use in control strategies. There are several functions available for reading information.

Reading Data Points

Import data points are used to acquire field and system data from the INFI 90 OPEN system. Import points acquire outputs and statuses from function blocks within control modules.

For an import point to begin acquiring data from function blocks, an exception report route must be established to the remote point. The established functions, in the configuration category, assign the point to an index number in the computer interface point table.

Module status is automatically established in point index zero and is available to be read.

There are three main data acquisition methods: read exception reports, read point lists and reading point groups.

READING EXCEPTION REPORTS

After an import point route has been established, the import point will begin receiving updates (exception reports) from the function block. The function block will issue exception reports to all established routes when one of the following conditions occur:

- The change in the function block output value is greater than that specified by the function blocks significant change parameter.
- The output value of the function block has exceeded any limits (alarm conditions) set by the corresponding function code.
- The output value of the function block has not changed in the time limit specified by the maximum exception report time parameter.

The host computer can read exception reports from the computer interface by using **READ DATA EXCEPTIONS**. Bits in the work flag (Appendix A) indicate that exception reports exist in the computer interface.

When the host computer reads exception reports, it receives them in more or less the same order the computer interface received them from the control module (depending on the function used to read the exception reports). If the computer interface receives a second exception report for a point before the host computer reads the first, the latter or newest value is reported to the host computer. In this manner new exception reports are never lost, but old exceptions are overwritten and thus lost.

The host computer cannot assume that the order the points are returned in is the actual sequence of events. Use the time stamp value returned in **READ DATA EXCEPTIONS** to determine the actual sequence of events. The time stamp value is enabled by **RESTART**.

READING POINT LISTS

The host computer can read the current values of a sequential lists of import points established in the computer interface. A list is specified by its first and last point index. Use the **READ DATA LIST** command to read lists of points

The maximum list size depends on the function and type of computer interface. Some of the functions have restrictions on the types of points for which they will return values. Refer to the particular function in Section 6 for restrictions. When selecting indices for applications that read lists of points, remember that all points in the list must be established as a point type consistent with the read function being performed. When reading lists of points, it is recommended that similar point types have consecutive indices. If an improper point type is included in a requested list, the computer interface will return a reply code *ICI_PT_TYP_INCM* message *(Point Type Incompatible With Command)*. Refer to Section 9 for more information about this message and other reply codes.

There are inherent advantages and disadvantages to reading point lists. The advantage of this approach is that the reading of an organized data base (similar point types having consecutive indices) requires only a few short functions. This is a very fast way of reading point data. When controlling slow changing processes, the host computer could use the exception report screening option and still retain some measure of security by reading lists of points at time intervals substantially greater than the one second allowed by the maximum report time interval. The disadvantage of this approach is that the host computer must read each point at the scan rate established by itself.

READING POINT GROUPS

The host computer can read the current values of a group of import points established in the computer interface. A group is specified by listing each point index individually. Use **READ DATA GROUP** function to read groups of points.

The typical maximum group size is 50 unless the maximum reply size limits the group to 35. Some of the functions have restrictions on the types of points for which they will return values. Refer to the particular function in Section 6 for any restrictions.

If an improper point type is included in a requested list, the computer interface will return a reply code *ICI_PT_TYP_INCM* message *(Point Type Incompatible With Command)*. Refer to Section 9 for more information about this message and other reply codes.

There are inherent advantages and disadvantages to reading point groups. An advantage of this approach is that the point indices can be spread out through the computer interface point table. The disadvantage of this approach is that the host computer must read each point at the scan rate established by itself.

The reading exceptions reports method is good for monitoring and tracking field and system data. The reading point lists and groups methods are better suited for taking instantaneous samples of data.

WRITE FUNCTION DESCRIPTION

This section describes the functions in the category of Section 6. These functions relate to writing data out to the computer interface.

Export points are used by the host computer to output exception report values. These exception reports are sent to all network interfaces that have established exception report routes to the export point. The exception report data can be used just like function block data is used.

TIME FUNCTION DESCRIPTION

This section describes the functions in the ***TIME FUNCTIONS*** category of Section 6. These functions are used to read and set time on the INFI 90 OPEN system via the computer interface.

All modules in the INFI 90 OPEN system must be time synchronized for the INFI 90 OPEN trending features to function properly. A host computer, through a computer interface, can synchronize the system.

The read time functions can be issued at any time to receive the current computer interface time. If the computer interface has been time synchronized by another computer or console, the sync field of the reply will equal one. This informs the host computer that the computer interface time is equal to the system time. This time data is used to set the host computer internal clock. A bit in the work flag (Appendix A) is set to one when the time message is received from another node.

By reading the time, the host computer is synchronized with the INFI 90 OPEN system time, without setting the system time. If the sync field of the reply to the read function is equal to zero, time synchronization did not occur and the computer interface contains a default time.

After the time synchronization has begun, the computer interface (independent of the host computer) continues to keep the remainder of the system in synchronization. Issuing the set function at any time causes the host computer to synchronize the system.

MANAGER FUNCTION DESCRIPTION

This section describes the functions under ***MANAGER FUNCTIONS*** in Section 6. These functions manage points, establish and connect requests. Each application will automatically maintain a point table. These tables automatically reload the computer interface in the case of a failure.

The manager functions handle some of the features of the computer interface communications which are not specifically built in to the semAPI function library but are useful for host applications which are connecting to an INFI 90 OPEN system. The three main areas of added value in the computer interface management functions are:

- Automatic restarts.
- Multiple computer interfaces and host access.
- Security.

Manager functions keep track of the computer interface configuration functions and connection requests. Each application maintains its own point table. This table is used to automatically reload the computer interface with the point table in the case of a failure.

Host Access to Multiple Computer Interfaces

To allow multiple connections to a single computer interface, the device driver provides the coordination and message handling capabilities. The device driver provides unsolicited messages to allow the manager to coordinate the action of all applications that have enabled management. This frees the manager from communicating directly to the other applications. The manager can obtain performance statistics and status information from multiple computer interfaces.

Security

Security is accomplished via a series of tokens that the device driver maintains. All token ownership is on a first-come-first-serve basis.

A token can be owned indefinitely. This allows the manager to request and own all tokens that govern management. The manager will own these tokens until it terminates. If it does not disconnect from the computer interface (abort or otherwise exit), the device driver will timeout the connection if it is inactive, and return the tokens.

Token requests are internal to the functions. The applications will have calls that request ability to restart, for example. They will not access a token directly.

Summary

The following summarizes the manager functions. The following functions will occur in the background and will appear transparent to the user application:

1. The manager keeps track of all points that are established in a computer interface to make it possible to rebuild a point table after an interface module fails.

2. The ICI manager will monitor the status of computer interfaces through the use of a watchdog enabled in the restart command.

3. The manager will monitor the status of an interface through in-line code in s_general_onebyte_reply. This function is called by ***all*** functions of the semAPI function library.

4. The manager provides failure recovery logic that is responsible for keeping a computer interface on-line and running at all times.

5. The failure control logic is responsible for restarting the dead interface and downloading the point table.

6. Automatic re-connect to a device driver that can no longer communicate. This may happen if a remote node is shutdown or a user accidentally stops the device driver manually.

MISCELLANEOUS FUNCTION DESCRIPTION

This section describes the functions under ***MISCELLANEOUS FUNCTIONS*** in Section 6. The functions of this category are not necessarily related to any particular category mentioned earlier.

The **CANCEL QUICK MESSAGE** removes a stored *QUICK* function from the queue. **READ WORK FLAG** returns the work flag from the computer interface. This work flag contains statuses that may be used for information about the computer interface to determine if it is necessary to issue certain functions.

ERROR HANDLING DESCRIPTION

The error handling scheme consists of an error structure and an error status that are returned from each function call. The return status indicates the success or failure of an operation. It also serves as an indicator to the user that additional information about the system has been returned in the error structure. The error structure consists of several associated variables which detail the location and nature of the error.

The possible error statuses are cross-referenced with the calls used for the two methods of accessing the computer interface (refer to Table 5-1). An explanation of each status follows.

ICI_ERROR_TEXT This routine will take an error structure as an argument and return the error text associated with any errors.

The actual header file describing the error structure is included in the appendices (***ici_err.h***).

Table 5-1. semAPI Return Status vs. Access Method

Access Method	semAPI Return Status			
	ICI_OK	ICI_WARNING	ICI_FATAL	ICI_CONTINUE
WAIT	X	X	X	
QUICK	X	X	X	X

The **RETIEVE QUICK** function may additionally return *ICI_CONTINUE* to indicate the response is not ready from the computer interface.

A response of *ICI_FATAL* or *ICI_WARNING* indicates that an error message number has been loaded into the error structure.

Error Returns

The following error returns can be received from any application call to the semAPI function library.

ICI_OK An error status of *ICI_OK* indicates one of two possibilities: On a wait access call, a message was built, sent and received from the computer interface, decoded, and found to contain no computer interface error in the message. On a quick access call, the message has been built and sent. An *ICI_OK* return status indicates that the error structure is empty. The user data area is filled with information from the computer interface module.

ICI_WARNING An error status of *ICI_WARNING* indicates the same events described for *ICI_OK* have transpired with indications of potential problems. The error structure is updated with diagnostic information (error numbers, etc.), which help to determine where the potential problem exists. The error structure may contain warning errors from multiple layers. The user can check ***s_error_layer*** in the error structure to determine where the error (or errors) occurred. The user data area is filled in with information from the computer interface module but the data is *suspect* because a warning occurred.

ICI_FATAL A return status of *ICI_FATAL* ultimately indicates that an action of the function call was not completed. The error structure must be examined to determine the layer (sub layer, message driver, server, computer interface) where the fatal error condition occurred. The error structure may contain errors at multiple layers although there will only be one error which caused the return status to be *ICI_FATAL*. The user can check the layer mask (***s_err_layer***) in the error structure to determine where the error (or errors) occurred. No data has been filled into the user data area.

ICI_CONTINUE A return status of *ICI_CONTINUE* is returned on a *RETRIEVE QUICK* call when the client has not received the semAPI message from the server. An *ICI_CONTINUE* return status indicates that the error structure is empty. No data has been filled into the user data area.

> **NOTE:** The user application must clear the error structure before each use.

Error Translation

The **ICI ERROR TEXT** function is used to process error messages. This function converts the errors returned in an error structure into an English error message. Refer to Section 6 for details.

SECTION 6 - FUNCTION LIBRARY

INTRODUCTION

This section contains all of the Application Programming Interface (semAPI) functions. The semAPI function library provides host platform access to a INFI 90 OPEN system.

The API function software is available in two user levels: data acquisition and supervisory control.

Data Acquisition (DA) Provides data acquisition and process monitoring capabilities.

Supervisory Control (SC) Provides data acquisition, process monitoring and supervisory capabilities.

This section identifies the user level of each function. Also, refer to Table 1-5 for a complete list of functions and associated control levels.

The functions are categorized into the following groups:

- ICI configuration.
- PCU.
- Read.
- Write.
- Time.
- Manager.
- Miscellaneous.

FUNCTION FORMAT

All functions require a header file. Each function identifies the appropriate header files. Functions in each category are alphabetized. Each function has a description, listing of applicable user levels and interface modules, listing of header files, format access, valid point types, return parameters, user parameters and user data area:

Description Explains the function purpose and any requirements that may exist.

Applicable Levels and Modules Identifies the user level of the function; whether it is Data Acquisition (DA) or Supervisory Control (SC). Also identified here is the applicable interface module. A shaded nomenclature or user level indicates that it is ***not applicable*** to the function.

Header File Lists header files that an application must include in order to use the function. The order of the header files shown for each

function is the order that must be used in the application program.

Format Shows the specific function entry (**bold**) and the parameters. Most functions list two entry methods: wait access and quick access. The read exceptions functions list a third method called internal access. The wait method issues the function to the computer interface and waits for a reply.

The quick method does not wait for a reply. A message number returns indicating that the message was sent. The message number is used with a second function (**RETRIEVE QUICK**) to receive the reply.

The internal method is used for reading exception reports. When an application issues any of the read exceptions functions, the message driver layer of the software returns exception reports to the user data area. If the number of exception reports returned from the computer interface exceeds the user data area capacity, a warning is returned to the application. When this occurs the remaining exception reports are written to internal buffers. The application can then issue the internal method functions to read the buffered exception reports.

A table in the format section lists parameter types, parameters, and description of parameters.

Point Type Lists the allowable point types for the function.

Returns Replies indicating the status of the issued function. A table in the returns section lists the return type, and description of the return code.

User Parameters Lists the input to the computer interface function. This is the data the user must supply in order to call the function.

User Data Area Lists the output from the computer interface function. This is the data that the computer interface passes back to the user.

ICI CONFIGURATION

The ICI configuration category contains functions related to computer interface initialization and set up. These functions are used to startup the computer interface and establish the database in the computer interface.

Callup

DESCRIPTION

Allows access to all computer interface functions when the correct password is entered. The number of bytes in a computer interface is defined by the #define PASSWORD_SIZE. The original password bytes are defined at the diagnostic terminal connected to the termination unit. The password enabled dipswitch and diagnostics dipswitch must be enabled. This security measure provides password protection when using modems for remote host computer to computer interface communications.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (workflag, point types, INFI 90 OPEN address)
l3icconf.h

FORMAT

wait access: **s_callup_w**
*(s_log_ici, *stp_error, *stp_callup_param, *stp_work_flag_data, l_data_size)*

quick access: **s_callup_q**
*(s_log_ici, *stp_error, *stp_callup_param, *stp_msg_id)*

Type	Parameter	Description
short	s_log_ici	Logical computer interface.
ERR_STRUCT	*stp_error	Pointer to an error structure.
CALLUP_PARAM	*stp_callup_param	Pointer to call up parameter structure.
WORK_FLAG	*stp_work_flag_data	Pointer to user data area.
long	l_data_size	Size of user data area.
MSG_ID	*stp_msg_id	Refer to Appendix K for quick message ID structure.

POINT TYPE

None. Enables access to the entire computer interface given the correct password. Does not deal with a particular point type.

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL Quick Access: ICI_OK ICI_FATAL ICI_WARNING

USER PARAMETERS

Parameter	Description
CALLUP_PARAM	User parameter area for **CALLUP**.
uc_password[ ]	The computer interface password. There can be a maximum of PASSWORD_SIZE bytes defined as a computer interface password. Define the password using the diagnostic terminal.

USER DATA AREA

Parameter	Description
WORK_FLAG	Refer to Appendix A for an explanation of the work flag user data area.

Connect Point Group

DESCRIPTION

Connects a group of established points from the computer interface to the host computer. Maximum group size is defined by MAX_CONNECT_POINT_GROUP. The minimum group size is defined by MIN_CONNECT_POINT_ GROUP.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (workflag, point types, INFI 90 OPEN address)
l3icconf.h

FORMAT

wait access: **s_connect_point_group_w**
*(s_log_ici, *stp_conn_pt_grp_param, *stp_error, *stp_work_flag_data, l_data_size)*

quick access: **s_connect_point_group_q**
*(s_log_ici, *stp_conn_pt_grp_param, *stp_error, *stp_msg_id)*

Type	Parameter	Description
short	s_log_ici	Logical computer interface.
CONN_PT_GRP_PARAM	*stp_conn_pt_grp_param	Pointer to connect point group parameter structure.
ERR_STRUCT	*stp_error	Pointer to an error structure.
WORK_FLAG	*stp_work_flag_data	Pointer to user data area.
long	l_data_size	Size of user data area.
MSG_ID	*stp_msg_id	Refer to Appendix K for quick message ID structure.

POINT TYPES

PT_ANALOG
PT_ASCII_STRING
PT_CONTROL_POINT
PT_DAANG
PT_DADIG
PT_DD
PT_DIGITAL
PT_ENHANCED_TREND
PT_EXTENDED_MODULE_STATUS
PT_MODULE_STATUS
PT_MSDD
PT_PROCESS VARIABLE
PT_RATIO_INDEX
PT_RCM
PT_REAL4_ANALOG_READ
PT_REM_MOTOR_CONTROL
PT_RMSC
PT_SET_POINT
PT_STATION
PT_STATION_STATUS
PT_TEXT_SELECTOR

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL Quick Access: ICI_OK ICI_FATAL ICI_WARNING

USER PARAMETERS

Parameter	Description
CONN_PT_GRP_PARAM	User parameter area.
c_num_points	The actual number of point indices in group. This is the number of indices in the s_indices array of point indices.
s_indices[ ]	The array of point indices to connect. A maximum of MAX_CONNECT_POINT_GROUP indices can be included in this array in any one call.
ICI_TAGNAME st_tagnames[ ]	A character array containing the tag name associated with a particular computer interface index. A maximum of MAX_CONNECT_POINT_GROUP tag names can be included in this array in any one call.

USER DATA AREA

Parameter	Description
WORK_FLAG	Refer to Appendix A for an explanation of the work flag user data area.

Connect Point List

DESCRIPTION

Connects a sequential list of established points from the computer interface to the host computer.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (workflag, point types, INFI 90 OPEN address)
l3icconf.h

FORMAT

wait access: **s_connect_point_list_w**
*(s_log_ici, *stp_conn_pt_lis_param, *stp_error, *stp_work_flag_data, l_data_size*)

quick access: **s_connect_point_list_q**
*(s_log_ici, *stp_conn_pt_lis_param, *stp_error, *stp_msg_id)*

Type	Parameter	Description
short	s_log_ici	Logical computer interface.
CONN_PT_LIS_PARAM	*stp_conn_pt_lis_param	Pointer to connect point list parameter structure.
ERR_STRUCT	*stp_error	Pointer to an error structure.
WORK_FLAG	*stp_work_flag_data	Pointer to user data area.
long	l_data_size	Size of user data area.
MSG_ID	*stp_msg_id	Refer to Appendix K for quick message ID structure.

POINT TYPES

PT_ANALOG
PT_ASCII_STRING
PT_CONTROL_POINT
PT_DAANG
PT_DADIG
PT_DD
PT_DIGITAL
PT_ENHANCED_TREND
PT_EXTENDED_MODULE_STATUS
PT_MODULE_STATUS
PT_MSDD
PT_PROCESS VARIABLE
PT_RATIO_INDEX
PT_RCM
PT_REAL4_ANALOG_READ
PT_REM_MOTOR_CONTROL
PT_RMSC
PT_SET_POINT
PT_STATION
PT_STATION_STATUS
PT_TEXT_SELECTOR

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL Quick Access: ICI_OK ICI_FATAL ICI_WARNING

USER PARAMETERS

Parameter	Description
CONN_PT_LIS_PARAM	User parameter area for the connect point list.
s_first_index	The first index to connect.
s_last_index	The last index to connect.
ICI_TAGNAME st_first_tagname[ ]	A character array containing the first tag name associated with a particular computer interface index.
ICI_TAGNAME st_last_tagname[ ]	A character array containing the last tag name associated with a particular computer interface index.

NOTE: The computer interface connects all points in between the s_first_index and the s_last_index inclusive. If a point in the list is not actually established, the computer interface will not return an error message on the connect call.

USER DATA AREA

Parameter	Description
WORK_FLAG	Refer to Appendix A for an explanation of the work flag user data area.

Connect to Logical Computer Interface

DESCRIPTION

This function calls the necessary functions to make a connection with a computer interface. Connections can be either as ICI_SHARED or ICI_EXCLUSIVE. Shared connection allows multiple users to connect to the computer interface. Exclusive does not allow any other users to connect until the ICI_EXCLUSIVE user disconnects. Attempting to connect with ICI_EXCLUSIVE while other users are connected will fail. The st_time_out member indicates to the server a maximum amount of inactivity time to be allowed. If that time expires without activity, the server will disconnect the client.

NOTE: This function issues an environment function to the computer interface. It is used to determine if communications to the interface are established, the type of loop (Plant Loop or INFI-NET), and the maximum index number for the interface.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_time.h (time structure)
l3mang.h2
l3qual.h

FORMAT

s_connect_to_ici
*(s_log_ici, *stp_connect_param, *stp_error*)

Type	Parameter	Description
short	s_log_ici	Logical computer interface.
ICI_CONNECT_PARAM	*stp_connect_param	Pointer to user parameter area.
ERR_STRUCT	*stp_error	Pointer to an error structure.

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL

USER PARAMETERS

Parameter	Description
ICI_CONNECT_PARAM	User parameter area for connect to logical ICI.
c_type_connection	Type of user connection being requested. Valid values include ICI_EXCLUSIVE or ICI_SHARED.
st_time_out	Contains link connection time-out value. Refer to Appendix H for more information on TIME_STRUCT.
c_return_tagnames	Tells the read functions to return tag indices and tag names. Performance decreases significantly if tag names are retrieved with each read reply. Valid values include: YES and NO. Set this to YES only when using and INOSM01 module. If set to YES while using all other communication modules it will cause other functions to pass back a device driver error (INVAL_MSG_TYPE) when trying to return tag names.

Disconnect from Logical Computer Interface

DESCRIPTION

Disconnects the host from the computer interface.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
l3mang.h

FORMAT

s_disconnect_from_ici
*(s_log_ici, *stp_error)*

Type	Parameter	Description
short	s_log_ici	Logical computer interface.
ERR_STRUCT	*stp_error	Pointer to an error structure.

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL

Disconnect Point Group

DESCRIPTION

Disconnects a group of established and connected points from the computer interface. The maximum group size is defined by MAX_CONNECT_POINT_GROUP. A minimum group size is defined by MIN_CONNECT_POINT_GROUP. The computer interface cannot disconnect module status points.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (workflag, point types, INFI 90 OPEN address)
l3icconf.h

FORMAT

wait access: **s_disconnect_point_group_w**
*(s_log_ici, *stp_disconn_pt_grp_param, *stp_error, *stp_work_flag_data, l_data_size)*

quick access: **s_disconnect_point_group_q**
*(s_log_ici, *stp_disconn_pt_grp_param, *stp_error, *stp_msg_id)*

Type	Parameter	Description
short	s_log_ici	Logical computer interface.
CONN_PT_GRP_PARAM	*stp_disconn_pt_grp_param	Pointer to disconnect point group parameter structure.
ERR_STRUCT	*stp_error	Pointer to an error structure.
WORK_FLAG	*stp_work_flag_data	Pointer to user data area.
long	l_data_size	Size of user data area.
MSG_ID	*stp_msg_id	Refer to Appendix K for quick message ID structure.

POINT TYPES

PT_ANALOG
PT_ASCII_STRING
PT_CONTROL_POINT
PT_DAANG
PT_DADIG
PT_DD
PT_DIGITAL
PT_ENHANCED_TREND
PT_EXTENDED_MODULE_STATUS
PT_MODULE_STATUS
PT_MSDD
PT_PROCESS VARIABLE
PT_RATIO_INDEX
PT_RCM
PT_REAL4_ANALOG_READ
PT_REM_MOTOR_CONTROL
PT_RMSC
PT_SET_POINT
PT_STATION
PT_STATION_STATUS
PT_TEXT_SELECTOR

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL Quick Access: ICI_OK ICI_FATAL ICI_WARNING

USER PARAMETERS

Parameter	Description
CONN_PT_GRP_PARAM	User parameter area.
c_num_points	The actual number of point indices in group. This is either the number of indices in the s_indices array or the number of tag names in the st_tagname array.
s_indices[ ]	The array of point indices to disconnect. **NOTE**: A maximum of MAX_CONNECT_POINT_GROUP and a minimum of MIN_CONNECT_POINT_ GROUP indices can be included in this array in any one function call.
ICI_TAGNAME st_tagnames[ ]	A character array containing the tag name associated with a particular computer interface index.

USER DATA AREA

Parameter	Description
WORK_FLAG	Refer to Appendix A for an explanation of the work flag user data area. A maximum of MAX_CONNECT_POINT_GROUP tag names can be included in this array in any one call.

Disconnect Point List

DESCRIPTION

Disconnects a sequential list of established points from the computer interface.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (workflag, point types, INFI 90 OPEN address)
l3icconf.h

FORMAT

wait access: **s_disconnect_point_list_w**
*(s_log_ici, *stp_disconn_pt_lis_param, *stp_error, *stp_work_flag_data, l_data_size)*

quick access: **s_disconnect_point_list_q**
*(s_log_ici, *stp_disconn_pt_lis_param, *stp_error, *stp_msg_id)*

Type	Parameter	Description
short	s_log_ici	Logical computer interface.
CONN_PT_LIS_PARAM	*stp_disconn_pt_lis_param	Pointer to disconnect point list parameter structure.
ERR_STRUCT	*stp_error	Pointer to an error structure.
WORK_FLAG	*stp_work_flag_data	Pointer to user data area.
long	l_data_size	Size of user data area.
MSG_ID	*stp_msg_id	Refer to Appendix K for quick message ID structure.

POINT TYPES

PT_ANALOG
PT_ASCII_STRING
PT_CONTROL_POINT
PT_DAANG
PT_DADIG
PT_DD
PT_DIGITAL
PT_ENHANCED_TREND
PT_EXTENDED_MODULE_STATUS
PT_MODULE_STATUS
PT_MSDD
PT_PROCESS VARIABLE
PT_RATIO_INDEX
PT_RCM
PT_REAL4_ANALOG_READ
PT_REM_MOTOR_CONTROL
PT_RMSC
PT_SET_POINT
PT_STATION
PT_STATION_STATUS
PT_TEXT_SELECTOR

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL Quick Access: ICI_OK ICI_FATAL ICI_WARNING

USER PARAMETERS

Parameter	Description
CONN_PT_LIS_PARAM[1]	User parameter area.
s_first_index	The first index to connect.
s_last_index	The last index to connect.
ICI_TAGNAME st_first_tagname	A character array containing the first tag name associated with a particular computer interface index.
ICI_TAGNAME st_last_tagname	A character array containing the last tag name-associated with a particular computer interface index.

NOTE:

1. The computer interface will disconnect all points in between the s_first_index and s_last_index inclusive. If a point in the list is not actually established the computer interface will not return an error message on the disconnect call.

USER DATA AREA

Parameter	Description
WORK_FLAG st_work_flag	Refer to Appendix A for an explanation of the work flag user data area.

Disestablish Point

DESCRIPTION

This function disestablishes all point types. For export points, this function does not delete exception report routes established to the point by other modules. Before establishing a previously used index number as a different type of point, all other modules in the system must disestablish ***all*** exception report routes from that point. The computer interface sends bad quality exception reports with unassigned point type to all other modules in the system that have established exception report routes.

Disestablishing an import point read by the host computer causes the computer interface to disestablish the associated exception report route.

Establishing a previously used index number as a different type of point may cause portions of the computer interface memory to become fragmented. Refrain from re-establishing points and keep the data base as static as possible in order to prevent this from happening.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (workflag, point types, INFI 90 OPEN address)
l3icconf.h

FORMAT

wait access: **s_disestablish_point_w**
*(s_log_ici, *stp_disestab_pt_param, *stp_error, *stp_work_flag_data, l_data_size*)

quick access: **s_disestablish_point_q**
*(s_log_ici, *stp_disestab_pt_param, *stp_error, *stp_msg_id)*

Type	Parameter	Description
short	s_log_ici	Logical computer interface.
DISESTAB_PT_PARAM	*stp_disestab_pt_param	Pointer to disestablish point parameter.
ERR_STRUCT	*stp_error	Pointer to an error structure.
WORK_FLAG	*stp_work_flag_data	Pointer to user data area.
long	l_data_size	Size of user data area.
MSG_ID	*stp_msg_id	Refer to Appendix K for quick message ID structure.

POINT TYPES All.

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL Quick Access: ICI_OK ICI_FATAL ICI_WARNING

USER PARAMETERS

Parameter	Description
DISESTAB_PT_PARAM	User parameter.
s_pt_index	The point index of the point in the computer interface to be disestablished.

USER DATA AREA

Parameter	Description
WORK_FLAG	Refer to Appendix A for an explanation of the work flag user data area.

Environment

DESCRIPTION

Returns the computer interface environmental data to the host computer. Incudes the computer interface module type and status (on or off-line), operating mode, firmware level, and acknowledgment of a **RESTART**. Computer interface operating modes are backwardly compatible. Issue this function before a **RESTART** if desired.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (workflag, point types, INFI 90 OPEN address)
l3qual.h (yes and no)
l3icconf.h

FORMAT

wait access: **s_environment_w**
*(s_log_ici, *stp_error, *stp_env_data, l_data_size)*

quick access: **s_environment_q**
*(s_log_ici, *stp_error, *stp_msg_id)*

Type	Parameter	Description
short	s_log_ici	Logical computer interface.
ERR_STRUCT	*stp_error	Pointer to an error structure.
ENV_DATA	*stp_env_data	Pointer to user data area.
long	l_data_size	Size of user data area.
MSG_ID	*stp_msg_id	Refer to Appendix K for quick message ID structure.

POINT TYPE

None. This function does not refer to any point type in particular. It refers directly to the interface module. It causes the computer interface to pass back data internal to the computer interface module.

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL Quick Access: ICI_OK ICI_FATAL ICI_WARNING

USER PARAMETERS

None. No user parameters other than the logical computer interface (s_log_ici).

USER DATA AREA

Parameter	Description
ENV_DATA	User data area.
WORK_FLAG st_work_flag	Refer to Appendix A for an explanation of the work flag user data area.
c_type	The type of the computer interface module. Valid type is: TYPE_INICI03_IIMCP02.
c_mode	The mode of the computer interface module. Valid modes are: MODE_INPCI02, MODE_INICI03_IIMCP02. **NOTE**: The mode of the computer interface may be different than the computer interface type. A INICI03,IIMCP02 can be restarted in a INPCI02 (plant loop) mode.
c_rev_letter	The firmware revision letter of the computer interface module.
c_rev_number	The firmware revision number of the computer interface module. **NOTE**: The revision letter and number make up the firmware revision of the computer interface module. This would be a revision like C.1, where C is the revision letter and 1 is the revision number.
c_restarted	Indicates whether the computer interface has been restarted.
NO	The module has not been restarted.
YES	The module has been restarted.
c_online	Indicates whether the computer interface is on-line or off-line.
NO	The module is off-line.
YES	The module is on-line.
uc_node	The node (PCU) number of the computer interface.
uc_loop	The loop number of the computer interface.

NOTE: The loop and node number of the computer interface defines the location on the INFI 90 OPEN network.

Establish Export Point

DESCRIPTION

Establishes an export point (report point) in the computer interface point table and exports the specifications of the indicated point.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (workflag, point types, INFI 90 OPEN address)
l3icconf.h

FORMAT

wait access: **s_establish_export_point_w**
*(s_log_ici, *stp_estab_export_param, *stp_error, *stp_work_flag_data, l_data_size)*

quick access: **s_establish_export_point_q**
*(s_log_ici, *stp_estab_export_param, *stp_error, *stp_msg_id)*

Type	Parameter	Description
short	s_log_ici	Logical computer interface.
ESTAB_EXPORT_PARAM	*stp_estab_export_param	Pointer to establish point parameter structure.
ERR_STRUCT	*stp_error	Pointer to an error structure.
WORK_FLAG	*stp_work_flag_data	Pointer to user data area.
long	l_data_size	Size of user data area.
MSG_ID	*stp_msg_id	Refer to Appendix K for quick message ID structure.

POINT TYPES

PT_ANALOG_REPORT	PT_REAL4_ANALOG_REPORT
PT_DIGITAL_REPORT	PT_RMSC_REPORT
PT_RCM_REPORT	PT_STATION_REPORT

NOTE: Only the following point types are valid for Data Acquisition (DA) users: PT_ANALOG_REPORT, and PT_DIGITAL_REPORT.

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL Quick Access: ICI_OK ICI_FATAL ICI_WARNING

USER PARAMETERS

Parameter	Description
ESTAB_EXPORT_PARAM	User parameter area.
s_pt_index	The point index in the computer interface to use for the point being established.
c_pt_type	The point type being established. Refer to Appendix B for an explanation of the point types. Refer to ***POINT TYPE*** for the valid point types of this function.
union un_rpt report	A union of all of the possible report types. Only one of the following structures must be filled in based on the point type.

report: The member ***report*** is a union of the following structures:

Parameter	Description
ANA_REP analog	(Point type = PT_ANALOG_REPORT or PT_REAL4_ANALOG_REPORT) Analog report point structure.
s_eng_units	Engineering units.
f_zero	Analog zero.
f_span	Analog span.
f_high_alarm	Analog high alarm limit.
f_low_alarm	Analog low alarm limit.
DIG_REP digital	(Point type = PT_DIGITAL_REPORT) Digital report point structure.
c_alarm_spec	Digital report alarm specification.
LOGIC_0_ALARM	A digital state of zero is the alarm state.
LOGIC_1_ALARM	A digital state of one is the alarm state.
NO_ALARM_STATE	No digital alarm state is defined.
RCM_REP rcm	(Point type = PT_RCM_REPORT) RCM report point structure.
c_feedback	Feedback of RCM.
c_display_output	Display output of RCM.
RMSC_REP rmsc	(Point type = PT_RMSC_REPORT) RMSC report point structure.
s_eng_units	Engineering units.
f_zero	RMSC zero.
f_span	RMSC span.

Parameter	Description
f_high_limit	RMSC high alarm limit.
f_low_limit	RMSC low alarm limit.
STAT_REP station	(Point type = PT_STATION_REPORT) Station report point structure.
s_eng_units	Engineering units.
f_high_alarm	Station high alarm limit.
f_low_alarm	Station low alarm limit.
f_dev_alarm	Station deviation alarm.
f_pv_sp_span	Span of the Set Point (SP) and the Process Variable (PV).
f_pv_zero	Zero of the Process Variable (PV).
f_sp_zero	Zero of the Set Point (SP).
c_station_type	The type of the station.
BASIC_WITH_SETPOINT	Basic station with a set point variable.
RATIO_INDEX	Ratio index station.
CASCADE	Cascade type station.
BASIC_WITHOUT_SETPOINT	Basic station without the set point variable.
BASIC_WITH_BIAS	Basic station with BIAS.

USER DATA AREA

Parameter	Description
WORK_FLAG	Refer to Appendix A for an explanation of the work flag user data area.

Establish Import Point

DESCRIPTION

Establishes an import in the computer interface. Points must refer to unique pieces of data. Do not establish more than one point to the same data except for station variable point types.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Inteface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (workflag, point types, INFI 90 OPEN address)
l3icconf.h

FORMAT

wait access: **s_establish_import_point_w**
*(s_log_ici, *stp_estab_import_param, *stp_error*
**stp_work_flag_data, l_data_size)*

quick access: **s_establish_import_point_q**
*(s_log_ici, *stp_estab_import_param, *stp_error, *stp_msg_id)*

Type	Parameter	Description
short	s_log_ici	Logical computer interface.
ESTAB_IMPORT_PARAM	*stp_estab_import_param	Pointer to establish import point parameter structure.
ERR_STRUCT	*stp_error	Pointer to an error structure.
WORK_FLAG	*stp_work_flag_data	Pointer to user data area.
long	l_data_size	Size of user data area.
MSG_ID	*stp_msg_id	Refer to Appendix K for quick message ID structure.

POINT TYPES

PT_ANALOG
PT_ASCII_STRING[1]
PT_CONTROL_OUTPUT
PT_CONTROL_POINT
PT_DAANG
PT_DADIG
PT_DD
PT_DIGITAL
PT_ENHANCED_TREND
PT_EXTENDED_MODULE_STATUS
PT_MODULE_STATUS
PT_MSDD
PT_PROCESS VARIABLE
PT_RATIO_INDEX
PT_RCM
PT_REAL4_ANALOG_READ
PT_REM_MOTOR_CONTROL
PT_RMSC
PT_SET_POINT
PT_STATION
PT_STATION_STATUS
PT_TEXT_SELECTOR

NOTE: 1. PT_ASCII_STRING point type is only supported when using the INICI03 module.

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL Quick Access: ICI_OK ICI_FATAL ICI_WARNING

USER PARAMETERS

Parameter	Description
ESTAB_IMPORT_PARAM	User parameter area.
s_pt_index	The point index in the computer interface to use for the point being established.
c_pt_type	The point type being established. Refer to Appendix B for an explanation of the point types. Refer to ***POINT TYPE*** for this function.
c_connect_pt	Indicates whether the point is to be connected or not connected. Field is unused if point type is PT_ASCII_STRING. For point types PT_MODULE_STATUS, PT_EXTENDED_MODULE_STATUS, PT_SET_POINT_OUTPUT, PT_CONTROL_OUTPUT, PT_RATIO_INDEX_WRITTEN, and PLT_STATION_MODE use NO_CONNECT_PT.
CONNECT_PT	Connects point after establishing.
NO_CONNECT_PT	Does not connect point after establishing.
c_auto_disconnect	Indicates whether the point is to be automatically disconnected after receiving an exception report for the point. Field is only used if point type is PT_ASCII_STRING or the c_connect field is set to CONNECT_PT.
POINT_DISCONNECT_AFTER_XR	The point will be disconnected after the first exception report is received for the point.
POINT_NOT_DISCONNECTED	The point will remain connected until the point is either manually disconnected via a **DISCONNECT POINT LIST/GROUP** or disestablished from the computer interface.
INFI90ADR	INFI 90 OPEN address structure.
*stp_infi90_address	The INFI 90 OPEN address of the point.
s_loop	The loop number that contains the function code block.
s_node	The node (PCU) number that contains the function code block.
s_module	The module number that contains the function code block.
s_block	The block number that contains the function code block.

Parameter	Description
c_num_status_bytes	The number of status bytes to be returned in the exception report. Maximum status points defined by: MAX_STATUS_BYTES. Minimum status points defined by: MIN_STATUS_BYTES. Field used with PT_ASCII_STRING only.
c_num_data_bytes	The number of data bytes to be returned in the exception report: Maximum data bytes is defined by: MAX_DATA_BYTES. Minimum data bytes is defined by: MIN_DATA_BYTES. Field used with PT_ASCII_STRING only.
c_control_requested	Indicates whether control of the ASCII string is required. DISABLE defines control not requested. ENABLE defines control requested. Field used with PT_ASCII_STRING only.
c_control_buff_size	The number of bytes in the control buffer: Maximum number of bytes is defined by: MAX_CONTROL_BUFFER. Minimum number of bytes is defined by: MIN_CONTROL_BUFFER. Field used with PT_ASCII_STRING only.

USER DATA AREA

Parameter	Description
WORK_FLAG	Refer to Appendix A for an explanation of the work flag user data area.

Hangup

DESCRIPTION

Disables access to all computer interface functions except **CALLUP** and **ENVIRONMENT** when password protection is enabled. Issue this function for each port originally enabled with **CALLUP**.

The host computer uses this function to give up access to the communication channel when password protection is enabled. Using password protection provides some measure of protection when using modems for remote host computer to computer interface communications. This function will fail if password protection is not enabled.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (workflag, point types, INFI 90 OPEN address)
l3icconf.h

FORMAT

wait access: **s_hangup_w**
*(s_log_ici, *stp_error, *stp_work_flag_data, l_data_size)*

quick access: **s_hangup_q**
*(s_log_ici, *stp_error, *stp_msg_id)*

Type	Parameter	Description
short	s_log_ici	Logical computer interface.
ERR_STRUCT	*stp_error	Pointer to an error structure.
WORK_FLAG	*stp_work_flag_data	Pointer to user data area.
long	l_data_size	Size of user data area.
MSG_ID	*stp_msg_id	Refer to Appendix K for quick message ID structure.

POINT TYPE

None. Disables access to the entire computer interface. Does not deal with a particular point type.

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL Quick Access: ICI_OK ICI_FATAL ICI_WARNING

USER PARAMETERS

None. There are no user parameters to this function other than the logical computer interface (s_log_ici).

USER DATA AREA

Parameter	Description
WORK_FLAG	Refer to Appendix A for an explanation of the work flag user data area.

On-Line/Off-Line

DESCRIPTION

Sets the computer interface mode to on-line or off-line without restarting the computer interface. When changing to on-line mode the computer interface establishes and connects points that were configured when it was off-line. When on-line, the computer interface accepts exception reports. When off-line, the computer interface does not accept exception reports and appears off-line to other nodes.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (workflag, point types, INFI 90 OPEN address)
l3icconf.h

FORMAT

wait access: **s_ici_online_offline_w**
*(s_log_ici, *stp_on_off_param, *stp_error, *stp_work_flag_data, l_data_size)*

quick access: **s_ici_online_offline_q**
*(s_log_ici, *stp_on_off_param,*stp_error, *stp_msg_id)*

Type	Parameter	Description
short	s_log_ici	Logical computer interface.
ICI_ON_OFF_PARAM	*stp_on_off_param	Pointer to computer interface on/off-line parameter structure.
ERR_STRUCT	*stp_error	Pointer to an error structure.
WORK_FLAG	*stp_work_flag_data	Pointer to user data area.
long	l_data_size	Size of user data area.
MSG_ID	*stp_msg_id	Refer to Appendix K for quick message ID structure.

POINT TYPE

None. Does not refer to any point type in particular. It refers directly to the computer interface. It causes the computer interface hardware to modify its current state on the loop.

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL Quick Access: ICI_OK ICI_FATAL ICI_WARNING

USER PARAMETERS

Parameter	Description
ICI_ON_OFF_PARAM	User parameter area.
c_mode	The mode to put the computer interface into.
ICI_ONLINE	Make the computer interface energize onto the INFI 90 OPEN loop. Make the computer interface operate in primary mode.
ICI_OFFLINE	De-energize the computer interface from the INFI 90 OPEN loop. Make the computer interface operate in secondary (backup) mode.

USER DATA AREA

Parameter	Description
WORK_FLAG	Refer to Appendix A for an explanation of the work flag user data area.

Regenerate Specs

DESCRIPTION

Causes a module to send point specification information to the computer interface. Regenerating specs with an index of zero causes specifications to be regenerated for established points in the computer interface.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (workflag, point types, INFI 90 OPEN address)
l3icconf.h

FORMAT

wait access: **s_regenerate_specs_w**
*(s_log_ici, *stp_regen_specs_param, *stp_error, *stp_work_flag_data, l_data_size)*

quick access: **s_regenerate_specs_q**
*(s_log_ici, *stp_regen_specs_param, *stp_error, *stp_msg_id)*

Type	Parameter	Description
short	s_log_ici	Logical computer interface.
REGEN_SPECS_PARAM	*stp_regen_specs_param	Pointer to regenerate specs parameter structure.
ERR_STRUCT	*stp_error	Pointer to an error structure.
WORK_FLAG	*stp_work_flag_data	Pointer to user data area.
long	l_data_size	Size of user data area.
MSG_ID	*stp_msg_id	Refer to Appendix K for quick message ID structure.

POINT TYPE

None. Does not refer to any point type in particular. It refers directly to the computer interface. It causes the computer interface to reread the specification from the INFI 90 OPEN system for the given indices.

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL Quick Access: ICI_OK ICI_FATAL ICI_WARNING

USER PARAMETERS

Parameter	Description
REGEN_SPECS_PARAM	User parameter area.
s_index	The index in the computer interface to regenerate specs for.
ICI_TAGNAME st_tagname	A character array containing the tag name associated with a particular computer interface index.

NOTE: The special index of 0 (zero) is reserved to indicate that the computer interface should regenerate specs for all points currently established in the computer interface.

USER DATA AREA

Parameter	Description
WORK_FLAG	Refer to Appendix A for an explanation of the work flag user data area.

Restart

DESCRIPTION

Clears the computer interface point table and gives the computer interface executive control parameters. After a hardware reset, this should be the first function issued unless password protection is enabled. If so, issue **CALLUP** first then **RESTART**. The environment function may be issued before the **RESTART**.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (workflag, point types, INFI 90 OPEN address)
l3icconf.h

FORMAT

wait access: **s_ici_restart_w**
*(s_log_ici, *stp_error, *stp_restart_param, *stp_restart_data, l_data_size)*

quick access: **s_ici_restart_q**
*(s_log_ici, *stp_error, *stp_restart_param, *stp_msg_id)*

Type	Parameter	Description
short	s_log_ici	Logical computer interface.
ERR_STRUCT	*stp_error	Pointer to an error structure.
RESTART_PARAM	*stp_restart_param	Pointer to restart parameter structure.
RESTART_DATA	*stp_restart_data	Pointer to user data area.
long	l_data_size	Size of user data area.
MSG_ID	*stp_msg_id	Refer to Appendix K for quick message ID structure.

POINT TYPE

None. Does not refer to any point type in particular. It refers directly to the computer interface. It causes the computer interface to restart and clear the internal point table.

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL Quick Access: ICI_OK ICI_FATAL ICI_WARNING

USER PARAMETERS

Parameter	Description
RESTART_PARAM	User parameter area. The following parameters pertain to both INFI-NET and Plant Loop.
c_watchdog_timer	The value used in the computer interface as a watchdog timer. The value of the user supplied timer value is multiplied by 2.5 and used for the watchdog timer value in seconds. A value of 0 indicates no watchdog timer.
c_reply_delay	The reply delay value used by the computer interface. The value of the user supplied delay is multiplied by 0.01 and is used for the delay in seconds. This is the amount of time the computer interface will wait before sending the reply to the host.
c_time_synch	Tells the computer interface whether it can time synch the INFI 90 OPEN loop.
DISABLE	Computer interface cannot time synch INFI 90 OPEN.
ENABLE	Computer interface can time synch INFI 90 OPEN.
c_excpt_rpt_screen	Tells the computer interface whether to screen exception reports based on maximum reporting times expiring.
DISABLE	Does not screen exception reports for any reason.
ENABLE	Screens exception reports when the maximum reporting time expires.
c_station_control	Tells the computer interface whether station control will be permitted from the computer interface.
DISABLE	Station control is not allowed from this computer interface.
ENABLE	Station control is allowed from this computer interface.
c_primary	Tells the computer interface what mode to start-up in.
DISABLE	Start-up in the off-line mode.
ENABLE	Start-up in the on-line mode.
c_infinet_mode[1]	Tells the computer interface to start in INFI-NET mode or Plant Loop mode.
DISABLE	Start the computer interface in Plant Loop mode.
ENABLE	Start the computer interface in INFI-NET mode.

NOTE:
1. If c_infinet_mode is ENABLED, only the INFI-NET parameters are valid. If DISABLED, only the Plant Loop parameters are valid.

Plant Loop parameters: The following parameters are required if c_infinet_mode is DISABLE:

Parameter	Description
c_xon_xoff	Tells the computer interface to use the XON-XOFF communications protocol.
DISABLE	Disable using XON-XOFF.
ENABLE	Use the XON-XOFF protocol.
c_enhanced_mode	Tells the computer interface to restart in enhanced mode.
DISABLE	Do not start the computer interface in enhanced mode.
ENABLE	Start the computer interface in enhanced mode.
c_separate_commands	Tells the computer interface to separate RCM command exception reports from normal exception reports when returned to the user.
DISABLE	Do not separate commands.
ENABLE	Separate commands.

INFI-NET parameters: The following parameters are required if c_infinet_mode is ENABLE:

Parameter	Description
c_work_flag	Tells the computer interface whether to return the work flag with each reply.
DISABLE	Do not return the work flag with each function.
ENABLE	Return the work flag with each function.
c_bad_qual_alm	Tells the computer interface whether to use bad quality alarm management.
DISABLE	Do not use bad quality alarm management.
ENABLE	Use bad quality alarm management.
c_time_stamp	Tells the computer interface whether to return time stamps along with values in various functions.
DISABLE	Do not return time stamps along with values and statuses.
ENABLE	Return time stamps along with values and statuses.
c_wall_clock	Tells the computer interface whether to add the wall clock offset (local time) to the time stamps being returned.
DISABLE	Do not add the wall clock offset to time stamps.
ENABLE	Add the wall clock offset to time stamps at the computer interface.

USER DATA AREA

Parameter	Description
RESTART_DATA	User data area.
WORK_FLAG st_work_flag	Refer to Appendix A for an explanation of the work flag user data area.
uc_ici_node_num	The node number of the computer interface.
uc_ici_loop_num	The loop number of the computer interface.

PCU CONFIGURATION FUNCTIONS

This category of functions relates to the ICI communication module directly communicating to the INFI 90 OPEN process control unit. If not used correctly, these functions can cause severe degradation of system performance.

Demand Module Status

DESCRIPTION

Reads the status of any module in the system, even if there is no established point as the module status. The reply message is delayed until the module responds to the computer interface.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (work flag, INFI 90 OPEN address, point types)
l3btable.h (defines status table returned in DEMAND_MOD_STATUS_DATA)
l3pcconf.h

FORMAT

wait access: **s_demand_module_status_w**
*(s_log_ici, *stp_infi90adr_param, *stp_error, *stp_demand_mod_status_data, l_data_size)*

quick access: **s_demand_module_status_q**
*(s_log_ici, *stp_infi90adr_param, *stp_error, *stp_msg_id)*

Type	Parameter	Description
short	s_log_ici	Logical computer interface.
INFI90ADR	*stp_infi90adr_param	Pointer to demand module status parameter structure.
ERR_STRUCT	*stp_error	Pointer to an error structure.
DEMAND_MOD_STATUS_DATA	*stp_demand_mod_status_data	Pointer to user data area.
long	l_data_size	Size of user data area.
MSG_ID	*stp_msg_id	Refer to Appendix K for quick message ID structure.

POINT TYPE

None. This function does not refer to any point type in particular. It refers directly to the INFI 90 OPEN system (by address, loop, PCU, module, and block).

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL Quick Access: ICI_OK ICI_FATAL ICI_WARNING

USER PARAMETERS

Parameter	Description
INFI90ADR	INFI 90 OPEN address structure.
*stp_infi90_address	The INFI 90 OPEN address of the point.
s_loop	The loop number of the module to demand the status from.
s_node	The node (PCU) number of the module to demand the status from.
s_module	The module to demand the status from.
s_block	The block number field is not used in this function.

USER DATA AREA

Parameter	Description
DEMAND_MOD_STATUS_DATA	User data area.
WORK_FLAG st_work_flag	Refer to Appendix A for an explanation of the work flag user data area.
STATUS_TABLE st_status	The status_tables structure is explained in Appendix E.

Read Block

DESCRIPTION

Reads block configuration data of a configured block. The block configuration can be read only when the module is in configure or execute mode. This function does not use the computer interface point table.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (work flag, INFI 90 OPEN address, point types)
l3pcconf.h

FORMAT

wait access: **s_read_block_w**
*(s_log_ici, *stp_infi90adr_param, *stp_error, *stp_read_block_data, l_data_size)*

quick access: **s_read_block_q**
*(s_log_ici, *stp_infi90adr_param, *stp_error, *stp_msg_id)*

Type	Parameter	Description
short	s_log_ici	Logical computer interface.
INFI90ADR	*stp_infi90adr_param	Pointer to read block parameter structure.
ERR_STRUCT	*stp_error	Pointer to an error structure.
READ_BLOCK_DATA	*stp_read_block_data	Pointer to user data area.
long	l_data_size	Size of user data area.
MSG_ID	*stp_msg_id	Refer to Appendix K for quick message ID structure.

POINT TYPE

None. Does not refer to any point type in particular. It refers directly to the INFI 90 OPEN system (by address, loop, PCU, module, and block).

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL Quick Access: ICI_OK ICI_FATAL ICI_WARNING

USER PARAMETERS

Parameter	Description
INFI90ADR	INFI 90 OPEN address structure.
*stp_infi90_address	The INFI 90 OPEN address of the point.
s_loop	The loop number of the module to demand the status from.
s_node	The node (PCU) number of the module to demand the status from.
s_module	The module to demand the status from.
s_block	The block number field is not used in this function.

USER DATA AREA

Parameter	Description
READ_BLOCK_DATA	User data area.
WORK_FLAG st_work_flag	Refer to Appendix A for an explanation of the work flag user data area.
c_block_data[ ]	The actual bytes in the block data returned from the computer interface. A maximum of MAX_BLOCK_DATA_SIZE bytes can be returned. A minimum of MIN_BLOCK_DATA_ SIZE bytes can be returned.

Tune Block

DESCRIPTION

Changes the tunable parameters of the function block. Module must be in the execute mode. The module must be in execute mode for parameters to be changed.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (work flag, INFI 90 OPEN address, point types)
l3pcconf.h

FORMAT

wait access: **s_tune_block_w**
*(s_log_ici, *stp_tune_block_param, *stp_error, *stp_work_flag_data, l_data_size)*

quick access: **s_tune_block_q**
*(s_log_ici, *stp_tune_block_param, *stp_error, *stp_msg_id*)

Type	Parameter	Description
short	s_log_ici	Logical computer interface.
TUNE_BLOCK_PARAM	*stp_tune_block_param	Pointer to tune block parameter structure.
ERR_STRUCT	*stp_error	Pointer to an error structure.
WORK_FLAG	*stp_work_flag_data	Pointer to user data area.
long	l_data_size	Size of user data area.
MSG_ID	*stp_msg_id	Refer to Appendix K for quick message ID structure.

POINT TYPE

None. Does not refer to any point type in particular. It refers directly to the INFI 90 OPEN system (by address, loop, PCU, module, and block).

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL Quick Access: ICI_OK ICI_FATAL ICI_WARNING

USER PARAMETERS

Parameter	Description
INFI90ADR	INFI 90 OPEN address structure.
*stp_infi90_address	INFI 90 OPEN address of point.
s_loop	The loop number of the block to tune.
s_node	The node (PCU) number of the block to tune.
s_module	The module number of the block to tune.
s_block	The block number to tune.
s_function_code	The function code of the block to tune.
c_buffer[ ]	The array containing the block parameters structure. A maximum of MAX_BLOCK_ DATA_SIZE bytes can be contained in the c_buffer array. Refer to Appendix F for an explanation of the block parameters structure.
c_buffer_len	Length of c_buffer[ ].

USER DATA AREA

Parameter	Description
WORK_FLAG	Refer to Appendix A for an explanation of the work flag user data area.

READ FUNCTIONS

This section describes the functions that read data from the computer interface to the host computer.

Read Command Exceptions

DESCRIPTION

Reads commands received by station report and remote manual set constant report points. Also reads process control commands received by remote control memory report points if the commands separation option is enabled in **RESTART**.

Use **READ WORK FLAG** to determine if exceptions have been received.

If the computer interface has received more than one command for a particular station variable since issuing **READ COMMAND EXCEPTIONS**, the reply contains the latest value. The host computer receives no indication of previous values.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (work flag, INFI 90 OPEN address, point types)
l3read.h

FORMAT

wait access: **s_read_command_except_w**
*(s_log_ici, *stp_error, *stp_rd_command_except_data, l_data_size, s_max_returned)*

quick access: **s_read_command_except_q**
*(s_log_ici, *stp_error,*stp_msg_id, s_max_returned)*

internal access: **s_get_command_except**
*(s_log_ici, *stp_error, *stp_rd_command_except_data, l_data_size)*

Type	Parameter	Description
short	s_log_ici	Logical computer interface.
ERR_STRUCT	*stp_error	Pointer to an error structure.
RD_COMMAND_EXCEPT_DATA	*stp_rd_command_except_data	Pointer to user data area.
long	l_data_size	Size of user data area.
short	s_max_returned	Maximum number of reports to be returned with this function.
MSG_ID	*stp_msg_id	Refer to Appendix K for quick message ID structure.

POINT TYPES

PT_RCM_REPORT
PT_RMSC_REPORT
PT_STATION_REPORT

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL Quick Access: ICI_OK ICI_WARNING ICI_FATAL

USER PARAMETERS

Parameter	Description
s_max_returned	The maximum number of reports to be returned with this function. A value of zero (0) causes the computer interface to return as many command exceptions as possible. A maximum value for this parameter is MAX_COMMANDS_INFI90 for INFI-NET ICIs and MAX_COMMANDS_PLANT for plant loop ICIs.

USER DATA AREA

Parameter	Description
RD_COMMAND_EXCEPT_DATA	User data area.
WORK_FLAG st_work_flag	Refer to Appendix A for an explanation of the work flag user data area.
s_count	The number of RD_COMMAND_ EXCEPT_ELEMENT structures that follow. For each function exception that is returned a RD_COMMAND_ EXCEPT_ELEMENT structure will be filled in. Maximum number of exceptions defined by MAX_COMMANDS_INFI90 for INFI-NET ICIs and MAX_COMMANDS_ PLANT for plant loop ICIs.
RD_COMMAND_EXCEPT_ELEMENT *stp_elements	A pointer to an array of command exception elements. The user application must allocate an array of RD_COMMAND_EXCEPT_ ELEMENT. Then the application must point *stp_elements to this allocated array.

Parameter	Description
RD_COMMAND_EXCEPT_ELEMENT	A pointer to an array of command exception elements.
s_index	The computer interface index that this command exception refers to.
ICI_TAGNAME st_tagname	A character array containing the tag name associated with a particular computer interface index.
uc_command	The type of command exception that has been returned. The type of command exception (uc_command) will indicate which member of the un_cmd union to access for valid data.
RD_CMD_EX_STATION_STATE	Command exception report for a station state also known as a station status or station mode. Access data in the st_station_state structure.
RD_CMD_EX_SET_POINT	Command exception report for a station set point. Access data in the st_set_point structure.
RD_CMD_EX_RATIO_INDEX	Command exception report for a station ratio index. Access data in the st_ratio_index structure.
RD_CMD_EX_CONTROL_OUTPUT	Command exception report for a station control output. Access data in the st_control_output structure.
RD_CMD_EX_RCM	Command exception report for an RCM. Access data in the st_rcm structure.
RD_CMD_EX_RMSC	Command exception report for an RMSC. Access data in the st_rmsc structure.
un_cmd	A union of sturctures for various types of expectations. The uc_command member indicates which structure is filled in.
st_station_state	
uc_station_mode	The mode of the station
GOTO_LOCAL_MANUAL	The station is in the local-manual (console/station-manual) mode.
GOTO_LOCAL_AUTO	The station is in the local-auto (console/station-auto) mode.
GOTO_LOCAL_CASCADE	The station is in the local cascade/ratio (console/station-cascade/ratio) mode.
GOTO_COMPUTER_MANUAL	The station is in the computer-manual mode.
GOTO_COMPUTER_CASCADE	The station is in the computer-auto mode.
GOTO_COMPUTER_AUTO	The station is in the computer cascade/ratio mode.
GOTO_LOCAL_LEVEL	The station is at the local level (cascade/station level).
GOTO_COMPUTER_LEVEL	The station is at the computer level.
GOTO_COMPUTER_BACKUP	The station is in the computer back-up state.
COMPUTER_OK	The station has the Computer OK signal from the host computer.
GOTO_PREVIOUS_STATE	The station has resumed the previous mode requested.
st_set_point	The set point structure.
uc_source	The source level. Valid sources include: LOCAL_SOURCE, COMPUTER_SOURCE.
f_value	The floating point value of the set point.

Parameter	Description
st_ratio_index	The ratio index structure.
uc_source	The source level. Valid sources include: LOCAL_SOURCE, COMPUTER_SOURCE.
f_value	The floating point value of the ratio index.
st_control_output	The control output structure.
uc_source	The source level. Valid sources include: LOCAL_SOURCE, COMPUTER_SOURCE.
f_value	The floating point value of the ratio index.
st_rcm	The RCM structure.
uc_mode	The mode of the RCM.
SUSTAIN_RESET	The RCM has a mode that indicates a sustained reset signal.
SUSTAIN_SET	The RCM has a mode that indicates a sustained set signal.
PULSE_RESET	The RCM has a mode that indicates a pulsed reset signal.
PULSE_SET	The RCM has a mode that indicates a pulsed set signal.
st_rmsc	The RMSC structure.
f_value	The floating point value of the RMSC point.

Read Data Exceptions

DESCRIPTION

Reads the current status or value of all import points that have been received by the computer interface. The order in which the reports are returned is not necessarily the same order as they occurred. Use the time stamp to verify the exact sequence of events. Point index zero (the interface module status) is also read with this command.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (work flag, INFI 90 OPEN address, point types)
l3read.h

FORMAT

wait access: **s_read_data_except_w**
*(s_log_ici, *stp_error, *stp_rd_data_except_data, l_data_size, s_max_returned)*

quick access: **s_read_data_except_q**
*(s_log_ici, *stp_error, *stp_msg_id, s_max_returned)*

internal access: **s_get_data_except**
*(s_log_ici, *stp_error, *stp_rd_data_except_data, l_data_size)*

Type	Parameter	Description
short	s_log_ici	Logical computer interface.
ERR_STRUCT	*stp_error	Pointer to an error structure.
RD_DATA_EXCEPT_DATA	*stp_rd_data_except_data	Pointer to user data area
long	l_data_size	Size of the user data area.
short	s_max_returned	Maximum number of reports to be returned with this function.
MSG_ID	*stp_msg_id	Refer to Appendix K for quick message ID structure.

POINT TYPES

PT_ANALOG	PT_PROCESS _VARIABLE
PT_ASCII_STRING	PT_RATIO_INDEX
PT_CONTROL_POINT	PT_RCM
PT_DAANG	PT_REAL4_ANALOG_READ
PT DADIG	PT_REM_MOTOR_CONTROL
PT_DD	PT_RMSC
PT_DIGITAL	PT_SET_POINT
PT_EXTENDED_MODULE_STATUS	PT_STATION
PT_MODULE_STATUS	PT_STATION_STATUS
PT_MSDD	PT_TEXT_SELECTOR

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL Quick Access: ICI_OK ICI_FATAL ICI_WARNING

USER PARAMETERS

Parameter	Description
s_max_returned	The maximum number of reports to be returned with this function. A value of zero (0) will cause the computer interface to return as many data exceptions as possible. A maximum value for this parameter is MAX_DATA_EXCEPTIONS.

USER DATA AREA

Parameter	Description
RD_DATA_EXCEPT_DATA	User data area.
WORK_FLAG st_work_flag	Refer to Appendix A for an explanation of the work flag user data area.
s_count	The number of RD_DATA_EXCEPT_ELEMENT structures that follow. For each data exception that is returned a RD_DATA_EXCEPT_ELEMENT structure will be filled in. Maximum number of exceptions defined by MAX_DATA_EXCEPTIONS.
RD_DATA_EXCEPT_ELEMENT *stp_elements	An array of data exception elements returned for each data structure. The user application must allocate an array of RD_DATA_EXCEPT_ELEMENT structures. Then the application must point *stp_elements to this allocated array.
RD_DATA_EXCEPT_ELEMENTS	An array of data exception elements returned for each data exception read.
uc_pointtype	The point type of the computer interface index for which an exception report has been returned.
ICI_TIMESTAMP st_timestamp	Refer to Appendix C for an explanation of the time stamp user data area. The actual time stamp structure.

Parameter	Description
s_index	The computer interface index for which an exception report has been returned.
ICI_TAGNAME st_tagname	A character array containing the tag name associated with a particular computer interface index.
VALUE_TABLE st_value	Refer to Appendix D for an explanation of the value table user data area.

Read Data Group

DESCRIPTION

Reads the status or value of a group of import points. This function returns the status or value data that is currently stored in the computer interface for the input point. Point index zero (the interface module status) can also be read with this command.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (work flag, INFI 90 OPEN address, point types)
l3read.h

FORMAT

wait access: **s_read_data_group_w**
*(s_log_ici, *stp_error, *stp_rd_data_group_data, l_data_size, *stp_group_args_param)*

quick access: **s_read_data_group_q**
*(s_log_ici, *stp_error, *stp_msg_id, *stp_group_args_param)*

Type	Parameter	Description
short	s_log_ici	Logical computer interface.
ERR_STRUCT	*stp_error	Pointer to an error structure.
RD_DATA_GROUP_DATA	*stp_rd_data_group_data	Pointer to user data area.
long	l_data_size	Size of user data area.
MSG_ID	*stp_msg_id	Refer to Appendix K for quick message ID structure.
GENERAL_GROUP_ARGS_PARAM	*stp_group_args_param	Structure containing a list of point indexes and index count.

POINT TYPES

PT_ANALOG
PT_CONTROL_POINT
PT_DAANG
PT_DADIG
PT_DD
PT_DIGITAL
PT_EXTENDED_MODULE_STATUS
PT_MODULE_STATUS
PT_MSDD
PT_PROCESS_VARIABLE
PT_RATIO_INDEX
PT_RCM
PT_REAL4_ANALOG_READ
PT_REM_MOTOR_CONTROL
PT_RMSC
PT_SET_POINT
PT_STATION
PT_STATION_STATUS
PT_TEXT_SELECTOR

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL Quick Access: ICI_OK ICI_FATAL ICI_WARNING

USER PARAMETERS

Parameter	Description
GENERAL_GROUP_ARGS_PARAM	User parameter area.
s_count	The number of computer interface indices that have been filled in the s_indices array. A maximum of MAX_READ_GROUP_COUNT indices are allowed in one function.
s_indices[ ]	An array of computer interface point indices. A maximum of MAX_READ_GROUP_COUNT indices are allowed to be read in a single computer interface function.
ICI_TAGNAME st_tagnames[ ]	An array containing the tag names associated with a particular computer interface index. A maximum of MAX_READ_GROUP_COUNT tag names are allowed to be read in a computer interface function.

USER DATA AREA

Parameter	Description
RD_DATA_GROUP_DATA	User data area.
WORK_FLAG st_work_flag	Refer to Appendix A for an explanation of the work flag user data area.
s_count	The number of RD_DATA_GROUP_ELEMENT structures that follow. For each point that data is returned for a RD_DATA_GROUP_ ELEMENT structure will be filled in. The maximum number of read data group indices that can be returned in each call is MAX_READ_ DATA_GROUP_COUNT.
RD_DATA_GROUP_ELEMENT *stp_elements	An array of elements returned for each data point. The user application must allocate an array of RD_DATA_GROUP_ELEMENT structures. Then the application must point *stp_elements to this allocated array.

Parameter	Description
RD_DATA_GROUP_ELEMENT	An array of data group elements returned for each value/status read.
s_index	The computer interface index for which data has been returned.
ICI_TAGNAME st_tagname	A character array containing the tag name associated with a particular computer interface index.
uc_point_type	The point type of the computer interface index for which data has been returned.
ICI_TIME_STAMP st_timestamp	Refer to Appendix C for an explanation of the time stamp user data area.
VALUE_TABLE st_data	Refer to Appendix D for an explanation of the value table user data area.

Read Data List

DESCRIPTION

Reads the status or value of a sequential list of import points. **READ DATA LIST** returns the status or value data that is currently stored in the computer interface for the import point. Point index zero (the interface module status) can also be read with this command.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (work flag, INFI 90 OPEN address, point types)
l3read.h

FORMAT

wait access: **s_read_data_list_w**
*(s_log_ici, *stp_error, *stp_rd_data_group_data, l_data_size, *stp_list_args_param)*

quick access: **s_read_data_list_q**
*(s_log_ici, *stp_error, *stp_msg_id, *stp_list_args_param)*

Type	Parameter	Description
short	s_log_ici	Logical computer interface.
ERR_STRUCT	*stp_error	Pointer to an error structure.
RD_DATA_GROUP_DATA	*stp_rd_data_group_data	Pointer to user data area.
long	l_data_size	Size of user data area.
MSG_ID	*stp_msg_id	Refer to Appendix K for quick message ID structure.
GENERAL_LIST_ARGS_PARAM	*stp_list_args_param	Structure containing a list of point indexes and index count.

POINT TYPES

PT_ANALOG
PT_CONTROL_POINT
PT_DAANG
PT_DADIG
PT_DD
PT_DIGITAL
PT_EXTENDED_MODULE_STATUS
PT_MODULE_STATUS
PT_MSDD
PT_PROCESS_VARIABLE
PT_RATIO_INDEX
PT_RCM
PT_REAL4_ANALOG_READ
PT_REM_MOTOR_CONTROL
PT_RMSC
PT_SET_POINT
PT_STATION
PT_STATION_STATUS
PT_TEXT_SELECTOR

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL Quick Access: ICI_OK ICI_FATAL ICI_WARNING

USER PARAMETERS

Parameter	Description
GENERAL_LIST_ARGS_PARAM	To define a list of indices a start index and a stop index is all that is needed. The list will then comprise the start index, the stop index, and all indices between the start and the stop indices. **NOTE**: The start index must be less than or equal to the stop index. Index lists are validated by the computer interface functions.
s_start	The starting index in the list of indices.
ICI_TAGNAME st_start_tagname	A character array containing the tag name of a particular computer interface start index.
s_stop	The stopping index in the list of indices.
ICI_TAGNAME st_stop_tagname	A character array containing the tag name of a particular computer interface stop index.

USER DATA AREA

Parameter	Description
RD_DATA_GROUP_DATA	User data area.
WORK_FLAG st_work_flag	Refer to Appendix A for an explanation of the work flag user data area.
s_count	The number of RD_DATA_GROUP_ ELEMENT structures that follow. For each point that data is returned for a RD_DATA_ GROUP_ELEMENT structure will be filled in. The maximum number of read data list indices that can be returned in each call is MAX_READ_DATA_LIST.
RD_DATA_GROUP_ELEMENT *stp_elements	An array of elements for each data point. The user application must allocate an array of RD_DATA_GROUP_ELEMENT structures. Then the application must point *stp_elements to this allocated array.
READ_DATA_GROUP_ELEMENTS	An array of elements for each data value/status read.
s_index	The computer interface index for which data has been returned.
ICI_TAGNAME st_tagname	A character array containing the tag name of a particular computer interface.
uc_point_type	The point type of the computer interface index for which data has been returned.
ICI_TIME_STAMP st_time_stamp	Refer to Appendix C for an explanation of the time stamp user data area. The actual time stamp structure.

Parameter	Description
VALUE_TABLE st_data	Refer to Appendix D for an explanation of the value table user data area. The actual value structure.

Read Data Specs

DESCRIPTION

Reads point specification information of established import points. Specifications include alarm limits like high alarm, low alarm, etc.

APPLICABLE LEVEL AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (work flag, INFI 90 OPEN address, point types)
l3read.h

FORMAT

wait access: **s_read_data_specs_w**
*(s_log_ici, *stp_error, *stp_rd_data_specs_data, l_data_size, s_max_specs*)

quick access: **s_read_data_specs_q**
*(s_log_ici, *stp_error,*stp_msg_id, s_max_specs)*

Type	Parameter	Description
short	s_log_ici	Logical computer interface.
ERR_STRUCT	*stp_error	Pointer to an error structure.
RD_DATA_SPECS_DATA	*stp_rd_data_specs_data	Pointer to user data area.
long	l_data_size	Size of user data area.
short	s_max_specs	Maximum number of specs to be returned.
MSG_ID	*stp_msg_id	Refer to Appendix K for quick message ID structure.

POINT TYPES

PT_ANALOG
PT_ASCII_STRING
PT_DAANG
PT_DADIG
PT_DD
PT_DIGITAL
PT_MSDD
PT_PROCESS_VARIABLE
PT_RCM
PT_REAL4_ANALOG_READ
PT_REAM_MOTOR_CONTROL
PT_RMSC
PT_SET_POINT
PT_STATION
PT_TEXT_SELECTOR

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL Quick Access: ICI_OK ICI_FATAL ICI_WARNING

USER PARAMETERS

Parameter	Description
s_max_specs	The maximum number of specifications to be returned with this function. A value of zero (0) will cause the computer interface to return as many data specifications as possible. A maximum value for this parameter is MAX_DATA_SPECS.

USER DATA AREA

Parameter	Description
READ_DATA_SPECS_DATA	User data area.
WORK_FLAG st_work_flag	Refer to Appendix A for an explanation of the work flag user data area.
s_count	The number of RD_DATA_SPECS_ELEMENT structures that follow. For each point that data is returned for a RD_DATA_SPECS_ELEMENT structure will be filled in. The maximum number of read data specifications that can be returned in each call is MAX_DATA_SPECS.
RD_DATA_SPECS_ELEMENT *stp_elements	An array of elements returned for each data type. The user application must allocate an array of RD_DATA_SPECS_ELEMENT structures. Then the application must point *stp_elements to this allocated array.
RD_DATA_SPECS_ELEMENT	An array of elements returned for each data spec read.
s_index	The computer interface index for which data has been returned.
ICI_TAGNAME st_tagname	A character array containing the tag name associated with a particular computer interface index.
uc_point_type	The point type of the computer interface index for which data has been returned.
SPEC_TABLE st_specs	Refer to Appendix G for an explanation of the spec table user data area. The actual specification structure.

Read Enhanced Block Output

DESCRIPTION

Reads the output of any function block. The function block does not have to be established in the computer interface point table. This function does not conform to the exception reporting scheme, is inefficient, and is not recommended for normal process input or output. The reply to this function is delayed until the module responds to the computer interface. The function operates exactly like **READ BLOCK OUTPUT** except that it will also return real-4 values and user defined data.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (work flag, INFI 90 OPEN address, point types)
l3read.h

FORMAT

wait access: **s_read_enh_block_output_w**
*(s_log_ici, *stp_error, *stp_rd_block_output, l_data_size, *stp_infi90adr_param)*

quick access: **s_read_enh_block_output_q**
*(s_log_ici, *stp_error, *stp_msg_id, *stp_infi90adr_param)*

Type	Parameter	Description
short	s_log_ici	Logical computer interface.
ERR_STRUCT	*stp_error	Pointer to an error structure.
RD_BLOCK_OUTPUT_DATA	*stp_rd_block_output	Pointer to user data area.
long	l_data_size	Size of user data area.
INFI90ADR	*stp_infi90adr_param	INFI 90 OPEN address structure.
MSG_ID	*stp_msg_id	Refer to Appendix K for quick message ID structure.

POINT TYPE

None. Does not refer to any point type in particular. It refers directly to the INFI 90 OPEN system (by address, loop, PCU, module, and block).

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL Quick Access: ICI_OK ICI_FATAL ICI_WARNING

USER PARAMETERS

Parameter	Description
INFI90ADR	INFI 90 OPEN address structure.
*stp_infi90_address	The INFI 90 OPEN address of the point.
s_loop	The loop number of the block.
s_node	The node (PCU) of the block.
s_module	The module number of the block.
s_block	The block number.

USER DATA AREA

Parameter	Description
RD_BLOCK_OUTPUT_DATA	User data area.
WORK_FLAG	Refer to Appendix A for an explanation of the work flag user data area.
st_work_flag	The actual work flag from the computer interface.
uc_report_type	The actual block output report type that is returned from the INFI 90 OPEN block. Determines which member of the un_report union to access. Note that blocks in INFI 90 OPEN return a default report type, this type is listed under NORMAL OPERATION. The computer interface can be forced to return values in a particular format by using DATA FORMATS to indicate the report type for this subroutine. The data formats report types are listed in the following text.
NORMAL OPERATION:	
READ_BLOCK_OUTPUT_REPORT_REAL2	Returns the floating point number in the INFI 90 OPEN Real-2 format. Access the data from the report_real structure.
READ_BLOCK_OUTPUT_REPORT_REAL3	Returns the floating point number in the INFI 90 OPEN Real-3 format. Access the data from the report_real structure.
READ_BLOCK_OUTPUT_REPORT_BOOLEAN	Returns the value as a digital value (one or zero). Access the data from the report_boolean structure.
READ_BLOCK_OUTPUT_REPORT_INT1	Returns the integer number in the INFI 90 OPEN Integer-1 format. Access the data from the report_integer structure.
READ_BLOCK_OUTPUT_REPORT_INT2	Returns the integer number in the INFI 90 OPEN Integer-2 format. Access the data from the report_integer structure.
READ_ENH_BLOCK_OUTPUT_REPORT_REAL4	Returns the floating point number in the INFI 90 OPEN Real-4 format. Access the report_real structure.

Parameter	Description
READ_ENH_BLOCK_ OUTPUT_REPORT_USER	Returns the user defined exception report data. Access the data from the report_user structure.
unreport	Union of block output data types. The uc_report_type field will indicate which member of the union to access for valid data.
report_real	Report for a block with a floating point value.
real	Name of the structure for a block with a floating point value.
uc_tracking	Status information about tracking. POINT_NOT_ TRACKING (0) is normal operation and POINT_ TRACKING (1) is tracking.
uc_deviation	Status information about the deviation alarm state. NO_ALARM (0) is normal operation, LOW_DEV_ LIMIT_EXCEEDED (1) is low deviation alarm, and HIGH_DEV_LIMIT_EXCEEDED (2) is high deviation alarm.
uc_high_low	Status information about the high/low alarm state. NO_ALARM (0) is normal operations (no alarm), LOW_LIMIT_EXCEEDED (1) is low alarm, and HIGH_LIMIT_EXCEEDED (2) is high alarm.
uc_quality	Status information about the quality. GOOD_QUALITY (0) is normal operations (good quality) and BAD_QUALITY (1) indicated bad quality.
uc_disabled	This is the status information about the disabled point (IMAMM02). Zero (0) is normal operations, and one (1) indicates the block is being serviced.
f_value	The floating point value for the block.
report_integer	Report format for a block with an integer value.
integer	Name of the structure for a block with an integer value.
s_value	The integer value of the block.
report_boolean	Report format for a block with a boolean value.
boolean	Name of the structure for a block with a boolean value.
uc_alarm	Status information about the alarm state. NORMAL (0) is normal operations (no alarm), and ALARM (1) indicates the block is in alarm.
uc_quality	Status information about the quality. GOOD_QUALITY (0) is normal operations (good quality) and BAD_QUALITY (1) indicates bad quality.
uc_value	Value of the boolean block. A value of ZERO (0) or ONE (1) are valid values for a boolean block.
report_user	Report format for a UDXR block.
user	Name of the structure for a UDXR block.

USER DATA AREA

Parameter	Description
WORK_FLAG st_work_flag	Refer to Appendix A for an explanation of the work flag user data area.

Read Performance Data

DESCRIPTION

This function provides computer interface performance data.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (work flag, INFI 90 OPEN address, point types)
l3mang.h

FORMAT

s_read_performance_data
*(s_log_ici, *stp_performance_data, *stp_error)*

Type	Parameter	Description
short	s_log_ici	Logical computer interface.
ERR_STRUCT	*stp_error	Pointer to an error structure.
PERFORMANCE_DATA	*stp_performance_data	User data area.

POINT TYPE

None. Does not refer to any point type in particular. It returns performance data about the computer interface module.

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL

USER DATA AREA

Parameter	Description
PERFORMANCE_DATA	User data area.
ul_app_mess_s	Number of messages sent by application.
ul_app_mess_r	Number of messages received by application.
ul_app_unsol	Number of unsolicited messages received by application.
ul_back_mess	Number of messages sent to the backup device driver.
ul_bytes_proc	Number of bytes processed by application.
mgr_ici_performance ici_perf_arr[2]	Array of performance structures. Array element 0 is for the primary computer interface and array element 1 is reserved .
mgr_ici_performance	Performance structure on a per computer interface basis.
ul_app_mess_s	Number of messages sent by application.
ul_app_mess_r	Number of messages received by application.

Parameter	Description
ul_app_unsol	Number of unsolicited messages received by application.
ul_bytes_proc	Number of bytes processed by application.
ul_dd_mess_tot	Total number of messages processed by the device driver.
ul_ici_mess_tot	Total number of messages processed.
ul_ser_mess_tot	Total number of service center message processed.
ul_uns_mess_tot	Total number of unsolicited messages processed.
ul_bytes_ici	Number of bytes processed by the computer interface.
ul_bytes_ser	Number of bytes processed by the service center.

Trend Data Poll

DESCRIPTION

Obtains trend data from a module. This function allows the host computer to access trend data collected in control modules. Set the start time field of the function to zero to receive the oldest data. Set this field to the six byte millisecond time of the last previous trend data value obtained to receive the next sequence of data.

The module uses the trend sample number field value to determine what trend data to send. The begin trend sample number field of the reply specifies the sample number of first trend data sent.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (work flag, INFI 90 OPEN address, point types)
l3read.h

FORMAT

wait access: **s_trend_data_poll_w**
*(s_log_ici,*stp_error, *stp_trend_data_poll_data, l_data_size, *stp_rd_trend_poll_param)*

quick access: **s_trend_data_poll_q**
*(s_log_ici,*stp_error, *stp_msg_id, *stp_rd_trend_poll_param)*

Type	Parameter	Description
short	s_log_ici	Logical computer interface.
ERR_STRUCT	*stp_error	Pointer to an error structure.
RD_TREND_POLL_DATA	*stp_rd_trend_data_poll_data	Pointer to user data area.
long	l_data_size	Size of user data area.
RD_TREND_POLL_PARAM	*stp_rd_trend_poll_param	Structure containing a list of point indexes and index count.
MSG_ID	*stp_msg_id	Refer to Appendix K for quick message ID structure.

POINT TYPE

None. Does not refer to any point type in particular. It refers directly to the INFI 90 OPEN system (by address, loop, PCU, module, and block).

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL Quick Access: ICI_OK ICI_FATAL ICI_WARNING

USER PARAMETERS

Parameter	Description
RD_TREND_POLL_PARAM	User parameter area.
INFI90ADR	INFI 90 OPEN address structure.
st_infi90_address	The INFI 90 OPEN address of the point.
s_loop	The loop number of the trend block to read.
s_node	The node (PCU) of the trend block to read.
s_module	The module number of the trend block to read.
s_block	The block number of the trend block.
uc_trend_type	The type of trend being polled. The valid values for uc_trend_type include:
ONE_MINUTE_TREND	One minute trends.
FIFTEEN_SECOND_TREND	Fifteen second trends.
l_sample_number	The trend sample number to read. The module used the sample number to determine which trend data to return. This parameter can be any number from 0 through 65,535.

USER DATA AREA

Parameter	Description
RD_TREND_POLL_DATA	User data area.
WORK_FLAG st_work_flag	Refer to Appendix A for an explanation of the work flag user data area.
uc_trend_type	The type of trend being polled. The valid values for uc_trend_type include:
ONE_MINUTE_TREND	One minute trends.
FIFTEEN_SECOND_TREND	Fifteen second trends.

Parameters	Description
uc_trend_mode	The mode that the trend is operating in. The valid values for uc_trend_mode include:
SAMPLE_TREND	A sample trend.
MEAN_TREND	A mean trend or average trend.
MINIMUM_TREND	A minimum trend.
MAXIMUM_TREND	A maximum trend.
SUM_TREND	A sum trend.
l_begin_sample_num	The sample number returned to the computer interface from the INFI 90 OPEN module.
uc_sample_count	The number of floating point trend values that have been filled in the f_trend_data array. A maximum of TREND_MAX_POINTS values can be inserted into the f_trend_data array.
f_trend_data[ ]	The array of floating point trend values. This array will contain uc_sample_count number of floating point numbers.

WRITE FUNCTIONS

This category of functions relate to writing data to the computer interface.

Output Control Point

DESCRIPTION

Writes to the following function blocks: RCM, RMSC, DAANG, DADIG, MSDD, DD, RMC, STATION and ASCII STRING. The user supplies the index associated with the function block, a key that indicates the type of operation that is performed, a tag name (optional) and the information appropriate to the type of operation to be performed.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (work flag, INFI 90 OPEN address, point types)
l3write.h

FORMAT

wait access: **s_output_control_point_w**
*(s_log_ici, *stp_output_control_params, *stp_error, *stp_work_flag_data, l_data_size)*

quick access: **s_output_control_point_q**
*(s_log_ici, *stp_output_control_params, *stp_error, *stp_msg_id,)*

Type	Parameter	Description
short	s_log_ici	Logical computer interface.
OUTPUT_CONTROL_PARAM	*stp_output_control_params	Pointer to user parameter area.
ERR_STRUCT	*stp_error	Pointer to an error structure.
WORK_FLAG	*stp_work_flag_data	Pointer to user data area.
long	l_data_size	Size of the user data area.
MSG_ID	*stp_msg_id	Refer to Appendix K for quick message ID structure.

POINT TYPES

PT_ASCII_STRING
PT_CONTROL_OUTPUT
PT_DAANG
PT_DADIG
PT_DD
PT_MSDD
PT_RATIO_INDEX_WRITTEN
PT_RCM
PT_REM_MOTOR_CONTROL
PT_RMSC
PT_SET_POINT_OUTPUT
PT_STATION
PT_STATION_MODE

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL Quick Access: ICI_OK ICI_FATAL ICI_WARNING

USER PARAMETERS

Parameter	Description
OUTPUT_CONTROL_PARAM	User parameter area.
uc_control_key	Indicates the kind of operation that is to be performed.
CHANGE_STATION_MODE	Changes mode of the station. (PT_STATION or PT_STATION_MODE)
CHANGE_DAANG_MODE	Changes the mode of the daang block. (PT_DAANG)
CHANGE_DAANG_VALUE	Changes the value and status of daang block. (PT_DAANG)
CHANGE_DADIG	Changes the value and status of a dadig block. (PT_DADIG)
CHANGE_RMSC	Changes the value and status of a RMSC block. (PT_RMSC)
CHANGE_RCM	Changes the value and status of a RCM block. (PT_RCM)
CHANGE_MSDD	Changes the value and status of a MSDD block. (PT_MSDD)
CHANGE_DD	Changes the value and status of a DD block. (PT_DD)
CHANGE_RMC	Changes the value and status of a RMC block. (PT_RMC)
CHANGE_SP_OUTPUT	Changes the value (but not the status) of a station. (PT_STATION) ***- or -*** Changes the value and status of a station. (PT_SET_POINT_OUTPUT)
CHANGE_CONTROL_OUTPUT	Changes the value (but not the status) of a station. (PT_STATION) ***- or -*** Changes the value and status of a station. (PT_CONTROLOUTPUT)
CHANGE_RATIO_INDEX	Changes the value (but not the status) of a station. (PT_STATION) ***- or -*** Changes the value and status of a station. (PT_RATIO_INDEX_WRITTEN)
CHANGE_ASCII_STRING	Writes ASCII strings to FC 194 block. (PT_ASCII_STRING)
s_index	Index of point to export data.
ICI_TAGNAME st_tagname	Tag name of point to export data to.
CONTROL_TYPE st_control	Union of various point types that can be output to. See tables that follow for a description of each.
CONTROL_TYPE	Union of possible point types.
CP_STATION st_station	Structure of station point.

Parameter	Description
CP_RMSC st_rmsc	Structure for a remote manual set constant (RMSC) point type.
CP_DAANG st_daang	Structure for a DAANG point type.
CP_MODE st_mode	Structure for a station or DAANG mode type.
CP_RCM st_rcm	Structure for a remote control memory (RCM) point type.
CP_MSDD st_msdd	Structure for a multistate device driver (MSDD) point type.
CP_DD st_dd	Structure for a device driver (DD) point type.
CP_RMC st_rmc	Structure for a remote motor control (RMC) point type.
CP_DADIG st_dadig	Structure for a DADIG point type.
CP_ASCII_STRING st_ascii_string	Structure for an ASCII string point type.
CP_STATION	Structure for a station point type.
c_quality [1]	Indicates the quality of the station.
GOOD_QUALITY	Indicates good quality.
BAD_QUALITY	Indicates bad quality.
c_limit_alarm[1]	Indicates the limit alarm.
NO_ALARM	Indicates no alarm condition.
HIGH_LIMIT_EXCEEDED	Indicates the high alarm limit has been exceeded.
LOW_LIMIT_EXCEEDED	Indicates the low alarm limit has been exceeded.
c_deviation_alarm[1]	Indicates the deviation alarm. A deviation alarm is a rate of change alarm.
NO_ALARM	Indicates no alarm condition.
HIGH_DEV_LIMIT_EXCEEDED	Indicates the high deviation alarm limit has been exceeded.
LOW_DEV_LIMIT_EXCEEDED	Indicates the low deviation alarm limit has been exceeded.
c_red_tagged[1]	Indicates whether the point is red tagged.
NOT_TAGGED	Indicates that the point is not red tagged.
RED_TAGGED	Indicates that the point is red tagged.
c_set_point_tracking[1]	Indicates whether the point is tracking.
POINT_NOT_TRACKING	Indicates the point has tracking disabled (off).
POINT_TRACKING	Indicates the point has tracking enabled (on).
f_value	The floating point value of the station.

NOTE: 1. Writing c_quality, c_limit_alarm, c_deviation_alarm, c_red_tagged, and c_set_point_tracking elements to a station block is not supported and has no effect on the station block. Use the zero state value for each of these fields.

Parameter	Description
CP_RMSC	Structure for a remote manual set constant (RMSC) point type.
c_quality[1]	Indicates the quality of the RMSC.
GOOD_QUALITY	Indicates good quality.
BAD_QUALITY	Indicates bad quality.
c_limit_alarm[1]	Indicates the limit alarm.
NO_ALARM	Indicates no alarm condition.
HIGH_LIMIT_EXCEEDED	Indicates the high alarm limit has been exceeded.
LOW_LIMIT_EXCEEDED	Indicates the low alarm limit has been exceeded.
c_deviation_alarm[1]	Indicates the deviation alarm. A deviation alarm is a rate of change alarm.
NO_ALARM	Indicates no alarm condition.
HIGH_DEV_LIMIT_EXCEEDED	Indicates the high deviation alarm limit has been exceeded.
LOW_DEV_LIMIT_EXCEEDED	Indicates the low deviation alarm limit has been exceeded.
c_red_tagged[1]	Indicates whether the point is red tagged.
NOT_TAGGED	Indicates that the point is not red tagged.
RED_TAGGED	Indicates that the point is red tagged.
c_set_point_tracking[1]	Indicates whether the point is tracking.
POINT_NOT_TRACKING	Indicates the point has tracking disabled (off).
POINT_TRACKING	Indicates the point has tracking enabled (on).
f_value	The floating point value of the remote manual set constant.
CP_DAANG	Structure for a DAANG point type.
f_value	The floating point value of the DAANG point.
CP_MODE	Structure for a station mode type.

NOTE: 1. Writing the c_quality, c_limit_alarm, c_deviation_alarm, c_red_tagged, and c_set_point_tracking elements to an RMSC block is not supported and does not affect the RMSC block. Use the zero state value for each of these fields.

Parameter	Description
CP_MSDD	Structure for a multistate device driver (MSDD) point type.
c_auto_mode_req	Indicates whether the automatic mode has been requested.
NO	Indicates the automatic mode has not been requested.
YES	Indicates the automatic mode has been requested.
c_manual_mode_req	Indicates whether the manual mode has been requested.
NO	Indicates the manual mode has not been requested.
YES	Indicates the manual mode has been requested.
c_requested_mask	Indicates the user requested mask value.
MASK_ZERO	Indicates a user mask of zero (0).
MASK_ONE	Indicates a user mask of one (1).
MASK_TWO	Indicates a user mask of two (2).
MASK_THREE	Indicates a user mask of three (3).
CP_DD	Structure for a device driver (DD) point type.

Parameter	Description
c_auto_mode_req	Indicates whether the automatic mode has been requested.
NO	The automatic mode has not been requested.
YES	The automatic mode has been requested.
c_manual_mode_req	Indicates whether the manual mode has been requested.
NO	The manual mode has not been requested.
YES	The manual mode has been requested.
c_reset_control_out	Indicates whether the reset control output is equal to zero (0).
DD_UNCHANGED	The reset control output signal has not changed.
SET_TO_ZERO	The reset control output signal is set to zero (0).
c_set_control_out	Indicates whether the set control output is equal to one (1).
DD_UNCHANGED	The set control output signal has not changed.
SET_TO_ONE	The set control output signal is set to one (1).
CP_RMC	Structure for a remote motor control (RMC) point type.
c_ack_fault	Indicates whether an acknowledge fault has occurred.
NO	Indicates an acknowledge fault has not occurred.
YES	Indicates an acknowledge fault has occurred.
c_reset	Indicates whether a reset signal has been received.
RMC_UNCHANGED	Indicates the RMC is unchanged.
RMC_STOPPED	Indicates the RMC is reset (stopped).
c_set	Indicates whether a set signal has been received.
RMC_UNCHANGED	Indicates the RMC is unchanged.
RMC_STARTED	Indicates the RMC is set (started).
CP_DADIG	Structure for a DADIG point type.
c_set_user_value	Indicates whether to set the user inserted value.
NO	Indicates not to set the user inserted value.
YES	Indicates to set the user inserted value.
c_reset_user_value	Indicates whether to reset the user inserted value.
NO	Indicates not to reset the user inserted value.
YES	Indicates to reset the user inserted value.

Parameter	Description
c_command	Indicates the actual command to the DADIG.
SELECT_USER_VALUE	Indicates to select the user inserted value.
SELECT_PRIMARY_VALUE	Indicates to select the primary value.
SELECT_ALTERNATE_VALUE	Indicates to select the alternate value.
ENABLE_ALARM_SUPPRESSION	Indicates to enable (turn on) alarm suppression.
DISABLE_ALARM_SUPPRESSION	Indicates to disable (turn off) alarm suppression.
DISABLE_REPORT_MODE	Indicates to disable (turn off) report mode.
ENABLE_REPORT_MODE	Indicates to enable (turn on) report mode.
FORCE_AN_XR	Indicates to force an exception report.
CLEAR_EXT_STATUS	Indicates to clear extended status.
REQUEST_INPUT_STATES	Indicates to request input states.
CP_ASCII_STRING	Structure for an ASCII string point type.
uc_sub_options	This is the sub options field of the function. This will indicate what the UDXR block should do. The following are status sub-options. In these sub-options the s_count parameter should be zero and there should be no data bytes to send to the UDXR block.
UDXR_GOTO_MANUAL	Indicates that the UDXR block should go to the manual mode.
UDXR_GOTO_AUTOMATIC	Indicates that the UDXR block should go to the automatic mode.
UDXR_SET_DATA	Indicates to set the UDXR data in the block.
uc_alarm_level	Indicates the alarm level of the UDXR block.
NORMAL	Indicates the UDXR block is not in an alarm condition.
ALARM	Indicates the UDXR block is in an alarm condition.
uc_sequence_number	Indicates the sequence number for the process data.
uc_byte_count	Indicates the byte count of the process data.
uc_data[ ]	The actual data bytes in the process data message. The number of bytes in this array is equal to s_count which has a maximum of MAX_SEND_PROC_DATA.

USER DATA AREA

Parameter	Description
WORK_FLAG	Refer to Appendix A for an explanation of the work flag user data area.

Output Report

DESCRIPTION

Output exception reports for the specified export points.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (work flag, INFI 90 OPEN address, point types)
l3write.h

FORMAT

wait access: **s_output_report_w**
*(s_log_ici, *stp_report_point_param, s_num_reports, *stp_error, *stp_work_flag_data, l_data_size)*

quick access: **s_output_report_q**
*(s_log_ici, *stp_report_point_param, s_num_reports, *stp_error, *stp_msg_id)*

Type	Parameter	Description
short	s_log_ici	Logical computer interface.
REPORT_POINT	*stp_report_point_param	Pointer to array of report points.
short	s_num_reports	Number of reports to output.
ERR_STRUCT	*stp_error	Pointer to an error structure.
WORK_FLAG	*stp_work_flag_data	Pointer to user data.
long	l_data_size	Size of user data area.
MSG_ID	*stp_msg_id	Refer to Appendix K for quick message ID structure.

POINT TYPES

PT_ANALOG_REPORT	PT_REAL4_ANALOG_REPORT
PT_DIGITAL_REPORT	PT_RMSC_REPORT
PT_RCM_REPORT	PT_STATION_REPORT

NOTE: 1. Only the following point types are valid for Data Aquisition (DA) users: PT_ANALOG_REPORT, and PT_DIGITAL_REPORT.

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL Quick Access: ICI_OK ICI_FATAL ICI_WARNING

USER PARAMETERS

Parameter	Description
REPORT_POINT	User parameter area.
c_pt_type	The point type for the report to be output. The point type will indicate which member of the REPORT_TYPE union will be filled in.
PT_ANALOG_REPORT	Analog point type. Access the ANALOG_REPORT member of the union.
PT_REAL4_ANALOG_REPORT	Analog point type. Access the ANALOG_REPORT member of the union. Not valid for DA users.
PT_DIGITAL_REPORT	Digital point type. Access the DIGITAL_REPORT member of the union.
PT_RCM_REPORT	Remote Control Memory (RCM) point type. Access the RCM_REPORT member of the union. Not valid for DA users.
PT_RMSC_REPORT	Remote Manual Set Constant (RMSC) point type. Access the RMSC_REPORT member of the union. Not valid for DA users.
PT_STATION_REPORT	Station point type. Access the STATION_REPORT member of the union. Not valid for DA users.
s_pt_index	The computer interface point index of the report point to output the report to.
ICI_TAGNAME st_tagname	A character array containing the tag name associated with a particular computer interface index.
REPORT_TYPE report	A union of various report point types. The c_pt_type member indicates which member of the structure to access.
REPORT_TYPE	A union of various report point types. The c_pt_type member indicates which member of the structure to access.
ANALOG_REPORT	Format of analog report.
analog	Name of the analog report.
DIGITAL_REPORT	Format of digital report.
digital	Name of the digital report.
STATION_REPORT	Format of station report.
station	Name station report. Not valid for DA users.
RCM_REPORT	Format of RCM report.
rcm	Name of the RCM report. Not valid for DA users.
RMSC_REPORT	Format of RMSC report.
rmsc	The name of the RMSC report. Not valid for DA users.
ANALOG_REPORT	Format of analog report.
c_quality	Indicates the quality of the analog point.
GOOD_QUALITY	Indicates good quality.
BAD_QUALITY	Indicates bad quality.

Parameter	Description
c_limit_alarm	Indicates the limit alarm. All station variables have the same limit alarm as the process variable (PV).
NO_ALARM	Indicates no alarm condition.
HIGH_LIMIT_EXCEEDED	Indicates the high alarm limit has been exceeded.
LOW_LIMIT_EXCEEDED	Indicates the low alarm limit has been exceeded.
c_cal_correct_val	Indicates the status of the calibration correction values.
OK	Indicates the correction values are within range.
OUT_OF_RANGE	Indicates the correction values are out of range.
c_red_tagged	Indicates whether the point is red tagged.
NOT_TAGGED	Indicates the point is not red tagged.
RED_TAGGED	Indicates the point is red tagged.
c_point_tracking	Indicates whether the point is tracking.
POINT_NOT_TRACKING	Indicates the point has tracking disabled (off).
POINT_TRACKING	Indicates the point has tracking enabled (on).
f_value	The floating point value for the analog point.
DIGITAL_REPORT	Format of digital report.
c_quality	Indicates the quality of the digital point.
GOOD_QUALITY	Indicates good quality.
BAD_QUALITY	Indicates bad quality.
c_limit_alarm	Indicates if the limit alarm has been exceeded.
NO	Indicates the limit alarm has not been exceeded.
YES	Indicates the limit alarm has been exceeded.
c_value	Indicates the value of the digital point.
ZERO	The digital has a value of zero (0).
ONE	The digital has a value of one (1).
RCM_REPORT	Format of RCM report.
c_quality	Indicates the quality of the RCM point.
GOOD_QUALITY	Indicates good quality.
BAD_QUALITY	Indicates bad quality.
c_alarm	Indicates the alarm status.
NORMAL	Indicates normal operation (no alarm).
ALARM	Indicates an alarm condition.
c_tagged	Indicates the red tag status.
NOT_TAGGED	Indicates the RCM IS NOT red tagged.
RED_TAGGED	Indicates the RCM IS red tagged.

Parameter	Description
c_output_value	Indicates the output value of the RCM.
ZERO	The RCM has a value of zero (0).
ONE	The RCM has a value of one (1).
c_log_set_input_rec	Indicates the logic set input received status.
NO	Indicates the logic set input has not been received.
YES	Indicates the logic set input has been received.
c_set_permissive_input_rec	Indicates the set permissive input received status.
NO	Indicates the set permissive input has not been received.
YES	Indicates the set permissive input has been received.
c_log_reset_input_rec	Indicates the logic reset input received status.
NO	Indicates the logic reset input has not been received.
YES	Indicates the logic reset input has been received.
c_override	Indicates the override status.
NO	Indicates that the value is not being overriden.
YES	Indicates that the value is being overriden.
c_feedback	Indicates the feedback status.
ZERO	Indicates that there is a feedback signal of zero (0).
ONE	Indicates that there is a feedback signal of one (1).
c_set_command_rec	Indicates the set command received status.
NO	Indicates the set command has not been received.
YES	Indicates the set command has been received.
c_reset_command_rec	Indicates the reset command received status.
NO	Indicates the reset command has not been received.
YES	Indicates the reset command has been received.
STATION_REPORT	Format of station report.
c_quality	Indicates the quality of the station.
GOOD_QUALITY	Indicates good quality.
BAD_QUALITY	Indicates bad quality.
c_limit_alarm	Indicates the limit alarm. All station variables have the same limit alarm as the process variable (PV).
NO_ALARM	Indicates no alarm condition.
HIGH_HIGH_LIMIT_EXCEEDED	The high alarm limit has been passed.
LOW_LIMIT_EXCEEDED	The low alarm condition has been passed.

Parameter	Description
c_dev_alarm	Indicates the deviation alarm. A deviation alarm is a rate of change alarm. This is the difference between the set point (SP) and the process variable (PV) for a station.
NO_ALARM	Indicates no alarm condition.
HIGH_DEV_LIMIT_EXCEEDED	Indicates the high deviation alarm limit has been exceeded.
LOW_DEV_LIMIT_EXCEEDED	Indicates the low deviation alarm limit has been exceeded.
c_red_tagged	Indicates whether the point is red tagged.
NOT_TAGGED	Indicates the point is not red tagged.
RED_TAGGED	Indicates the point is red tagged.
c_set_point_tracking	Indicates whether the point is tracking.
POINT_NOT_TRACKING	Indicates the point has tracking disabled (off).
POINT_TRACKING	Indicates the point has tracking enabled (on).
c_bypassed	Indicates whether the station is in the bypass mode.
NO	Indicates the station is not in the bypass mode.
YES	Indicates the station is in the bypass mode.
c_interlock	Indicates whether the station is in manual interlock mode.
DISABLED	Indicates manual interlock mode is disabled.
ENABLED	Indicates manual interlock mode is enabled.
c_output_tracking	Indicates whether output tracking is enabled on the station.
POINT_NOT_TRACKING	Indicates the output has tracking disabled (off).
POINT_TRACKING	Indicates the output has tracking enabled (on).
c_digital_failure	Indicates a failure in the digital station.
NO	The digital station has not failed.
YES	The digital station has failed.
c_computer_ok	Indicates whether the COMPUTER OK signal has been received from the host computer.
NOT_RECEIVED	Indicates the computer OK signal has not been received from the host.
RECEIVED	Indicates the computer OK signal has been received from the host.
c_control_level	Indicates the control level of the station.
STATION_LOCAL_LEVEL	Indicates the station is in the local level.
STATION_COMPUTER_LEVEL	Indicates the station is in the computer level.
c_ratio_control	Indicates whether the Cascade/Ratio control strategy is enabled.
DISABLED	Indicates the control strategy is disabled.
ENABLED	Indicates the control strategy is enabled.
c_automatic_manual	Indicates the mode of the station.
STATION_AUTOMATIC_MODE	Indicates the station is in the automatic mode of operation.
STATION_MANUAL_MODE	Indicates the station is in the manual mode of operation.
f_process_variable	The floating point value of the process variable.
f_set_point	The floating point value of the set point.

Parameter	Description
f_control_output	The floating point value of the control output.
f_ratio_index	The floating point value of the ratio index.
RMSC_REPORT	
c_quality	Indicates the quality of the RMSC.
GOOD_QUALITY	Indicates good quality.
BAD_QUALITY	Indicates bad quality.
c_limit_alarm	Indicates the limit alarm.
NO_ALARM	Indicates no alarm condition.
HIGH_LIMIT_EXCEEDED	Indicates the high alarm limit has been exceeded.
LOW_LIMIT_EXCEEDED	Indicates the low alarm limit has been exceeded.
c_dev_alarm	Indicates the deviation alarm. A deviation alarm is a rate of change alarm.
NO_ALARM	Indicates no alarm condition.
HIGH_DEV_LIMIT_EXCEEDED	Indicates the high deviation alarm limit has been exceeded.
LOW_DEV_LIMIT_EXCEEDED	Indicates the low deviation alarm limit has been exceeded.
c_red_tagged	Indicates whether the point is red tagged.
NOT_TAGGED	Indicates the point is not red tagged.
RED_TAGGED	Indicates the point is red tagged.
c_set_point_tracking	Indicates whether the point is tracking.
POINT_NOT_TRACKING	Indicates the point has tracking disabled (off).
POINT_TRACKING	Indicates the point has tracking enabled (on).
f_value	The floating point value of the remote manual set constant.

USER DATA AREA

Parameter	Description
WORK_FLAG	Refer to Appendix A for an explanation of the work flag user data area.

TIME FUNCTIONS

This category contains functions that read and set the time on the INFI 90 OPEN system via the computer interface.

Read System Date and Time

DESCRIPTION

This function sends the function to read the system date and time, loop and node address of the module containing the master time synchronization signal, time stamp value (absolute time), and wall clock offset value.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (work flag, INFI 90 OPEN address, point types)
l3time.h

FORMAT

wait access: **s_read_system_date_time_w**
*(s_log_ici, *stp_error, *stp_sdt_data, l_data_size)*

quick access: **s_read_system_date_time_q**
*(s_log_ici, *stp_error, *stp_msg_id)*

Type	Parameter	Description
short	s_log_ici	Logical computer interface.
ERR_STRUCT	*stp_error	Pointer to an error structure.
SDT_TIME	*stp_sdt_data	Outputs pointer to the time structure.
long	l_data_size	Size of the time structure.
MSG_ID	*stp_msg_id	Refer to Appendix K for quick message ID structure.

POINT TYPE

None. Does not refer to any point type in particular. Reads the date and time in the computer interface.

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL Quick Access: ICI_OK ICI_FATAL ICI_WARNING

USER PARAMETERS None. There are no additional user parameters other than the logical computer interface.

USER DATA AREA

Parameter	Description
SDT_TIME	User data area.
WORK_FLAG st_work_flag	Refer to Appendix A for an explanation of the work flag user data area.
s_synced	Indicates whether the computer interface has been time synchronized by the INFI 90 OPEN loop. If the value is a one (1) the computer interface has been time synchronized if the value is a zero (0) then the computer interface has not been time synchronized.
FORM_TIME st_form_time	This is the actual time structure.
l_sample	This is the trend sample number.
s_looptm	This is the loop number of the time sync master on the INFI 90 OPEN loop.
s_nodetm	This is the node (PCU) number of the time sync master on the INFI 90 OPEN loop.
c_timestamp[ ]	This is the six byte timestamp value. This is an absolute time (usually GMT time) in milliseconds. Not available on Plant Loop systems.
c_wallclock_offset[ ]	This is a six byte time offset. It represents the difference in milliseconds between the timestamp (c_timestamp) and the local time. Not available on Plant Loop systems.
s_accuracy	This parameter is not used for this computer interface function.
s_timezone	This parameter is not used for this computer interface function.
s_option	This parameter is not used for this computer interface function.
l_wallclock	This parameter is not used for this computer interface function.
FORM_TIME st_form_time	The actual time structure.
s_year	A number representing the year. The valid values for year are 0-99. 0 to 48 represent the years 2000 to 2048. 87 to 99 represent the years 1987 to 1999.
s_month	A number representing the month. The valid values for month are 1-12. 1 is January and 12 is December.
s_day	A number representing the day. The valid values for day are 1-31.
s_weekday	A number representing the day of the week. The valid values for weekday are 1-7. 1 is Sunday and 7 is Saturday.
s_hour	A number representing the hour. The clock is based on a 24-hour clock. The valid values for hour are 0-23.
s_minute	A number representing the minute. The valid values for minute are 0-59.
s_second	A number representing the second. The valid values for second are 0-59.
s_hund_sec	This parameter is not used for this computer interface function.
s_tzdif	This parameter is not used for this computer interface function.

Set System Time and Date

DESCRIPTION

Allows the host computer to set the system date and time of all the nodes on the system that have enabled the time synchronization option of **RESTART**. If this option is not selected, the computer interface expects another node to time synchronize the system. If no node synchronizes the system, this function may be used to set the local date and time. The use of the time zone functionality must be system wide. Unpredictable results may occur if some computer interfaces compensate for time zones and others do not.

The year, month, day, hour, minute, and second fields are used to set the local time of this host computer. The time zone difference field represents the difference (in minutes) between the local time and the absolute time. Coordinated Universal Time or Greenwich Mean Time should be used as absolute time but any time can be used. The time zone difference can be zero, positive, or negative depending on the chosen absolute time. Refer to the following example for more information.

Example: A single INFI-NET loop spans the world. There are four computer interfaces on this loop. They are located in Los Angeles, Cleveland, London, and Moscow. To have absolute time represent time in Cleveland, set the time zone difference as follows:

City	Local Time	Time Zone Difference (min.)	Absolute Time (Local Time – Time Zone Difference)
Los Angeles	2:00 AM	-180	5:00 AM
Cleveland	5:00 AM	0	5:00 AM
London	10:00 AM	+300	5:00 AM
Moscow	1:00 PM	+540	5:00 AM

To have Coordinated Universal Time or Greenwich Mean Time to represent the absolute time, set the time zone difference as follows:

City	Local Time	Time Zone Difference (min.)	Absolute Time (Local Time – Time Zone Difference)
Los Angeles	2:00 AM	-480	10:00 AM
Cleveland	5:00 AM	-300	10:00 AM
London	10:00 AM	0	10:00 AM
Moscow	1:00 PM	+240	10:00 AM

Wait at least eight minutes after the computer interface becomes the primary (online) for a time sync message from another node before issuing this function. After eight minutes it is assumed that there is no time sync master on the loop and your computer interface can then be set. Issuing this function prior to eight minutes will result in an error code

message. This function causes the computer interface to synchronize all nodes in the system.

> **NOTE:** To set the system time and date of all nodes in the system, the ICI interface must have the highest time sync accuracy of all nodes on the system. Refer to Appendix M for details on setting the ICI interface default time sync accuracy. If the ICI does not have the highest time sync accuracy, the return status will be *ICI_OK* but the system time will not change.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (work flag, INFI 90 OPEN address, point types)
l3time.h

FORMAT

wait access: **s_set_system_date_time_w**
*(s_log_ici, *stp_error, *stp_work_flag_data, l_data_size, *stp_form_time)*

quick access: **s_set_system_date_time_q**
*(s_log_ici, *stp_error,*stp_msg_id, *stp_form_time)*

Type	Parameter	Description
short	s_log_ici	Logical computer interface.
ERR_STRUCT	*stp_error	Pointer to an error structure.
WORK_FLAG	*stp_sdt_data	Outputs pointer to the time structure.
long	l_data_size	Size of the time structure.
FORM_TIME	*stp_form_time	The pointer to a time structure parameter.
MSG_ID	*stp_msg_id	Refer to Appendix K for quick message ID structure.

POINT TYPE

None. Does Not Refer To Any Point Type In Particular. This Function Sets The System Date And Time In The Computer Interface.

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL Quick Access: ICI_OK ICI_FATAL ICI_WARNING

USER PARAMETERS

Parameter	Description
FORM_TIME	User parameter area.
s_year	A number representing the year. The valid values for year are 0-99. 0 to 48 represents the years 2000 to 2048. 87 to 99 represents the years 1987 to 1999.
s_month	A number representing the month. The valid values for month are 1-12. 1 is January and 12 is December.
s_day	A number representing the day. The valid values for day are 1-31.
s_weekday	This parameter is not used for this computer interface function.
s_hour	A number representing the hour. The clock is based on a 24-hour clock. The valid values for hour are 0-23.
s_minute	A number representing the minute. The valid values for minute are 0-59.
s_second	A number representing the second. The valid values for second are 0-59.
s_hund_sec	A number representing hundredth of seconds. The valid values for hundreds of seconds is 0-99.
s_tzdif	On the first execution of this function, the time zone difference field is used to coordinate the system date and time of interconnected computer interfaces located in different time zones. If all the computer interfaces are located in the same time zone, set the time zone difference field to zero. On subsequent function executions, this field is ignored.

USER DATA AREA

Parameter	Description
WORK_FLAG st_work_flag	Refer to Appendix A for an explanation of the work flag user data area.

MANAGER FUNCTIONS

This category of functions relate to communications between the computer interface and the application.

Enable Management

PURPOSE

Enables the application to be the manager of the computer interface. It allows an application to receive index updates. Index updates will notify an application of points being established in the computer interface by other applications that are connected to the computer interface.

Enabling management may return a warning if another application is already the manager of the computer interface. In this case, the requesting user will be signed up to be the manager. If the existing manager exits, the next manager in line takes over management.

The following summarizes the manager functions. The following functions will occur in the background and will appear transparent to the user application:

1. The manager keeps track of all points that are established in a computer interface to make it possible to rebuild a point table after an interface module fails.

2. The manager will monitor the status of an interface through in-line code in *s_general_onebyte_reply*. This function is called by ***all*** functions of the semAPI function library.

3. The manager provides failure recovery logic that is responsible for keeping a computer interface on-line and running at all times.

4. The failure control logic is responsible for restarting the dead interface and downloading the point table.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (work flag, INFI 90 OPEN address, point types)
l3qual.h
l3mang.h

FORMAT

s_enable_ici_management
*(s_mgr_ups, s_ind_ups, *stp_error)*

Type	Parameter	Description
short	s_mgr_ups	Indicates if the application is the manger. If yes, then user wishes to enable management. If no, then user disables management.
short	s_ind_ups	Indicates if the application wants index updates. If YES, then user wishes index updates. If NO, then user does not want index updates.
ERR_STRUCT	*stp_error	Error structure.

POINT TYPE

None. Does not refer to any point type in particular. It causes the application program to connect to the device driver task. This function is required in order to communicate with the computer interface hardware module.

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL

Read ICI Status

PURPOSE

Returns status information about a computer interface. An application does not have to be manager to use this function.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (work flag, INFI 90 OPEN address, point types)
l3mang.h

FORMAT

s_read_ici_status
*(s_log_ici, *stp_read_ici_status_data, *stp_error)*

Type	Parameter	Description
short	s_log_ici	Logical computer interface.
READ_ICI_STATUS_DATA	*stp_read_ici_status_data	User data area.
ERR_STRUCT	*stp_error	Error structure.

POINT TYPE

None. Does not refer to any point type in particular. It returns the status of computer interface to the user. It indicates the current status of the computer interface hardware module.

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL

USER DATA AREA

Parameter	Description
READ_ICI_STATUS_DATA	User data area.
s_current_ici	Indicates the current (communicating) ICI.
ICI_PRIMARY_CUR	The primary ICI is the current device.
ICI_BACKUP_CUR	The backup ICI is the current device.
s_is_backup_config	Reserved.
BACKUP_CONFIGURED	
BACKUP_NOT CONFIGURED	

Parameter	Description
s_primary_ici_status	Status of the primary ICI status.
OFF_LINE_ICI	The ICI is off-line.
DEAD_ICI	The ICI is dead.
DYING_ICI	The ICI is terminal.
RESTARTING_ICI	The ICI is restarting.
DISCONNECTED_ICI	ICI is disconnected.
OK_ICI	ICI is ok.
c_primary_device[ ]	Name of the primary device. (i.e. COM2.)
c_primary_node[ ]	Indicates the network name that the ICI device resides on.
us_primary_ici_type	The type of primary ICI module.
us_primary_ici_mode	The mode of the primary ICI.
ICI_TIMESTAMP st_primary_online_time	The time the primary ICI module was put on-line.
s_backup_ici_status	Reserved.
OFF_LINE_ICI	
DEAD_ICI	
DYING_ICI	
RESTARTING_ICI	
DISCONNECTED_ICI	
OK_ICI	
c_backup_device[ ]	Reserved.
c_backup_node[ ]	Reserved.
us_backup_ici_type	Reserved.
us_backup_ici_mode	Reserved.
ICI_TIME_STAMP st_backup_online_time	Reserved.
uc_loop	Loop number of the ICI module
uc_node	Node (PCU) number of the ICI module
uc_communication_protocol	Indicates the communication protocol of the ICI module.
ICI_IS_SCSI	SCSI ICI.
ICI_IS_SERIAL	Serial ICI.

Force Restart

PURPOSE

Initiates a forced restart of a computer interface. Function operates as follows. The ICI modules will restart with the restart parameters of the last successfully issued **RESTART**.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (work flag, INFI 90 OPEN address, point types)
l3mang.h

FORMAT

s_force_restart
*(s_log_ici, *stp_error, s_download*)

Type	Parameter	Description
short	s_log_ici	Logical computer interface.
ERR_STRUCT	*stp_error	Error structure.
short	s_download	Indicates if the computer interface point table should be downloaded after a RESTART. NO_DOWNLOAD. Does not download point table. DOWNLOAD_POINT_TABLE. Downloads point table.

POINT TYPE

None. Does not refer to any point type in particular. It refers to the computer interface directly. It causes the interface module to restart and clear its internal memory.

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL

MISCELLANEOUS FUNCTIONS

This category of functions does not relate to any specific semAPI functions.

Cancel Quick Message

DESCRIPTION

Cancels a currently outstanding quick function.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (work flag, INFI 90 OPEN address, point types)
l3misc.h

FORMAT

s_cancel_quick
*(s_msg_num, *stp_error)*

Type	Parameter	Description
short	s_msg_num	Quick message to cancel.
ERR_STRUCT	*stp_error	Pointer to an error structure.

POINT TYPE

None. Does not refer to any point type in particular. It cancels an outstanding quick access semAPI call.

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL

ICI Error Text

PURPOSE

Returns an English text string with ia particular error number in the error sturcture. Use this function in a loop to generate text error strings with all of the error leels that are set in an error structure.

The #define MAX_ERROR_LENGTH defines the maximum number of characters that are copied into the cp_buff parameter. Be sure to have a character array declared large enough to accommodate the error text string.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (work flag, INFI 90 OPEN address, point types)
l3mang.h

FORMAT

s_ici_error_text
*(*stp_error, *sp_next, *cp_buff)*

Type	Parameter	Description
ERR_STRUCT	*stp_error	Error structure.
short	*sp_next	Index value into the error structure. Set to zero (0) on first call. Do not modify the value of this parameter after the first call.
char	*cp_buff	Error text string.

POINT TYPE

None. Does not refer to any point type in particular. It converts the ERR_STRUCT structure into printable text describing the error.

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL

EXAMPLE

```
ERR_STRUCT st_error;
short s_more_errors = ICI_CONTINUE;
short s_next=0;
char c_message[MAX_ERROR_LENGTH];
.
.
.
.
while (s_more_errors ==ICI_CONTINUE)

{
s_more_errors = s_ici_error_text (&st_error, &s_next, &c_message[0]);
printf ("error: %s\n", &c_message[0]);

}
```

Retrieve Quick Message

DESCRIPTION

Retrieves the response for a semAPI function that was invoked using the Quick Access method. The quick access method replies are buffered internally. The application is responsible for retrieving the responses to the quick access commands periodically to keep the internal buffers cleaned out.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (work flag, INFI 90 OPEN address, point types)
l3misc.h

FORMAT

s_retrieve_quick_reply
*(s_msg_num, *cp_data, l_data_size, *stp_error)*

Type	Parameter	Description
short	s_msg_num	Quick message to retrieve.
char	*cp_data	User data area.
long	l_data_size	Size of user data area.
ERR_STRUCT	*stp_error	Pointer to an error structure.

POINT TYPE

None. Does not refer to any point type in particular. It gets the response to an outstanding quick access semAPI call.

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL ICI_CONTINUE

Read Work Flag

DESCRIPTION

Determines whether other semAPI functions need to be issued. If the work flag bytes are not set when issuing this function, the computer interface waits (no longer than the maximum waiting period parameter) for the work flag bytes to become set before replying.

APPLICABLE LEVELS AND MODULES

User Level	DA
	SC
Applicable Interface Modules	INICI03
	INOSM01

HEADER FILE

ici_err.h (error structure)
ici_user.h (work flag, INFI 90 OPEN address, point types)
l3misc.h

FORMAT

wait access: **s_read_work_flag_w**
*(s_log_ici, *stp_error, s_max_wait_time, *stp_work_flag_data l_data_size)*

quick access: **s_read_work_flag_q**
*(s_log_ici, *stp_error, s_max_wait_time, *stp_msg_id)*

Type	Parameter	Description
short	s_log_ici	Logical computer interface.
ERR_STRUCT	*stp_error	Pointer to an error structure.
short	s_max_wait_time	Short containing time in milliseconds to wait for flag to be set.
WORK_FLAG	*stp_work_flag_data	Pointer to user data area.
long	l_data_size	Size of user data area.
MSG_ID	*stp_msg_id	Refer to Appendix K for quick message ID structure.

POINT TYPE

None. Does not use any particular point type. This function reads the work flag in the computer interface.

RETURNS

Type	Parameter	Description
short	s_status	Wait Access: ICI_OK ICI_WARNING ICI_FATAL Quick Access: ICI_OK ICI_FATAL ICI_WARNING

USER PARAMETERS

Parameter	Description
s_max_wait_time	The time in milliseconds to wait for the work flag to be set.

USER DATA AREA

Parameter	Description
WORK_FLAG	Refer to Appendix A for an explanation of the work flag user data area.

SECTION 7 - TEST PROGRAM (TALK90)

INTRODUCTION

This section describes a test program that is used to test some of the semAPI functions. The menu driven program allows the user to call semAPI functions using various parameter combinations.

TALK90 DESCRIPTION

This application allows access to supported API function calls, providing the user an interactive method of accessing semAPI commands. When TALK90 is invoked, a menu lists the supported semAPI commands. After selecting an option from the menu, a prompt asks for all of the data necessary to issue the function. After entering the data, the actual semAPI function is invoked, passing the user supplied parameters. TALK90, upon return of the semAPI function, prints the reply code from the computer interface along with the decoded data that has been passed back from the computer interface. Figure 7-1 is a flowchart showing how to use semAPI with TALK90.

TALK90 OPERATION

Use the following procedure to invoke TALK90 HP-UX:

1. Log in to the HP-UX work station.

2. Run the TALK90 application. Type:

   cd /icl/ exe Enter

   talk90 Enter

When the TALK90 main menu appears, it extends into two screens. When selecting *ICI Management* or *Quick Commands* from the menu, subselections appear. To select from the menus, type the number of the semAPI function into the *Option* field. The TALK90 program will prompt for the appropriate parameters.

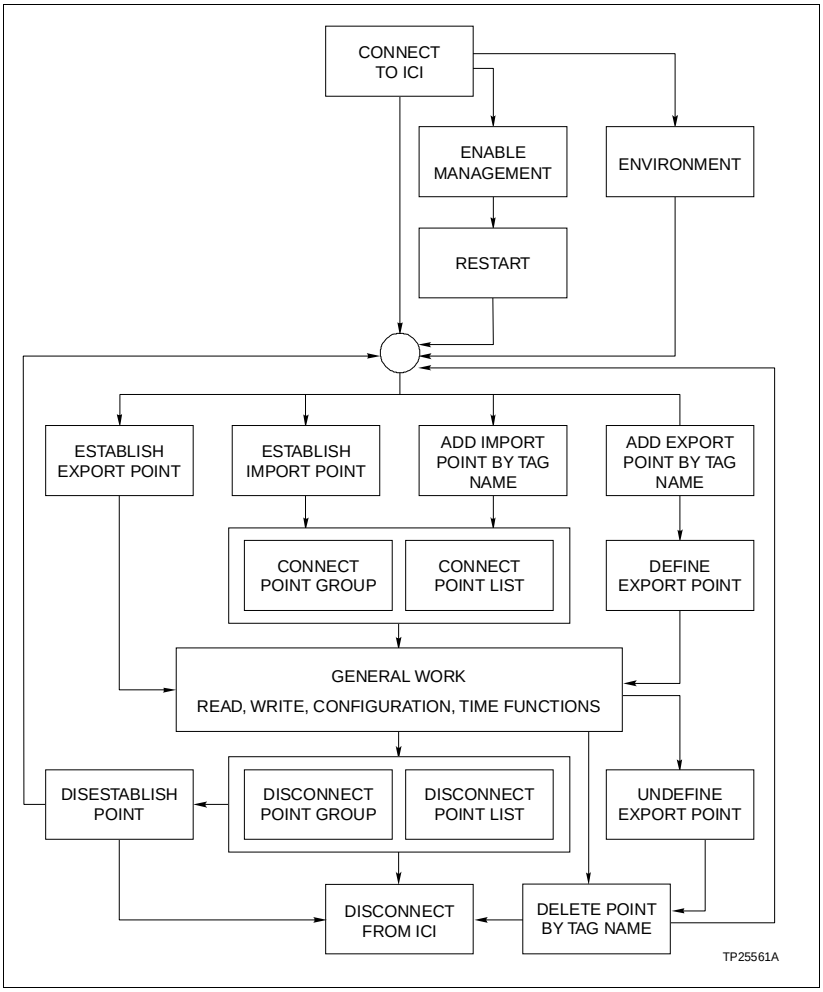


Figure 7-1. semAPI Usage Flowchart

SECTION 8 - APPLICATION EXAMPLE

INTRODUCTION

This section provides a sample application using the Strategic Enterprise Management Application Programming Interface (semAPI) functions. This program communicates with only one ICI communication module.

SAMPLE PROGRAM

```
/*****************************************************************************/
/*    FILE:                SAMPLE1.C                                         */
/*                                                                           */
/*    SOFTWARE:            ANSI C Compiler                                   */
/*                                                                           */
/*    DESCRIPTION:         This file contains sample code for using the semAPI function set. It will connect to the */
/*                         user specifies logical ICI, restart that logical ICI, establish four analog import points and */
/*                         four digital import points, connect the points and issue a read data list to obtain the */
/*                         floating point values and statuses of the analog import points. If the program fails in any */
/*                         of these operations, an error will be reported to the user and the program will terminate. */
/*                                                                           */
/*    HISTORY:                                                               */
/*    Version              Date          Name            Description         */
/*    -----------          -------       ---------------  -------------      */
/*    1.0                  25-OCT-94     Joe Sample      Original            */
/*****************************************************************************/


/*****************************************************************************/
/*                                INCLUDE FILES                              */
/*****************************************************************************/
#include <stdio.h>
#include <stdlib.h>

#include <ici_err.h>
#include <ici_time.h>
#include <ici_user.h>
#include <l3icconf.h>
#include <l3mang.h>
#include <l3read.h>
#include <l3qual.h>
#include <platform.h>
#if ICI_PC
#include <mem.h>
#include <dos.h>
#endif
```

```
/*********************************************************************************/
/*                          LOGICAL PROTOTYPES                                   */
/*********************************************************************************/
void v_print_error_struct(
        short,
        ERR_STRUCT *) ;
short s_print_data(
        RD_DATA_GROUP_DATA *,
        RD_DATA_GROUP_ELEMENT *) ;
short s_setup_estab_import_pt_call(
        short,      /* Logical ICI */
        short,      /* Point Index */
        short,      /* Point Type */
        short,      /* Loop number */
        short,      /* Node or PCU number */
        short,      /* Module Number */
        short);     /* Block Number */
short s_setup_read_data_list_call(
        short,      /* Logical ICI */
        short, /* First Index to read */
        short) ;    /* Last Index to read */
short s_setup_restart_call(
        short); /*Logical ICI */


/*********************************************************************************/
/*                          LOCAL DEFINES                                        */
/*********************************************************************************/
#define S_NUM_ANALOGS               4
#define S_NUM_DIGITALS              4
#define S_NUM_TOTAL_POINTS          S_NUM_ANALOGS + S_NUM_DIGITALS
#define S_ANALOG_START              1
#define S_DIGITAL_START             S_ANALOG_START + S_NUM_ANALOGS
#define S_LOOP                      1
#define S_NODE                      7
#define S_MODULE                    14
#define S_ANALOG_BLOCK              670
#define S_DIGITAL_BLOCK             184
#define S_MAX_NUM_READS             10

#if ICI_PC
extern unsigned_stklen = 30000U;
#endif

short  s_Logical_ici = 1;
```

```
/*****************************************************************************/
/*      FUNCTION:               main ( )                                     */
/*                                                                           */
/*      DESCRIPTION:            This program will connect to and restart that logical ICI, establish four analog */
/*                              import points, establish four digital import points, connect them, and issue a read */
/*                              data list S_MAX_NUM_READS times to obtain the values and statuses of the */
/*                              import points. If the program fails in any of these operations, an error will be */
/*                              reported to the user and the program will terminate.  */
/*      PARAMETERS:             NONE                                         */
/*                                                                           */
/*      RETURNS:                NONE                                         */
/*                                                                           */
/*****************************************************************************/
#if (ICI_PC)
main( )
#endif
#if (ICI_HP)
main (int i_argc, char  **cp_argv, char  **cp_arge)
#endif
#if (ICI_VAX)
sample1 ( )
main_program
#endif
{
        char                    c_line [80]
        ICI_CONNECT_PARAM       st_connect_param ;
        ERR_STRUCT              st_error ;
        short                   s_stat ;
        short                   s_loop_count ;

#if ICI_HP
        s_nc_hp_init_args (i_argc, cp_argv, cp_arge);
#endif
#if ICI_PC
        s_Logical_ici = 0;
#else
        printf ("Logical ICI: ");
        gets (c_line);
        s_Logical_ici = atoi (c_line);
#endif


        /* clear the error structure */
        memset(&st_error, 0, sizeof(ERR_STRUCT));

        /* connect to the Server */
        st_connect_param.c_type_connection = ICI_EXCLUSIVE ;
        st_connect_param.st_time_out.uc_units = ICI_HRS ;
        st_connect_param.c_return_tagnames = NO ;
        st_connect_param.st_time_out.s_number = 1 ;
        s_stat = s_connect_to_ici (s_Logical_ici, &st_connect_param, &st_error) ;

        /* if connect failed (warning or fatal) print the error message */
        if (s_stat != ICI_OK)
                {
                v_print_error_struct(s_stat, &st_error) ;

                /* exit on fatal errors, continue on warnings */
                if (s_stat == ICI_FATAL)
                        return ;
```

```
        }
    /* call function to restart a particular ICI */
    s_stat = s_setup_restart_call (s_Logical_ici) ;

    /* if restart failed disconnect and exit, on warnings continue */
    if (s_stat == ICI_FATAL)
        {
        s_stat = s_disconnect_from_ici (s_Logical_ici, &st_error) ;
        return ;
        }

    /* loop through and establish analog import points in the ICI */
    for (s_loop_count = 0 ; s_loop_count < S_NUM_ANALOGS ; s_loop_count++)
        {
        /* call function to establish an import point in the ICI */
        s_stat = s_setup_estab_import_pt_call (s_Logical_ici,
                                    s_loop_count + S_ANALOG_START,
                                    PT_ANALOG, S_LOOP, S_NODE, S_MODULE,
                                    S_ANALOG_BLOCK + s_loop_count) ;

        /* if establish failed disconnect and exit */
        if (s_stat == ICI_FATAL)
                                    {
                                    s_stat = s_disconnect_from_ici (s_Logical_ici, &st_error) ;
                                    return ;

                                    }

        }

    /* loop through and establish digital import points in the ICI */
    for (s_loop_count = 0 ; s_loop_count < S_NUM_DIGITALS ; s_loop_count++)
        {

    /* call function to establish an import point in the ICI */
    s_stat = s_setup_estab_import_pt_call (s_Logical_ici,
                                    s_loop_count + S_DIGITAL_START,
                                    PT_DIGITAL, S_LOOP, S_NODE, S_MODULE,
                                    S_DIGITAL_BLOCK + s_loop_count) ;

    /* if establish failed disconnect and exit  */
    if (s_stat == ICI_FATAL)
        {
        s_stat = s_disconnect_from_ici (s_Logical_ici, %st_error) ;
        return;
        }
    }
    /* wait a second to allow exception reports to to the ICI */
    sleep (5) ;

    s_loop_count = 1 ;

    while (s_loop_count <= S_MAX_NUM_READS)
        {
        /* call function to read data list */
        s_stat = s_setup_read_data_list_call (s_Logical_ici, S_ANALOG_START,
                    (S_ANALOG_START + S_NUM_ANALOGS + S_NUM_DIGITALS - 1)) ;
        sleep (1) ;
        s_loop_count++ ;
        } ;
    /* disconnect from ICI */
    s_stat = s_disconnect_from_ici (s_Logical_ici, &st_error);

    return;

}
```

```
/*****************************************************************************/
/*    FUNCTION:          void v_print_error_struct ( )                       */
/*                                                                           */
/*    DESCRIPTION:       This routine will call the semAPI function that parses the error structure and will print */
/*                       error messages (text) for each error that it encounters in the error structure. */
/*                                                                           */
/*    PARAMETERS:  short s_stat          The original return value from one of the semAPI calls. This */
/*                                       function prints the status along with the error messages. */
/*                                                                           */
/*          ERR_STRUCT *stp *stp_error   A pointer to the error structure that was passed to the semAPI */
/*                                       routine.                            */
/*                                                                           */
/*                                                                           */
/*                                                                           */
/*    RETURNS:           NONE                                               */
/*                                                                           */
/*****************************************************************************/
void v_print_error_struct (short s_stat, ERR_STRUCT *stp_error)
{
        short s_more_errors = ICI_CONTINUE ;
        short s_next=0;
        char c_message [132] ;

        printf ("Status = %d\n", s_stat) ;
        while (s_more_errors == ICI_CONTINUE)  {
                s_more_errors = s_ici_error_text (stp_error, &s_next, &c_message [0] ) ;
                printf ("%s\n" , c_message) ;

        }
}
```

```
/***************************************************************************/
/*   FUNCTION:          short s_print_data ( )                              */
/*                                                                          */
/*   DESCRIPTION:       This routine will parses the status table  structure for PT_ANALOG and PT_DIGITAL   */
/*                      type points and print the value and status of the point.   */
/*                                                                          */
/*   PARAMETERS:        RD_DATA_GROUP_DATA *stp_read_data_lis_data          */
/*                          Pointer to the user data area for a read data list semAPI call.   */
/*                                                                          */
/*                      RD_DATA_GROUP_ELEMENT *stp_read_data_lis_ele        */
/*                          Pointer to the read data list elements returned in the user data area of the read   */
/*                          data list semAPI call.                          */
/*                                                                          */
/*   RETURNS:           NONE                                                */
/***************************************************************************/
short s_print_data(
            RD_DATA_GROUP_DATA *stp_read_data_lis_data,
            RD_DATA_GROUP_ELEMENT *stp_read_data_lis_ele)
      {
      short                     s_stat = ICI_OK ;
      short                     s_count, i ;
      short                     s_quality ;
      VALUE_TABLE               *pt_value ;
      STATUS_TABLE              *pt_status ;

      /* determine how many values were returned */
      s_count = stp_read_data_lis_data ->s_count ;

      /* loop through and print the values */
      for (i=0; i < s_count; i++)
               {
            printf ("  Index: %d     ", stp_read_data_lis_ele[i].s_index) ;
            printf ("  Point Type: %d     ", stp_read_data_lis_ele[i].uc_point_type) ;

            /* assign pointer to VALUE TABLE structure */
            pt_value = &(stp_read_data_lis_ele [i] .st_data) ;

            /* NOTE: This application only uses PT_ANALOG and PT_DIGITAL */
            if (pt_value -> s_point_type == VALUE_ANALOG_POINTS)
                              {
                              printf ( " Value: %f ", pt_value->unpoints.st_analog.f_value) ;
                              /* assign pointer to STATUS TABLE structure */
                              pt_status = &(pt_value->un_points.st_analog.st_status) ;
                              /* process the quality */
                              s_quality = pt_status->un_status.st_analog.q ;
                              if (s_quality ==1)
                              printf ("Quality: %d %s\n" , s_quality, "BAD") ;
                              else
                              printf("Quality: %d %s\n" , s_quality, "GOOD") ;
                              }
            else
                              {
                              /* assign pointer to STATUS TABLE structure */
                              pt_status = &(pt_value->un_points.st_digital) ;
                              printf ("  Value: %d  ", pt_status->un_status.st_digital.v) ;
                              /* process the quality */
                              s_quality = pt_status->un_status.st_digital.q ;
                              if (s_quality == 1)
                              printf ("Quality: %d %s\n" , s_quality, "BAD") ;
                              else
                              printf("Quality: %d %s\n" , s_quality, "GOOD") ;
                              }
            }
            printf ("\n") ;
            return (s_status) ;  }
```

```
/*****************************************************************************/
/*    FUNCTION:          short s_setup_read_data_list_call( )                */
/*                                                                           */
/*    DESCRIPTION:       This routine will call the semAPI function that reads the values of a list of import points */
/*                       in the ICI.  This function is written specifically to handle  PT_ANALOG and PT_DIGITAL */
/*                       points only.                                        */
/*                                                                           */
/*    PARAMETERS:        short s_log_ici  - the Logical ICI to read values from.  */
/*                       short s_start_index - the starting index to read the value for. */
/*                       short s_stop_index - the ending index to read the value for. */
/*                                                                           */
/*    RETURNS:           Status of the read value list command               */
/*                                                                           */
/*****************************************************************************/

short s_setup_read_data_list_call(
        short s_log_ici,
        short s_start_index
        short s_stop_index)
{
        RD_DATA_GROUP_DATA      st_read_data_lis_data;
        RD_DATA_GROUP_ELEMENTst_read_data_lis_ele[S_NUM_TOTAL_POINTS];
        ERR_STRUCT              st_error;
        GENERAL_LIST_ARGS_PARAMst_list_args;
        short                   s_stat = ICI_OK;

        /* set up the read data list parameters */
        st_read_data_lis_data.s_count = MAX_READ_DATA_GROUP_COUNT;
        st_read_data_lis_data.stp_elements = &st_read_data_lis_ele[0];

        /* fill in the starting and ending index to read */
        st_list_args.s_start = s_start_index;
        st_list_args.s_stop = s_stop_index;

        /* clear the error structure */
        memset(&st_error, 0, sizeof(ERR_STRUCT));

        /* issue the read data list command */
        s_stat = s_read_data_list_w(s_log_ici, &st_error,
                                    &st_read_data_lis_data,
                                    sizeof(RD_DATA_GROUP_DATA),
                                    &st_list_args);

        /* error reading ICI */
        if (s_stat != ICI_OK)
                v_print_error_struct(s_stat, &st_error);
        else
                s_print_data (&st_read_data_lis_data,
                                    &st_read_data_lis_ele[0]);

        /* return the status of the read command */
        return(s_stat);
}
```

```
/ ************************************************************************************ /
/*    FUNCTION:          short s_setup_establish_import_pt_call( )                     */
/*                                                                                     */
/*    DESCRIPTION:       This routine will set up all the parameters necessary to call the semAPI function that */
/*                       establishes a point in the ICI.                               */
/*                                                                                     */
/*                                                                                     */
/*    PARAMETERS:        short s_log_ici - The logical ICI to establish the point in.  */
/*                       short s_point_index - The point index to use for the established point. */
/*                       short s_point_type - The point type to establish the point as. */
/*                       short s_loop - The INFI 90 loop number of the point.          */
/*                       short s_node -The INFI 90 node (PCU) number of the point.     */
/*                       short s_module -The INFI 90 module number of the point.       */
/*                       short s_block -The INFI 90 block number of the point.         */
/*                                                                                     */
/*    RETURNS:           Status of the establish point command                         */
/*                                                                                     */
/ ************************************************************************************ /
short s_setup_estab_import_pt_call(
        short s_log_ici, short s_point_index,
        short s_point_type, short s_loop,
        short s_node, short s_module,
        short s_block)
{
        ESTAB_IMPORT_PARMA      st_estab_import_param ;
        INFI90ADR               st_infi90_param ;
        ERR_STRUCT              st_error ;
        WORK_FLAG               st_work_data ;
        short                   s_stat = ICI_OK ;

        /* set up the establish point parameters */
        st_estab_import_param.stp_infi90_address = &st_infi90_param ;
        st_estab_import_param.s_pt_index = s_point_index ;
        st_estab_import_param.c_pt_type =  (char)s_point_type ;
        st_estab_import_param.c_connect_pt = CONNECT_PT ;
        st_estab_import_param.c_auto_disconnect = POINT_NOT_DISCONNECTED;

        /* set up the INFI 90 address parameters */
        st_infi90_param.s_loop =      s_loop ;
        st_infi90_param.s_node =      s_node ;
        st_infi90_param.s_module =    s_module ;
        st_infi90_param.s_block =     s_block ;

        /* clear the error structure */
        memset(&st_error, 0, sizeof(ERR_STRUCT)) ;

        /* issue the establish point command */
        s_stat = s_establish_import_point_w(s_log_ici, &st_estab_import_param, &st_error,
                                        &st_work_data, sizeof(WORK_FLAG)) ;

        /* if there was an error print INFI 90 address */
        if (s_stat != ICI_OK) {
                printf("  Infi90 Index:    %d\n", s_point_index);
                printf("Infi90 Address:  L:%d, P:  %d, M:  %d, B:  %d\n",
                                        st_infi90_param.s_loop, st_infi90_param.s_node,
                                        st_infi90_param.s_module, st_infi90_param.s_block);
                v_print_error_struct(s_stat, &st_error);
        }

        /* return the status of the establish point command */
        return(s_stat) ;

}
```

```
/*****************************************************************************/
/*    FUNCTION:          short s_setup_restart_call( )                       */
/*                                                                           */
/*    DESCRIPTION:       This routine will call the semAPI function that restarts the logical ICI.  */
/*                                                                           */
/*                                                                           */
/*                                                                           */
/*    PARAMETERS:        short s_log_ici  -  The logical ICI to restart.     */
/*                                                                           */
/*                                                                           */
/*    RETURNS:           Status of the restart ICI command.                  */
/*                                                                           */
/*****************************************************************************/
short s_setup_restart_call(
        short s_log_ici)
{
        ERR_STRUCT              st_error ;
        RESTART_PARAM           st_restart_param ;
        RESTART_DATA            st_restart_data ;
        short s_stat = ICI_OK ;

        /* set up the restart parameters */
        st_restart_param.c_watchdog_timer = (char) 0 ;
        st_restart_param.c_reply_delay = (char) 0;
        st_restart_param.c_time_synch = (char) ENABLE;
        st_restart_param.c_excpt_rpt_screen = (char) ENABLE;
        st_restart_param.c_primary = (char) ENABLE;
        st_restart_param.c_station_control = (char) ENABLE;
        st_restart_param.c_infinet_mode = (char) ENABLE;
        st_restart_param.c_work_flag = (char) DISABLE;
        st_restart_param.c_bad_qual_alm = (char) DISABLE;
        st_restart_param.c_time_stamp = (char) ENABLE;
        st_restart_param.c_wall_clock = (char) DISABLE;

        /* clear the error structure */
        memset(&st_error, 0, sizeof(ERR_STRUCT));

        /* issue the restart command */
        s_stat = s_ici_restart_w(s_log_ici, &st_error, &st_restart_param,
                                 &st_restart_data, sizeof(RESTART_DATA));

        /* error restarting the ICI */
        if (s_stat != ICI_OK)
                v_print_error_struct(s_stat, &st_error);

        /* return the status of the restart command */
        return(s_stat);
}
```

SECTION 9 - TROUBLESHOOTING

INTRODUCTION

This section provides troubleshooting information for the Strategic Enterprise Management Application Programming Interface (semAPI) software.

- Table 9-1 lists function error codes. The error codes are reported from the function layer and are loaded into the ***sub*** member of the error structure.
- Table 9-2 lists message driver error codes. The error codes are reported from the message driver layer and are loaded into the ***md*** member of the error structure.
- Table 9-3 lists device driver error codes. The error codes are reported from the device driver layer and are loaded into the ***dd*** member of the error structure.
- Table 9-4 lists computer interface error codes. The error codes are loaded into the ***ici*** member of the error structure.
- Table 9-5 lists general error codes. The error codes are reported from any layer as a general error and are loaded into the ***gen*** member of the error structure.

The following describes the table format:

The ***Value*** column lists the number of the error code that is returned to the error structure, the ***Reply Code*** column lists the parameter associated with the error code and the ***Meaning/Possible Corrective Action*** describes the error code. The function **ICI ERROR TEXT** can be used to return an English text string for an error value.

The following describes the possible error structures:

```
struct ici_err
  {
  short             s_fatal;          /* ICI fatal error          */
  short             s_warn;           /* ICI warning              */
  };
struct dd_err
  {
  short             s_fatal;          /* DD fatal error           */
  short             s_warn;           /* DD warning               */
  };
struct md_err
  {
  short             s_fatal;          /* message  fatal error     */
  short             s_warn;           /* message warning          */
  };
struct sub_err
  {
  short             s_fatal;          /* subroutine fatal         */
  short             s_warn;           /* subroutine warning       */
  short             s_stat;           /* extra status information */
  };
struct gen_err
  {
  short             s_fatal;          /* message  fatal error     */
  short             s_warn;           /* message warning          */
  };
typedef struct error
  {
  short             s_level;          /* overall error level      */
  short             s_err_layer;      /*  layer failure flags     */
  struct ici_err    ici;              /*  ici module errors       */
  struct dd_err     dd;               /*  device driver errors    */
  struct md_err     md;               /*  message driver errors   */
  struct sub_err    subs;             /*  ici subroutine errors   */
  struct gen_err    gen;              /*   general errors         */
  } ERR_STRUCT;
```

ERROR CODES

Table 9-1. Sub Level Error Codes

Value	Reply Code	Meaning/Possible Corrective Action
1	SUB_INVAL_LOG_ICI	Invalid logical computer interface unit. This indicates that the application is not connected to the computer interface unit or the inactivity timer has expired and the application has been disconnected from the computer interface unit.
2	SUB_INVAL_INDEX	Invalid index.
3	SUB_INVAL_NUM_PTS	Invalid number of points.
4	SUB_INVAL_NODE_TYPE	Invalid node type.
5	SUB_INVAL_NUM_NODES	Invalid number of nodes.

Table 9-1. Sub Level Error Codes (continued)

Value	Reply Code	Meaning/Possible Corrective Action
6	SUB_INVAL_INDEX_ORDER	Invalid index order.
7	SUB_INVAL_AUTO_DISCON	Invalid auto disconnect.
8	SUB_INVAL_INFI90_ADDR	Invalid INFI 90 OPEN address.
9	SUB_INVAL_POINT_TYPE	Invalid point type.
10	SUB_INVAL_NUM_EXCPTS	Invalid number of exceptions.
11	SUB_NULL_POINTER	Null data pointer.
12	SUB_INVAL_ICI_TYPE	Invalid computer interface type.
13	SUB_INVAL_REPLY_SIZE	Invalid reply size.
14	SUB_NULL_ST_BUFFER	st_buffer is set to null.
15	SUB_INVAL_REPLY_CODE	Bad reply code from computer interface.
16	SUB_INVAL_ARGS	General invalid arguments.
17	SUB_INVAL_NUM_GROUP	Too many elements for group.
18	SUB_ALLOC_ERROR	Unable to allocate memory.
19	SUB_INVAL_NUM_LIST	Too many elements for list.
20	SUB_ALREADY_CONN	Already connected to computer interface.
21	SUB_CONN_FAIL	Connect to computer interface failed.
22	SUB_USERDATA_TOOSMALL	User data area too small.
23	SUB_RESTART_LOCK	Unable to lock computer interface for restart.
24	SUB_CONF_READ	Unable to read configuration.
25	SUB_CONNECT_W	Error establishing connect to computer interface.
26	SUB_CONNECT_Q	Error establishing connect to computer interface.
27	SUB_CON_INIT_W	Error initializing connection to computer interface.
28	SUB_CON_INIT_Q	Error initializing connection to computer interface.
29	SUB_DISCONN_W	Error disconnecting from computer interface.
30	SUB_DISCONN_Q	Error disconnecting from computer interface.
31	SUB_DISCONN_DD	Error disconnecting from server.
32	SUB_ENV_FAIL	Error obtaining environmental information.
33	SUB_BUFF_BOUNDRY_EXCEDED	Buffer length insufficient for reply.
34	SUB_EST_MESS	Error establishing message system.
35	SUB_RETURN_RSTRT	Error returning restart lock.
36	SUB_RETURN_ONOFF	Error returning on/off-line lock.
37	SUB_ONOFF_LOCK	Unable to lock computer interface for on/off line.
38	SUB_RETURN_LOCK	Error returning locks.
39	SUB_SET_CONF	Error defining configuration (R) to server.
40	SUB_SET_INDEX	Error defining configuration (I) to server.
41	SUB_FAIL_ON	Error failing over to backup computer interface.
42	SUB_RESTART	Error on automatic restart of computer interface.
43	SUB_GET_ONOFF_L	Unable to lock interface for on/off-line command (p).
44	SUB_GET_RESTART_L	Unable to lock computer interface for restart command.
45	SUB_GET_MANAGER_L	Unable to lock computer interface for manager command.
52	SUB_UPDATE_USERA	Error updating user list (add).

Table 9-1. Sub Level Error Codes (continued)

Value	Reply Code	Meaning/Possible Corrective Action
53	SUB_UPDATE_USERD	Error updating user list (delete).
54	SUB_MORE_EXCEPTIONS	More exceptions returned that did not fit in the user data space.
56	SUB_WAIT_WATCHDOG	Waiting on watchdog time to time-out.
57	SUB_INIT_ERROR	Error initializing sub layer in connect.
58	SUB_TAGNAME_UNDEFINED	Tag name is not defined in the database.
59	SUB_CHKLIST	Errors exist in some but not all of the tags. Check the tag status code array to determine the reason for failure.
62	SUB_INVAL_NUM_TAGS	An invalid number of tags was specified in the request.
63	SUB_INVAL_NUM_EXPORTS	An invalid number of exports was specified in the request.
64	SUB_INVAL_OPTION	An invalid option was specified when defining or undefining export points.
66	SUB_NO_MANAGEMENT	Client not manager for requested type.
67	SUB_UNKN_PRB_REPORT	Unknown type of problem report returned for module.

Table 9-2. Message Driver Error Codes

Value	Reply Code	Meaning/Possible Corrective Action
1	MD_NULL_RESPONSE	Null pointer passed to decode by message driver.
2	MD_INIT_ERROR	Message driver communication failed to initialize.
3	MD_GET_TOKEN	Unable to get the requested token.
4	NULL_DECODE	Null decode function received.
5	INBUF_ERR	Error checking in buffer.
6	ILLEGAL_RETRIEVE	Illegal retrieve command.
7	NO_MSG_ERR	No message on the pending list.
8	CHECK_MSG_ERR	Error checking message.
9	LIST_ERR	Error putting message on list.
10	PARSE_ERR	Error parsing information flag.
11	GET_NODE_ERR	Error getting node off list.
12	NULL_BUF_ERR	Null buffer structure received.
13	NO_RESPONSE	No response from the device driver.
14	NO_MSG_NUM	No message number returned.
15	HEADER_ERR	Encoding header error.
16	PUT_TOK_ERR	Error returning token.
17	NO_TOKEN	Unable to get token for message.
18	ACK_FAILURE	Message did not receive and acknowledge.
19	ACK_RETRY	Message had retry on acknowledge list.
20	SEND_FAILED	Unable to send message to network connect.
21	MAKE_ACK_ERR	Error putting message on acknowledge list.
22	MALLOC_ERR	Error allocating memory for the message.
23	CONNECT_ERR	Error connecting to computer interface.
25	MD_NC_ALRDY_CONCTED	Network connect already connected.

Table 9-2. Message Driver Error Codes (continued)

Value	Reply Code	Meaning/Possible Corrective Action
26	MD_NC_INV_CALL	Invalid network connect function call.
27	MD_NC_REV_MISMATCH	Network connect software revision mismatch.
28	MD_NC_NO_LINK_AVAIL	Network connect no link (maximum clients connected).
29	MD_MSG_NOT_FOUND	Error message not found.
30	MD_FREE_LIST	Error putting onto free list.
31	MD_DD_INFO_ERR	Error setting server information.
32	MD_ALLOC_ERR	Error getting buffer from message.
33	MD_INV_BYTE_CNT	Invalid byte count in receiving message.
34	MD_MSG_CHK_ERR	Error checking message.
35	MD_WATCHDOG_ERR	Error scheduling watchdog timer.
36	MD DOWNLOAD_ERR	Error downloading computer interface point table.
37	MD_RESTART_ERR	Error restarting the computer interface.
38	MD_ONOFFLINE_ERR	Error putting computer interface on or off-line.
39	MD_STOP_DD_ERR	Error stopping the device driver.
40	MD_COPY_INEX_ERR	Error copying the index file.
41	RESTORE_INDEX_ERR	Error restoring indices to the device driver.
42	MD_DEL_KEY_ERR	Error deleting key.
43	MD_PRODID_INVALID	Product ID in software key is invalid.
44	MD_NUMUSERS_INVALID	Number of users in software key is invalid.
45	MD_UNSUPPORTED_MODULE	The INFI 90 OPEN communication module is not supported by the semAPI software.
46	ICK_CMD_ERR	Error sending ICK command to device driver.
47	WAIT_ON_FIRST_CLIENT	First client initializing the device driver.
48	MD_SET_PRIORITY_ERR	Error setting client priority.
49	MD_CANCEL_KEY_ERR	Error canceling a quick keyed message.

Table 9-3. Device Driver Error Codes

Value	Reply Code	Meaning/Possible Corrective Action
1	DD_OWNED_EXCL	Another user owns computer interface as exclusive.
2	DD_ALREADY_SHARED	Computer interface shared, exclusive not allowed.
3	DD_UNKNOWN_CONNECTION	Unknown connection type requested.
4	NOT_CONNECTED	User quick connect without preceding connect.
5	MSG_FORMAT_ERR	Invalid Information flag in transaction.
6	INVAL_MSG_TYPE	Invalid service message code. This may indicate that the target module does not support the semAPI command.
10	DD_HEADER_ERR	Error putting message driver header on.
11	FILL_TRANS_ERR	Error filling transaction structure.
13	CHECK_REPLY	Error communicating to computer interface.
14	CHECK_SUM_ERR	Check sum did not match reply.
15	NO_RESP_BYTES	No response bytes received.
16	EXCLUSIVE_EXISTS	Exclusive user already exists.

Table 9-3. Device Driver Error Codes (continued)

Value	Reply Code	Meaning/Possible Corrective Action
17	INVAL_TYPE_CONNECT	Invalid type connection.
18	TOKEN_NOT_AVAIL	Token not available.
20	ICI_SEND	Error sending to computer interface.
21	DD_NO_LINK_AVAIL	Service connect no link (maximum clients connected).
31	INIT_ICI_NC	Error initializing computer interface network connect.
32	INIT_MD_NC	Error initializing message driver and network connect.
33	READ_COM_FILE	Reading communication file.
34	INIT_DD	Initializing device driver.
35	GOING_TO_SLEEP	Going to sleep.
36	CHECKING_ACTIVE	Checking the active list.
37	CHECKING_RESP	Checking the response list.
38	ROUTING_MSG	Routing message.
39	NO_NODES	No nodes left on list.
40	NO_ST_TRANS	No transaction structure.
41	FREE_LIST	Error putting onto free list.
42	SENDING_REPLY	Error sending reply.
43	QRCV_ERR	Error doing quick receive.
44	TOK_NOT_RETURNED	Token not returned.
45	TOK_OWN_NOT_AVAIL	Token owners not available.
46	DISCONNECT_ERR	User disconnect error.
47	GET_PERFORM_ERR	Get performance data error.
48	QCONNECT_ERR	Quick connect error.
49	SEND_INDEX_ERR	Send index error.
50	GET_INDEX_ERR	Get index list error.
51	GET_INDEX_LIS_ERR	Get index error.
52	CLEAR_INDEX_ERR	Clear index error.
53	ILL_TYP_CON	Invalid connection (exclusive or shared).
54	NO_CONNECTIONS	No wait connection found for quick.
55	RESP_TOO_BIG	Unsolicited response too large for buffer.
56	RESP_LIST_ERR	Cannot put unsolicited response on list.
57	INIT_USER_ERR	Cannot initialize a new user.
58	CALLOC_USER_ERR	Cannot allocate memory for a new user.
59	ACTIVE_TRANS_ERR	Error getting active transaction.
60	GET_USER_ERR	Error getting user information.
61	DD_ILL_INDEX	Illegal index received.
62	DD_REVISION_MATCH	Client/server software mismatch.
63	DELETE_KEY_ERR	Error deleting key message.
64	DD_NO_KEYS	Error all keys are in use.
65	DD_SEC_DEV_CHANGED	The ICK device (software key) changed during normal operations.
66	DD_SEC_CHK_FAIL	The security check command failed to the ICI module.

Table 9-3. Device Driver Error Codes (continued)

Value	Reply Code	Meaning/Possible Corrective Action
67	DD_KEY _CLEARED	The command was cleared because of a restart. Issue the command again.
68	DD_CHECK_INBUF	Invalid message received from a client.
69	DD_PRIORITY_IN_USE	Priority channel already in use.
70	ICI_DECODE	ICI reply was not properly encoded.
71	ICI_COMM_FAIL	Command lost because of communications failure.

Table 9-4. ICI Error Codes

Value	Reply Code	Meaning/Possible Corrective Action
0	GMI_STATUS_OK ***- or -*** ICI_OK	GMI successful. ICI Successful.
1	ICI_WAIT_LOOP	Command queue to computer interface, waiting for reply.
2	ICI_INV_FORMAT	Improper format for command.
3	ICI_ILL_COMMAND	Illegal command issued.
4	ICI_IND_ALL_EST	Index already established in computer interface.
5	ICI_BLK_ALL_EST	Block already established as another point.
6	ICI_CMD_TO_LONG	Command is too long.
7	ICI_BD_NODE_REP	Bad reply from node interface.
8	ICI_EXP_AS_IMP	Export used as import.
9	ICI_RESTART_REP	Second **RESTART** needed.
10	ICI_UND_INDEX	Undefined index.
11	ICI_MEM_FULL	Memory full.
12	ICI_HOST_COM	Host communication error.
13	ICI_IN_MOD_NOT_REP	Computer interface internal module not responding.
14	ICI_IMP_AS_EXP	Import used as export.
15	ICI_TIMOUT_PL	Time-out of Plant Loop response.
16	ICI_NUM_RANGE	Number out of range.
17	ICI_ILL_KEY	Illegal key used.
18	ICI_NEED_RESTART	Computer interface requires **RESTART**.
19	ICI_MOD_STA_AS_IMP	Module status point used as import.
20	ICI_WAIT_REPLY	Message is active on loop.
21	ICI_INV_MOD_STAT	Import or export used as a module status.
22	ICI_EXC_SPEC_LOST	Exception report specifications.
23	ICI_NOTH_QUEUED	No message queued.
24	ICI_REP_TOO_LARGE	Reply to large.
25	ICI_ILL_STA_MOD	Illegal station mode command.
26	ICI_ILL_MOD_NUM	Illegal module number in command.
27	ICI_TIM_CMD	Time-out between bytes in command.
28	ICI_IND_EST_HOST	Index already established.
29	ICI_PT_TYP_INCM	Point type incompatible.

Table 9-4. ICI Error Codes (continued)

Value	Reply Code	Meaning/Possible Corrective Action
30	ICI_WATCH_TIME	Watchdog time-out.
31	ICI_CHECKSUM_ERR	Checksum compare error.
32	ICI_DEST_NODE_OFF	Destination node is off-line.
33	ICI_CALLUP_REQ	**CALLUP** command is required.
34	ICI_COMP_ERR	Computer interface internal error. Usually occurs when using an unsupported point type.
35	ICI_COMP_BUSY	Computer interface is busy.
36	ICI_IS_OFFLINE	Computer interface has gone off-line.
37	ICI_CON_MON_MOD	Conflict with monitor mode.
38	ICI_POINT_TYPE	Point type does not match computer interface point type.
39	ICI_DEST_LOOP_OFF	destination loop is off-line.
40	ICI_DEST_NOD_BUSY	Destination node is busy.
41	ICI_DEST_LOOP_BUSY	Destination loop is busy.
42	ICI_ENH_TRD_EST	Enhanced trend point is not established.
43	ICI_UDXR_NOT_ESTAB	User defined exception report point was not established.
44	ICI_INV_WALLCLOCK	Wallclock time is not valid. Error returned when the ICI is restarted with timestamps enabled and adding the wallclock offset enabled but the ICI interface has not received a time synch message from either the host computer or the INFI 90 OPEN system.
69	ICI_DECODE	ICI reply was not properly encoded.
70	ICI_COMM_FAIL	Command lost due to communications failure.
100	ICI_UND_MESS_TYPW	Undefined message type.
101	ICI_BUSY	Module is busy, cannot reply.
102	ICI_MODE_CONFLICT	Module mode conflicts with command.
103	ICI_ILL_DATA	Illegal message data.
104	ICI_INV_BLK_NUM	Function block is not valid.
105	ICI_UND_BLK_NUM	Function block is not configured.
106	ICI_BLK_NOT_READ	Function block has no readable parameters.
107	ICI_INV_FUNC_CODE	Invalid function code specified for module.
108	ICI_FUNC_BLK_MISS	Function code and block number not compatible.
109	ICI_INS_MEM	Insufficient memory in module to write block.
110	ICI_MOD_NOT_RESP	Module is not responding.
128	ICI_WAIT_MOD_REP	Waiting for module reply.
200	NO_SEC_DEV_PRESENT	Software key does not exist on the termination unit.
201	RETRY_COMMAND	Retry the semAPI command.
202	INVALID_API_DETECTED	INFI 90 OPEN communication module detected an invalid semAPI command attempting to access the INFI 90 OPEN system.
203	WAIT_AND_RETRY_COMMAND	Wait one second and try the semAPI command again.
204	INVAL_SEC_DEV_PRESENT	INFI 90 OPEN communication module has detected a software key but it is invalid.
301	ICI_GMISTATUS_BUSY	MFP or file is in use.

Table 9-4. ICI Error Codes (continued)

Value	Reply Code	Meaning/Possible Corrective Action
302	ICI_GMISTATUS_BUSY	No buffers available.
303	ICI_GMISTATUS_SMALL	Buffers are too small.
304	ICI_GMISTATUS_NOT_OPENED	MFP file or buffer is not open.
305	ICI_GMISTATUS_WRITE_PROT	File is write protected.
306	ICI_GMISTATUS_OFFSET_OUT	Offset is out of range.
307	ICI_GMISTATUS_OPENED	File is already open.
308	ICI_GMISTATUS_INV	Invalid operation.
309	ICI_GMISTATUS_MODE	Wrong mode specified.
310	ICI_GMISTATUS_ERRORIN	File has an error in data.
311	ICI_GMISTATUS_EXIST	File does not exist.
313	ICI_GMISTATUS_UNABLE	Could not create file.

Table 9-5. General Error Codes

Value	Reply Code	Meaning/Possible Corrective Action
1	GEN_SET_DD_STATUS	Error sending status to the device driver.
2	GEN_RESTART_ERR	Error in restarting the ICI module.

APPENDIX A - WORK FLAG DESCRIPTION

INTRODUCTION

There are two ways to get a work flag from a computer interface:

- Restart the computer interface with the work flag option set. This causes the computer interface to pass back the work flag as part of the user data area in each semAPI function.

  NOTE: If the work flag option is not enabled by **RESTART** the work flag member of the user data area in each semAPI function will not have valid data.

- Issue **READ WORK FLAG.** The work flag will tell an application program what type of information the computer interface has available for the application. This indicates which semAPI function should be issued. The following explains the work flag structure. The work flag structure is defined in the ***ici_user.h*** header file.

WORK_FLAG

Table A-1 lists the work flag parameters and what to do when the parameter is returned.

NOTE: Ignore instances where work flag bits are set indicating commands need to be issued that do not exist in the function library.

Table A-1. Work Flag Parameters

Parameter	Description
WORK_FLAG	Structure that defines the work flag that is returned.
wf_rce	Indicates that the computer interface has command exceptions buffered internally. Issue **READ COMMAND EXCEPTIONS** to clear the internal buffers.
wf_rse	Indicates that the computer interface has station exceptions buffered internally. Issue **READ DATA EXCEPTIONS** to clear the internal buffers.
wf_rst	Indicates that the computer interface has been time synchronized by the INFI 90 OPEN system. Issue **READ SYSTEM TIME/DATE** to determine the new system time. This means only one time synch has been received by the computer interface since the last read of the work flag. See the wf_ats member also.
wf_rs	Indicates that the computer interface has received specification information about established points in the computer interface. Issue **READ DATA SPECS** to obtain the new specification information.
wf_re	Indicates that the computer interface has exceptions buffered internally. Issue **READ DATA EXCEPTIONS** to clear the internal buffers.
wf_rme	This member being set indicates that the computer interface has miscellaneous exceptions buffered internally. Issue **READ DATA EXCEPTIONS** to clear the internal buffers.

Table A-1. Work Flag Parameters (continued)

Parameter	Description
wf_rred	Indicates that the computer interface has received information about connecting or disconnecting routes to export points established in this computer interface. Issue **READ REPORT ENABLE/DISABLE** to determine which points are affected.
wf_ats	Indicates that the computer interface has been time synchronized by the INFI 90 OPEN system. Issue **READ SYSTEM TIME/DATE** to obtain the new time. This means at least two time synchs have been received by the computer interface since the last read of the work flag. Also see the wf_rst member.
wf_rte	Indicates that the computer interface has trend exceptions buffered internally. Issue **READ TREND EXCEPTIONS** to clear the internal buffers.
wf_rpm	This member being set indicates that the computer interface has plant messages buffered internally. Issue **READ PLANT MESSAGE** to clear the internal buffers.
wf_dq	Indicates that the computer interface has queued messages buffered internally. Issue **READ PLANT MESSAGE** to clear the internal buffers.
wf_rde	Indicates that the computer interface has exceptions buffered internally. Issue **READ DATA EXCEPTIONS** to clear the internal buffers.

APPENDIX B - POINT TYPE DESCRIPTIONS

POINT TYPE DESCRIPTION

Table B-1 lists and describes the point types defined in the ***ici_user.h*** header file.

Table B-1. Point Type Descriptions (ici_user.h)

Point Type	Import/ Export Point	Description
PT_ANALOG	Import	Analog real-3 read point of a station.
PT_ANALOG_REPORT	Export	An analog real-3 write point.
PT_ASCII_STRING	Import	A user defined (UDXR) read point.
PT_CONTROL_OUTPUT	Import	Control output write point of a station.
PT_CONTROL_POINT	Import	Control output read point of a station.
PT_DAANG	Import	A data acquisition analog (DAANG) read point.
PT_DADIG	Import	A data acquisition digital (DADIG) read point.
PT_DD	Import	A device driver (DD) read point.
PT_DIGITAL	Import	A digital read point.
PT_DIGITAL_REPORT	Export	A digital write point.
PT_ENHANCED_TREND	Import	An enhanced trend read point.
PT_EXTENDED_MODULE_STATUS	Import	An extended module status read point.
PT_MODULE_STATUS	Import	A module status read point.
PT_MSDD	Import	A multi-state device driver (MSDD) read point.
PT_PROCESS_VARIABLE	Import	Process variable read point of a station.
PT_RATIO_INDEX	Import	Ratio index read point of a station.
PT_RATIO_INDEX_WRITTEN	Import	Ratio index write point of a station.
PT_RCM	Import	A remote control memory (RCM) read point.
PT_RCM_REPORT	Export	A remote control memory (RCM) write point.
PT_REAL4_ANALOG_READ	Import	An analog real-4 read point.
PT_REAL4_ANALOG_REPORT	Export	An analog real-4 write point.
PT_REM_MOTOR_CONTROL	Import	A remote motor control (RMC) read point.
PT_RMSC	Import	A remote manual set constant (RMSC) read point.
PT_RMSC_REPORT	Export	A remote manual set constant (RMSC) write point.
PT_SET_POINT	Import	Set point/read point of a station.
PT_SET_POINT_OUTPUT	Import	Set point/write point of a station.
PT_STATION	Import	A single index station read point. This point type encapsulates the PT_PROCESS_VARIABLE, PT_SET_POINT, PT_CONTROL_POINT, PT_RATIO_INDEX, and PT_STATION_STATUS into one point type.
PT_STATION_MODE	Import	Status (mode) write point of a station.
PT_STATION_REPORT	Export	A single index station write point.

Table B-1. Point Type Descriptions (ici_user.h) (continued)

Point Type	Import/ Export Point	Description
PT_STATION_STATUS	Import	Status (mode) read point of a station.
PT_TEXT_SELECTOR	Import	A text selector read point.
PT_UNDEFINED	NA	An undefined point has this value until established as a point in the computer interface internal tables.

APPENDIX C - TIME STAMP DESCRIPTION

INTRODUCTION

The Strategic Enterprise Management Application Programming Interface (semAPI) functions return a time stamp from the INFI 90 OPEN system. Table C-1 defines and tells the system how many characters to reserve for the time string (TIME_LENGTH). The time structure is defined in the header file named ***ici_user.h***.

Table C-1. Format and Description of ICI_TIME_STAMP Structure

Parameter	Description
ICI_TIME_STAMP	Structure that defines the time stamp that is returned from the function.
s_hrs	The number of hours after midnight. Valid values are 0 through 23.
s_mins	The number of minutes after the hour. Valid values are 0 through 59.
s_secs	The number of seconds after the minute. Valid values are 0 through 59.
s_msecs	The number of milliseconds after the second. Valid values are 0 through 999.
s_year	The year. Valid values are 1980 through 2014.
s_month	The month number. Valid values are 1 through 12 (January = 1).
s_day	The day of the month. Valid values are 1 through 31.
c_time[ ]	The actual numbers formatted into a printable ASCII string representing the time. The string is TIME_LENGTH characters in length and has the following format: 13-DEC-1994 12:00:32.341

APPENDIX D - VALUE TABLE DESCRIPTION

INTRODUCTION

The Strategic Enterprise Management Application Programming Interface (semAPI) functions return point values in a structure called VALUE_TABLE. This structure is a union of value types and a key into the union. The key member s_point_type will indicate which member in the structure has valid data. **READ DATA EXCEPTIONS** and **READ DATA GROUP/ LIST** are functions that return a VALUE_TABLE structure. The value table structure is defined in the ***l3atable.h*** header file.

VALUE TABLE DESCRIPTION

The following is the actual definition of a VALUE_TABLE structure:

```
typedef struct value_tables {
  short s_point_type;

union {

struct value_bad_quality st_bad_quality;
struct value_analog st_analog;
STATUS_TABLE st_station_status;
STATUS_TABLE st_digital;
struct value_station_mode st_station_mode;
STATUS_TABLE st_module_status;
STATUS_TABLE st_rcm;
struct value_station_read st_station_read;
struct value_real4 st_real4
STATUS_TABLE st_ext_module_status;
struct value_enhanced_trend st_enhanced_trend;
struct value_daang st_daang;
struct value_udxr st_udxr;
STATUS_TABLE st_multistate;
STATUS_TABLE st_device_driver;
STATUS_TABLE st_remote_motor;
STATUS_TABLE st_dadig;
struct value_text_selector st_text_selector;

} un_points;


} VALUE_TABLE
```

Table D-1 lists the cross reference between the computer interface point types and the structure in the VALUE_TABLE union to access. It also lists the cross reference between the union key s_point_type and the structure in the VALUE_TABLE union access.

The un_points member of the value table is the union of various point value types. The s_point_type member indicates which structure in the union to access.

Table D-2 describes the st_bad_quality structure.

Table D-3 describes the structure of the analog point type.

Table D-4 describes the station status point structure.

Table D-5 describes the digital point structure.

Table D-6 describes the structure of the station mode point.

Table D-7 describes the st_module structure.

Table D-8 describes the st_rcm structure.

Table D-9 describes the station read point structure.

Table D-10 describes the structure of the analog real 4 points.

Table D-11 describes the st_ext_module structure.

Table D-12 describes the enhanced trend point structure.

Table D-13 describes the data acquisition analog point structure.

Table D-14 describes the user defined exception report point structure.

Table D-15 describes the st_multistate structure.

Table D-16 describes the st_device structure.

Table D-17 describes the st_remote_motor structure.

Table D-18 describes the st_dadig structure.

Table D-19 describes the st_text_selector structure.

Table D-1. s_point_type Value

Value	Structure to Access	Point Type
VALUE_ANALOG_POINTS	st_analog	PT_ANALOG PT_ANALOG_REPORT PT_CONTROL_OUTPUT PT_CONTROL_POINT PT_PROCESS_VARIABLE PT_RATIO_INDEX PT_RATIO_INDEX_WRITTEN PT_RMSC PT_RMSC_REPORT PT_SET_POINT PT_SET_POINT_OUTPUT
VALUE_BAD_QUALITY_POINTS	st_bad_quality	All
VALUE_DAANG_POINTS	st_daang	PT_DAANG
VALUE_DADIG_POINTS	st_dadig	PT_DADIG
VALUE_DEVICE_DRIVER_POINTS	st_device_driver	PT_DD
VALUE_DIGITAL_POINTS	st_digital	PT_DIGITAL PT_DIGITAL_REPORT
VALUE_ENHANCED_TREND_POINTS	st_enhanced_trend	PT_ENHANCED_TREND
VALUE_EXT_MODULE_STATUS_POINTS	st_ext_module_status	PT_EXTENDED_MODULE_STATUS
VALUE_MODULE_STATUS_POINTS	st_module_status	PT_MODULE_STATUS
VALUE_MULTISTATE_POINTS	st_multistate	PT_MSDD
VALUE_RCM_POINTS	st_rcm	PT_RCM PT_RCM_REPORT
VALUE_REAL4_POINTS	st_real4	PT_REAL4_ANALOG_READ PT_REAL4_ANALOG_REPORT
VALUE_REMOTE_MOTOR_POINTS	st_remote_motor	PT_REM_MOTOR_CONTROL
VALUE_STATION_MODE_POINTS	st_station_mode	PT_STATION_MODE
VALUE_STATION_READ_POINTS	st_station_read	PT_STATION PT_STATION_REPORT
VALUE_STATION_STATUS_POINTS	st_station_status	PT_STATION_STATUS
VALUE_TEXT_SELECTOR	st_text_selector	PT_TEXT_SELECTOR
VALUE_UDXR_POINTS	st_udxr	PT_ASCII_STRING

Table D-2. st_bad_quality Structure

Parameter	Description
value_bad_qualityst_bad_quality	Value table entry for bad quality points.
uc_loop_failed	Indicates whether the loop containing the point has failed. Used for bad quality alarm management. Valid values include:
NO_FAILURE_EXISTS	No failures were encountered on the point.
FAILURE_EXISTS	The node (PCU) containing the point has failed.
uc_node_failed	Indicates whether the node (PCU) containing the point has failed. Used for bad quality alarm management. Valid values include:
NO_FAILURE_EXISTS	No failures were encountered on the point.
FAILURE_EXISTS	The loop containing the point has failed.
uc_module_failed	Indicates whether the module containing the point has failed. Used for bad quality alarm management. Valid values include:
NO_FAILURE_EXISTS	No failures were encountered on the point.
FAILURE_EXISTS	The module containing the point has failed.

Table D-3. st_analog Structure

Parameter	Description
value_analogst_analog	Value table entry for analog points.
STATUS_TABLE st_status	Refer to Appendix E for an explanation of the STATUS_TABLE user data area.
f_value	Floating point value of the Real3 analog point.

Table D-4. st_station_status Structure

Parameter	Description
STATUS_TABLE st_station_status	Refer to Appendix E for an explanation of the STATUS_TABLE user data area.

Table D-5. st_digital Structure

Parameter	Description
STATUS_TABLEst_digital	Refer to Appendix E for an explanation of the STATUS_TABLE user data area.

Table D-6. st_station_mode Structures

Parameter	Description
value_station_mode st_station_mode	Value table entry for station mode points.
uc_station_mode	Indicates the current mode of the station.
COMPUTER_OK	The station has the computer OK signal from the host computer.
GOTO_COMPUTER_AUTO	The station is in the computer-auto mode.
GOTO_COMPUTER_BACKUP	The station is in the computer back-up state.
GOTO_COMPUTER_CASCADE	The station is in the computer cascade/ratio mode.
GOTO_COMPUTER_LEVEL	The station is at the computer level.
GOTO_COMPUTER_MANUAL	The station is in the computer-manual mode.

Table D-6. st_station_mode Structures (continued)

Parameter	Description
GOTO_LOCAL_AUTO	The station is in the local-auto (console/station-auto) mode.
GOTO_LOCAL_CASCADE	The station is in the local cascade/ratio (console/station-cascade/ ratio) mode.
GOTO_LOCAL_LEVEL	The station is at the local level (cascade/station level).
GOTO_LOCAL_MANUAL	The station is in the local-manual (console/station-manual) mode.
GOTO_PREVIOUS_STATE	The station has resumed the previous mode requested.

Table D-7. st_module_status Structure

Parameter	Description
STATUS_TABLE st_module_status	Refer to Appendix E for an explanation of the STATUS_TABLE user data area.

Table D-8. st_rcm Structure

Parameter	Description
STATUS_TABLE st_rcm	Refer to Appendix E for an explanation of the STATUS_TABLE user data area.

Table D-9. st_station_read Structure

Parameter	Description
value_station_read st_station_read	Value table entry for station read points.
STATUS_TABLE st_status	Refer to Appendix E for an explanation of the STATUS_TABLE user data area.
f_process_variable_value	The floating point value of the station process variable (PV).
f_set_point_value	Floating point value of the station set point (SP).
f_control_output_value	Floating point value of the station control output (CO).
f_ratio_index_value	Floating point value of the station ratio index (RI).

Table D-10. st_real4 Structure

Parameter	Description
value_real4 st_real4	Value table entry for real4 points.
STATUS_TABLE st_status	Refer to Appendix E for an explanation of the STATUS_TABLE user data area.
f_value	Floating point value of the Real-4 analog point.

Table D-11. st_ext_module_status Structure

Parameter	Description
ST_STATUS_TABLE st_ext_module_status	Refer to Appendix E for an explanation of the STATUS_TABLE user data area.

Table D-12. st_enhanced_trend Structure

Parameter	Description
value_enhanced_trend st_enhanced_trend	Value table entry for trend points.
mtx	Indicates whether the maximum reporting time has been exceeded.
NO	The maximum reporting time has not been exceeded.
YES	The maximum reporting time has been exceeded.
imm	Indicates whether the request for immediate notification is active.
NO	The request for immediate notification is not active.
YES	The request for immediate notification is active.
msg	Indicates whether the message size limitation has been exceeded.
NO	Message limitation has not been exceeded.
YES	Message limitation has been exceeded.
alm	Indicates whether the trend block is in alarm. Valid values include:
NO	Enhanced trend block is not in alarm.
YES	Enhanced trend block is in alarm.
q	Indicates the quality of the trend block.
GOOD_QUALITY	Enhanced trend block is in good quality.
BAD_QUALITY	Enhanced trend block is in bad quality.
num	This is the sequence number of trend data. This number is incremented for each trend exception report. Valid values include all numbers between 0 and 31.

Table D-13. st_daang Structure

Parameter	Description
value_daang st_daang	Value table entry for DAANG points.
q	Indicates the quality of the DAANG block.
ha	Indicates whether the DAANG block is in high alarm.
la	Indicates whether the DAANG block is in low alarm.
al	Indicates the alarm level of the DAANG block.
x	Indicates whether the extended status has changed on the DAANG block.
tag1	Indicates whether the DAANG block is red tagged.
am1	Indicates the mode of the DAANG block.
uc_byte2	Reserved for future use.
constant_1	Reserved for future use.
tag3	Indicates whether the DAANG block is red tagged.
fq3	Indicates whether a hardware fault or bad quality input signal has been detected.
or	Indicates whether the DAANG block is out of range.
lim	Indicates whether the DAANG block is limited.
am3	Indicates the mode of the DAANG block.
cal3	Indicates whether the DAANG block has a calculated value.
qo	Indicates whether the DAANG block has the quality overridden.

Table D-13. st_daang Structure (continued)

Parameter	Description
ss	Indicates whether the DAANG block is off scan (no reporting).
hda	Indicates whether the DAANG block is in high deviation alarm.
lda	Indicates whether the DAANG block is in low deviation alarm.
hr	Indicates whether the DAANG block has a high rate.
lr	Indicates whether the DAANG block has a low rate.
va	Indicates whether the DAANG block has variable alarms.
asi	Indicates whether the DAANG block has alarm suppression.
ra	Indicates whether the DAANG block is in a re-alarm situation.
pis	Indicates whether the DAANG block permits input select.
ce	Indicates whether the DAANG block has constraints enabled.
cal5	Indicates whether the DAANG block has a calculated value.
fq5	Indicates whether a hardware fault or bad quality input signal has been detected.
ma	Indicates whether the DAANG block has multilevel alarming.
am5	Indicates the mode of the DAANG block.
f_output_value	Floating point value of the DAANG block.
f_higher_limit	Floating point value of the high limit of the DAANG block.
f_lower_limit	Floating point value of the low limit of the DAANG block.

Table D-14. st_udxr Structure

Parameter	Description
value_udxr st_udxr	Value table entry for ASCII string points.
uc_class	The class of the data that is being returned. Valid data classes include:
CLASS_DIGITAL	A digital point data type.
CLASS_ANALOG	An analog point data type.
CLASS_STATUS	A status point data type.
CLASS_UNDEFINED	An undefined data class.
CLASS_ASCII	An ASCII string data type.
uc_format	Format of the data that is being returned. Valid data classes include:
uc_status_size	Number of bytes in the c_status array of status bytes. The maximum number of status bytes can be MAX_USER_STATUS_SIZE.
uc_data_size	Number of bytes in the c_data array of data bytes. The maximum data bytes can be MAX_USER_DATA_SIZE.
c_status[MAX_USER_STATUS_SIZE]	Array of status bytes. This array contains the number of status bytes as specified by the uc_status_size parameter.
c_data[MAX_USER_DATA_SIZE]	Array of data bytes. This array contains the number of data bytes as specified by the uc_data_size parameter.

Table D-15. st_multistate Structure

Parameter	Description
STATUS_TABLE st_multistate	Refer to Appendix E for an explanation of the STATUS_TABLE user data area.

Table D-16. st_device_driver Structure

Parameter	Description
STATUS_TABLE st_device_driver	Refer to Appendix E for an explanation of the STATUS_TABLE user data area.

Table D-17. st_remote_motor Structure

Parameter	Description
STATUS_TABLE st_remote_motor	Refer to Appendix E for an explanation of the STATUS_TABLE user data area.

Table D-18. st_dadig Structure

Parameter	Description
STATUS_TABLE st_dadig	Refer to Appendix E for an explanation of the STATUS_TABLE user data area.

Table D-19. st_text_selector Structure

Parameter	Description
value_text_selector st_text_selector	Value table entry for text selection points.
c_color_number	The color of displayed text.
TEXT_BLACK	Text displays in black.
TEXT_WHITE	Text displays in white.
TEXT_RED	Text displays in red.
TEXT_GREEN	Text displays in green.
TEXT_BLUE	Text displays in blue.
TEXT_CYAN	Text displays in cyan.
TEXT_MAGENTA	Text displays in magenta.
TEXT_YELLOW	Text displays in yellow
TEXT_ORANGE	Text displays in orange.
TEXT_YELLOW_GREEN	Text displays in yellow.
TEXT_GREEN_CYAN	Text displays in green-cyan.

Table D-19. st_text_selector Structure (continued)

Parameter	Description
TEXT_CYAN_BLUE	Text displays in cyan-blue.
TEXT_BLUE_MAGENTA	Text displays in blue-magenta.
TEXT_MAGENTA_RED	Text displays in magenta-red.
TEXT_DARK_GRAY	Text displays in dark-gray.
TEXT_LIGHT_GRAY	Text displays in light-gray.
c_blink	Indicates whether the text is blinking.
YES	Text is blinking.
NO	Text is not blinking.
l_message_number	Message number of the text to display.
STATUS_TABLE st_status	Refer to Appendix E for an explanation of the STATUS_TABLE user data area.

APPENDIX E - STATUS TABLE DESCRIPTION

INTRODUCTION

The Strategic Enterprise Management Application Programming Interface (semAPI) functions return point statuses in a structure called a STATUS_TABLE. This structure is a union of various status types as well as a key into the union. The status table structure is defined in the ***l3btable.h*** header file. The key member s_point_type indicates which member in the structure has valid data.

The STATUS_TABLE is also returned in the VALUE_TABLE structure (refer to Appendix D) as part of the value for several point types.

STATUS_TABLE

The following is the actual definition of a STATUS_TABLE structure:

```
typedef struct status_tables {
  short s_status_type;

union {

struct station_status st_station;
struct memory_status st_memory;
struct module_status st_module;
struct extended_status st_extended;
struct analog_status st_analog;
struct process_status st_process;
struct digital_status st_digital;
struct acquisition_status st_acquisition
struct pointtype6_status st_pointtype6;
struct single_index_status st_single_index;
struct multistate_status st_multistate;
struct device_driver_status st_device_driver;
struct remote_motor_status st_remote_motor;
struct dadig_status st_dadig;
struct transaction_status st_transaction;
struct text_selector_status st_text_selector;

} un_status;
 STATUS_TABLE
```

Table E-1 describes the STATUS_TABLE structure.

Table E-2 describes the structure for the station point type.

Table E-3 describes the structure for the remote control memory (RCM) point type.

Table E-4 describes the structure for module status type.

Table E-5 describes the structure for ICI module types.

Table E-6 describes the structure for all other types of modules.

Table E-7 describes the structure of module types that are unknown.

Table E-8 describes the base status of the extended module status point (PT_EXTENDED_MODULE_STATUS). The base status is the same as that for a regular module status point (PT_MODULE_STATUS). Refer to the st_module structure described in Table E-4.

Table E-9 describes the structure for analog points.

Table E-10 describes the structure for station point types.

Table E-11 describes digital point types.

Table E-12 describes the data acquisition analog (DAANG) point types.

Table E-13 describes the structure for the station status point types.

Table E-14 describes the structure for the single index point type.

Table E-15 describes the structure for the multstate device driver.

Table E-16 describes the structure for the device driver point type.

Table E-17 describes the remote motor control (RCM) point type.

Table E-18 describes the structure for the data acquisition digital (DADIG) point type.

Table E-19 describes the control transaction status.

Table E-20 describes the st_text_selector structure.

Table E-1. s_status_type Structure

Value	Structure to Access	Point Type
ANALOG_STATUS	st_analog	PT_ANALOG PT_ANALOG_REPORT PT_REAL4_ANALOG_READ PT_REAL4_ANALOG_REPORT
AQUISITION_STATUS	st_aquisition	PT_DAANG
DADIG_STATUS	st_dadig	PT_DADIG
DEVICE_DRIVER_STATUS	st_device_driver	PT_DD
DIGITAL_STATUS	st_digital	PT_DIGITAL PT_DIGITAL_REPORT
EXTENDED_STATUS	st_extended	PT_EXTENDED_MODULE_STATUS
MEMORY_STATUS	st_memory	PT_RCM PT_RCM_REPORT
MODULE_STATUS	st_module	PT_MODULE_STATUS
MULTISTATE_STATUS	st_multistate	PT_MSDD
POINTTYPE6_STATUS	st_pointtype6	PT_STATION_STATUS
PROCESS_STATUS	st_process	PT_CONTROL_POINT PT_PROCESS_VARIABLE PT_RATIO_INDEX PT_RMSC PT_RMSC_REPORT PT_SET_POINT
REMOTE_MOTOR_STATUS	st_remote_motor	PT_REM_MOTOR_CONTROL
SINGLE_INDEX_STATUS	st_single_index	PT_STATION PT_STATION_REPORT
STATION_STATUS	st_station	PT_STATION PT_STATION_REPORT
TEXT_SELECTOR_STATUS	st_text_selector	PT_TEXT_SELECTOR
TRANSACTION_STATUS	st_transaction	PT_CONTROL_OUTPUT PT_DADIG PT_DD PT_MSDD PT_RATIO_INDEX_WRITTEN PT_RCM PT_REM_MOTOR_CONTROL PT_RMSC PT_SET_POINT_OUTPUT PT_STATION_MODE

Table E-2. st_station Structure

Parameter	Description
STATION_STATUS st_station	Status table entry for station points.
q	The quality of a station point.
GOOD_QUALITY	Station is in good quality.
BAD_QUALITY	Station is in bad quality.
la	The quality of a station point.
NO_ALARM	Station has no alarm.
LOW_LIMIT_EXCEEDED	Low alarm limit exceeded.
HIGH_LIMIT_EXCEEDED	High alarm limit exceeded.
da	The deviation alarm of a station point.
NO_ALARM	Station has no alarm.
LOW_LIMIT_EXCEEDED	Low limit exceeded.
HIGH_LIMIT_EXCEEDED	High limit exceeded.
rt	The red tagged status of the station point.
NOT_TAGGED	Station is not red tagged.
RED_TAGGED	Station is red tagged.
spt	Point tracking status of a station.
POINT_NOT_TRACKING	Station not tracking.
POINT_TRACKING	Station is tracking.
byp	Indicates the bypassed status of a station.
NO	The station is not being bypassed.
YES	The station is being bypassed.
mi	Sets the manual interlock of the station.
DISABLED	The station does not have manual interlock enabled.
ENABLED	The station has manual interlock enabled.
ot	Indicates output status of a station.
POINT_NOT_TRACKING	Station is not output tracking.
POINT_TRACKING	Station is output tracking.
dsf	Indicates the digital station failure status of a station.
NO	No digital station failure.
YES	A digital station failure has been encountered.
cok	Indicates the computer OK signal has not been received from the host computer.
NOT_RECEIVED	The computer OK signal has not been received from the host computer.
RECEIVED	The computer OK signal has been received from the host computer.

Table E-2. st_station Structure (continued)

Parameter	Description
lev	Indicates the control level of a station.
STATION_LOCAL_LEVEL	Station is at the local level.
STATION_COMPUTER_LEVEL	Station is at the computer level.
crn	indicates the cascade/ratio control strategy of a station.
DISABLE	Station cascade/ratio control strategy is not used.
ENABLED	Station cascade/ratio control is used.
am	Indicates the mode of the station.
STATION_MANUAL_MODE	Station is in manual mode.
STATION_AUTOMATIC_MODE	Station is in automatic mode.

Table E-3. st_memory Structure

Parameter	Description
MEMORY_STATUS st_memory	Status table entry for memory points.
q	Indicates the quality of the RCM.
GOOD_QUALITY	RCM is in good quality.
BAD_QUALITY	RCM is in bad quality.
alm	Indicates whether the RCM is in alarm.
NORMAL	RCM is in the normal state (no alarm condition).
ALARM	RCM is in alarm.
tag	Indicates the red tag status of a RCM.
NOT_RED_TAGGED	RCM is not red tagged.
RED_TAGGED	RCM is red tagged.
ov	Indicates the output value of a RCM.
ZERO	Value of the RCM is zero (0).
ONE	Value of the RCM is one (1).
si	Indicates whether the logic set input signal has been received.
NO	Logic set input signal has not been received.
YES	Logic set input signal has been received.
sp	Indicates whether the set permissive input signal has been received.
NO	Set permissive input signal not received.
YES	Set permissive input signal received.
ri	Indicates whether the logic reset input signal has been received.
NO	Logic reset input signal has not been received.
YES	Logic reset input signal has been received.
or	Indicates the override status of the RCM.
NO	RCM status not being overridden.
YES	RCM status is being overridden.

Table E-3. st_memory Structure (continued)

Parameter	Description
fb	Indicates the status of the feedback of a RCM.
NO	Feedback signal has not been received.
YES	Feedback has been received.
sc	Indicates whether the set command signal has been received.
NO	Set command signal has not been received.
YES	Set command signal has been received.
rc	Indicates whether the reset command signal has been received.
NO	Reset command signal has not been received.
YES	Reset command signal has been received.

Table E-4. st_module Structure

Parameter	Description
MODULE_STATUS st_module	Status table entry for module points.
es	Indicates the error summary status of a module status point.
NO	No errors exist in the module.
ERRORS_EXIST	At least one error exists in the module.
mode	Indicates the mode of the module.
CONFIGURE_MODE	Module is in the configure mode.
FAILED_MODE	Module is in the failed mode.
ERROR_MODE	Module is in the error mode.
EXECUTIVE_MODE	Module is in the execute mode.
type	Indicates the type of module.
TYPE_EXTENDED	Module is an extended module type.
TYPE_IMAOM01	Module is a IMAOM01.
TYPE_IMCOM	Module is a IMCOM02, IMCOM03, or a IMCOM04.
TYPE_IMLMM02	Module is a IMLMM02.
TYPE_IMMPC01	Module is a IMMPC01.
TYPE_INBTM	Module is a INBTM01, NLIM01, or a NLIM02.
TYPE_INGCM01	Module is a INGCM01.
TYPE_INLCM01	Module is a INLCM01.
TYPE_INLCM02	Module is a INLCM02.
TYPE_INLCM03	Module is a INLCM03.
TYPE_INPCI01	Module is a INPCI01.
TYPE_MFC	Module is a IMMFC01 or a IMMFC02.
TYPE_NAMM01	Module is a NAMM01.
TYPE_NAMM02	Module is a NAMM02.
TYPE NBIM	Module is a NLIM01, NLIM02, NBIM01, or a INBIM02.
TYPE_NCTM01	Module is a NCTM01.
TYPE_NLIM	Module is a NLIM01, NLIM02, or a NPIM01.
TYPE_NLMM01	Module is a NLMM01.

Table E-4. st_module Structure (continued)

Parameter	Description
TYPE_NLSM01	Module is a NLSM01 or an INPCT01.
ftx	Indicates whether this is the first time the module has been put into execute mode.
SET	Automatic initialization input is in the set mode.
msc	This is a miscellaneous status bit.
NO	Backup is ok (MFC/MFP) or Memory is ok (IMLMM02).
YES	Backup is bad (MFC/MFP) or Memory is bad (IMLMM02).
rio	Remote I/O status or remote output status (IMAMM02).
NO	I/O status or output status is good.
YES	I/O status or output status is bad.
lio	Local I/O status.
NO	I/O status is good.
YES	I/O status is bad.
cal	Calibration quality status.
NO	Calibration quality is good
YES	Calibration quality is bad.
aie	Automatic initialization input status.
NO	The input status is reset.
YES	The input status is set.
eai	Indicates whether the ROM memory contains the default configuration.
NO	ROM does not contain default configuration.
YES	ROM contains default configuration.
sta	indicates the summary station status.
GOOD_QUALITY	Summary station status is in good quality.
BAD_QUALITY	Summary station status is in bad quality.
s_cim_module	Indicates which structure in the un_s3 union to access.
COMPUTER_INTERFACE_MODULE	Module is a computer interface. Access the st_cim structure in the union below.
ALL_OTHER_MODULES	Module is not a computer interface module. Access the st_all structure in the union below.
UNKNOWN_MODULES	Module does not report its type. Thus the type of module is unknown. A limited amount of information is available in the st_unknown structure.
union un_s3	Union of all of the module types. One of the three structures defined in Tables E-5, E-6, and E-7. The s_cim_module member indicates which structure to access.

Table E-5. st_cim Structure

Parameter	Description
CIM_MODULES st_cim	Status table entry for cim points.
no	Indicates the nodes off-line status of a module status point.
NONE	None of the nodes are off-line.
ONE_OR_MORE	At least one node is off-line.
mf	Indicates the memory full status of a module status point.
NO	Memory is not full.
YES	Memory is full.
nef	Indicates the node environment failure status for a module status point. This only applies to a INPCI01 or INPCI03 module.
NO	There is not a node environment failure.
YES	There is a node environment failure.
lo	Indicates the loop off-line status of a module status point. This only applies to a INICI01 or INICI03 module.
NONE	None of the loops are off-line.
ONE_OR_MORE	At least one loop is off-line.
sdf	Indicates a failure in the keylock device. A grace period timer is ticking. After the grace period timer expires the ICI will go offline until the device is repaired.
YES	Indicates a failure in the keylock device.
NO	Indicates no failure in keylock device.
nodetype	Node type of the module.
INTERFACE_UNIT	Computer interface is a generic node in INFI 90 OPEN system.
MCS_CONSOLE	Computer interface belongs to an MCS console.
OIS_CONSOLE	Computer interface belongs to an OIS console.
ip	Indicates the internal problems or node environment failure status of a module status point.
NO	No internal problems exist.
YES	Internal problems exist.
rel1	Indicates whether a loop receive error is on channel one.
NO	No loop receive errors.
YES	Loop receive errors.
rel2	Indicates whether a loop receiver error is on channel two.
NO	No loop receive errors.
YES	Loop receive errors.
tel1	Indicates whether a loop transmit error is on channel one.
NO	No loop transmit errors.
YES	Loop transmit errors.
tel2	Indicates whether a loop transmit error is on channel two.
NO	No loop transmit errors.
YES	Loop transmit errors.

Table E-5. st_cim Structure (continued)

Parameter	Description
lc1	Indicates the status of loop channel one.
BUSY	Loop channel is busy.
IDLE	Loop channel is not busy (idle).
lc2	Indicates the status of loop channel two.
BUSY	Loop channel is busy.
IDLE	Loop channel is not busy (idle).
lcf	Indicates whether a loop communications failure has occurred on the local loop.
NO	No loop communication failures have been encountered.
YES	Loop communication failures have been encountered.
uc_hostvalue	This is the value set by the host computer. This can be any value that the host has set in the computer interface.

Table E-6. st_all Structure

Parameter	Description
ALL_MODULES st_all	Status table entry for all module points
uc_mod_err	Module error code.
uc_x	X value of the error that corresponds with the uc_mod_err.
uc_y	Y value of the error that corresponds with the uc_mod_err.

Table E-7. st_unknown Structure

Parameter	Description
UNKNOWN_MODULE st_unknown	Status table entry for unknown module points.
cim_modules st_cim	Refer to the explanation of the st_cim structure described in Table E-5.
all_modules st_all	Refer to the explanation of the st_all structure described in Table E-6.

Table E-8. st_extended Structure

Parameter	Description
EXTENDED_STATUS st_extended	Status table entry for extended points.
module_status st_base_status	Refer to Table E-4 for a description of module_status.
uc_extended_type	Indicates type of module.
MOD_TYPE_BCM	BCM type module.
MOD_TYPE_ICT	ICT type module.
MOD_TYPE_IIMCP02	IIMCP02 type module.
MOD_TYPE_IST	IST type module.
MOD_TYPE_MCP	MCP type module.

Table E-8. st_extended Structure (continued)

Parameter	Description
MOD_TYPE_MFP	MFP type module.
MOD_TYPE_NPM	NPM type module.
MOD_TYPE_PST	PST type module.
MOD_TYPE_SBM	SBM type module.
MOD_TYPE_SCM	SCM type module.
uc_module_version	Module version number. (i.e. an IMMFC01 = version 1, and an IMMFP03 = version 3)
uc_revision_level[2]	Firmware revision level (in ASCII) of the module. The uc_revision_level[0] character is the letter and uc_revision_level[1] is the number. (i.e. A1).

Table E-9. st_analog Structure

Parameter	Description
ANALOG_STATUS st_analog	Status table entry for analog points.
q	The quality of an analog point.
GOOD_QUALITY	Analog point is in good quality.
BAD_QUALITY	Analog point is in bad quality.
la	Limit alarm status of an analog point.
NO_ALARM	Analog point has no alarm.
LOW_LIMIT_EXCEEDED	Low alarm limit has been exceeded.
HIGH_LIMIT_EXCEEDED	High alarm limit has been exceeded.
da	Deviation alarm status of an analog point.
NO_ALARM	The analog point has no deviation alarm.
LOW_LIMIT_EXCEEDED	Low alarm limit has been exceeded.
HIGH_LIMIT_EXCEEDED	High alarm limit has been exceeded.
rt	The red tagged status of an analog point.
NOT_TAGGED	Analog point is not red tagged.
RED_TAGGED	Analog point is red tagged.
pt	Point tracking status of an analog point.
POINT_NOT_TRACKING	Analog point is not tracking.
POINT_TRACKING	Analog point is tracking.
ccv	The calibration correction value status of an analog point.
OK	Calibration correction value ok.
OUT_OF_RANGE	Calibration correction value is out of range.

Note: Elements rt, pt, and ccv are not used in the st_analog structure. These field so not contain relevant information when reading data information from PT_ANALOG and PT_REAL4_ANALOG import points. When writing to PT_ANALOG_REPORT and PT_REAL4_ANALOG _REPORT export points, the zero value should be used for these elements.

Table E-10. st_process Structure

Parameter	Description
PROCESS_STATUS st_process	Status table entry for process points.
q	Quality of a station point.
GOOD_QUALITY	Station is in good quality.
BAD_QUALITY	Station is in bad quality.
la	Limit alarm status of a station point.
NO_ALARM	Station point has no alarm limit.
LOW_LIMIT_EXCEEDED	Low alarm limit has been exceeded.
HIGH_LIMIT_EXCEEDED	High alarm limit has been exceeded.
da	Deviation alarm status of a station point.
NO_ALARM	The station point has no deviation alarm.
LOW_LIMIT_EXCEEDED	Low alarm limit has been exceeded.
HIGH_LIMIT_EXCEEDED	High alarm limit has been exceeded.
rt	Red tag status of a station point.
NOT_TAGGED	Station is not red tagged.
RED_TAGGED	Station is red tagged.
spt	Point tracking status of a station point.
POINT_NOT_TRACKING	Station is not set point tracking.
POINT_TRACKING	Station is set point tracking.

NOTE: Elements la, da, and rt are not used for PT_RMSC and PT_RMSC_REPORT. These fields do not contain relevant information when reading data information from a PT_RMSC import point. When writing to a PT_RMSC_REPORT export point, the zero value should be used for these elements.

Table E-11. st_digital Structure

Parameter	Description
DIGITAL_STATUS st_digital	Status table entry for digital points.
q	Quality of a digital point.
GOOD_QUALITY	Digital is in good quality.
BAD_QUALITY	Digital is in bad quality.
la	Limit alarm status of a digital.
NORMAL	Digital is not in alarm.
ALARM	Digital in is alarm.
v	The value of a digital point.
ZERO	Digital has a value of zero (0).
ONE	Digital has a value of one (1).

Table E-12. st_aquisition Structure

Parameter	Description
ACQUISITION_STATUS st_acquisition	Status table entry for acquisition points.
q	Quality of the DAANG point.
GOOD_QUALITY	DAANG is in good quality.
BAD_QUALITY	DAANG is in bad quality.
ha	High alarm status of the DAANG point.
NO	DAANG is not in high alarm.
YES	DAANG is in high alarm.
la	Low alarm status of the DAANG point.
NO	DAANG is not in low alarm.
YES	DAANG is in low alarm.
lev	Alarm level of the DAANG point.
DAANG_LEVEL1	DAANG is in alarm level 1.
DAANG_LEVEL2	DAANG is in alarm level 2.
DAANG_LEVEL3	DAANG is in alarm level 3.
xs	Extended status of the DAANG point.
NO	Extended status has not been changed.
YES	Extended status has been changed.
rt1	Red tag status of the DAANG point.
NOT_TAGGED	DAANG is not red tagged.
RED_TAGGED	DAANG is red tagged.
am1	Mode of the DAANG point.
DAANG_MANUAL_MODE	DAANG is in manual mode.
DAANG_AUTOMATIC_MODE	DAANG is in automatic mode.
pi	Permit input status of the DAANG point.
NO	DAANG will not permit input.
YES	DAANG will permit input.
ce	The constraints enabled status of the DAANG point.
NO	DAANG does not have constraints enabled.
YES	DAANG has constraints enabled.
cal2	Value calculated status of the DAANG point.
NO	DAANG value has not been calculated.
YES	DAANG value has been calculated.
hf2	Hardware failure/bad input quality status of the DAANG point.
NO	A hardware failure has not been detected.
YES	A hardware failure has been detected.
ml	Multilevel alarm status of the DAANG point.
NO	The DAANG is not in multilevel alarming.
YES	The DAANG is in multilevel alarming.

Table E-12. st_aquisition Structure (continued)

Parameter	Description
am2	Mode of the DAANG point.
DAANG_MANUAL_MODE	DAANG is in the manual mode.
DAANG_AUTOMATIC_MODE	DAANG is in the automatic mode.
uc_byte3	Reserved for future use.
rt4	Red tag status of the DAANG point.
NOT_TAGGED	DAANG is not red tagged.
RED_TAGGED	DAANG is red tagged.
hf4	Hardware failure/bad quality status of the DAANG point.
NO	No hardware failure has been detected.
YES	A hardware failure has been detected.
or	Suspect or out of range status of the DAANG point.
NO	DAANG status is not out of range.
YES	DAANG status is out of range.
lim	Limited status of the DAANG point.
NO	DAANG status is not limited.
YES	DAANG status is limited.
am4	Mode of the DAANG point.
DAANG_MANUAL_MODE	DAANG is in manual mode.
DAANG_AUTOMATIC_MODE	DAANG is in automatic mode.
cal4	Value calculated status of the DAANG point.
NO	DAANG value has not been calculated.
YES	DAANG value has been calculated.
qo	Quality override active status of the DAANG point.
NO	DAANG quality has not been overridden.
YES	DAANG quality has been overridden.
ss	No report or off scan status of the DAANG point.
NO	DAANG block is in normal exception reporting mode.
YES	DAANG block is off scan (no exceptions generated).
hd	High deviation status of the DAANG point.
NO	DAANG is not in high deviation alarm.
YES	DAANG is in high deviation alarm.
ld	Low deviation status of the DAANG point.
NO	DAANG is not in low deviation alarm.
YES	DAANG is in low deviation alarm.
hr	High rate status of the DAANG point.
NO	DAANG is not in high rate.
YES	DAANG is in high rate.

Table E-12. st_aquisition Structure (continued)

Parameter	Description
lr	Low rate status of the DAANG point.
NO	DAANG is not in low rate.
YES	DAANG is in low rate.
va	Variable alarm status of the DAANG point.
NO	DAANG has no variable alarms.
YES	DAANG has variable alarms.
as	Alarm suppression indication status of the DAANG point.
DISABLE	DAANG has alarm suppression disabled.
ENABLE	DAANG has alarm suppression enabled.
ra	Alarm in re-alarm condition status of the DAANG point.
NO	DAANG is not in the re-alarm condition.
YES	DAANG is in the re-alarm condition.

Table E-13. st_pointtype6 Structure

Parameter	Description
POINTTYPE6_STATUS st_pointtype6	Status table entry for pointtype6 points.
process_status st_process	Refer to the explanation of the st_process structure described in Table E-10.
station_status st_station	Refer to the explanation of the st_station structure described in Table E-2.

Table E-14. st_single index Structure

Parameter	Description
SINGLE_INDEX_STATUS st_single_index	Status table entry for single index points.
station_status st_station	Refer to the explanation of the st_station structure described in Table E-2.
uc_reply_ss	Reply code for the set station status control command. Refer to Table 9-4 for reply codes.
uc_reply_sp	Reply code for the set station set point control command. Refer to Table 9-4 for reply codes.
uc_reply_ri	Reply code for the set ratio index control command. Refer to Table 9-4 for reply codes.
uc_reply_rc	Reply code for the set control output control command. Refer to Table 9-4 for reply codes.

Table E-15. st_multistate Structure

Parameter	Description
MULTISTATE_STATUS st_multistate	Status table entry for multistate points.
q	Quality of the multistate device driver point.
GOOD_QUALITY	Multistate device driver point is in good quality.
BAD_QUALITY	Multistate device driver point is in bad quality.
alm	The alarm status of the MSDD point.
NORMAL	The MSDD is in normal mode.
ALARM	The MSDD is in alarm mode.
sor	The status override value equal to one status of the MSDD point.
NO	The status override is not equal to one.
YES	The status override is equal to one.
cor	The control override value equal to one MSDD point.
NO	The control override value is not equal to one.
YES	The control override is equal to one.
m	Operating mode of the MSDD point.
MSDD_MANUAL_MODE	MSDD is in the manual mode of operation.
MSDD_AUTOMATIC_MODE	MSDD is in the automatic mode of operation.
tag	Red tagged status of the MSDD point.
NOT_TAGGED	MSDD is not red tagged.
RED_TAGGED	MSDD is red tagged.
co	Control output value equal to one status of the MSDD point.
NO	Control output is not equal to one.
YES	Control output value is equal to one.
fb1	Input one feedback state equal to one status of the MSDD point.
NO	Input one feedback state is not equal to one (1).
YES	Input one feedback state is equal to one (1).
fb2	Input two feedback state equal to one status of the MSDD point.
NO	Input two feedback state is not equal to one (1).
YES	Input two feedback state is equal to one (1).
fb3	Input three feedback state equal to one status of one (1).
NO	Input three feedback state is not equal to one (1).
YES	Input three feedback state is equal to one (1).
fb4	Input four feedback state equal to one status of the MSDD.
NO	Input four feedback state is not equal to one (1).
YES	Input four feedback state is equal to one (1).

Table E-15. st_multistate Structure (continued)

Parameter	Description
gs	The good state value of the MSDD point.
MSDD_STATE_ZERO	The good state is equal to zero (0).
MSDD_STATE_ONE	The good state is equal to one (1).
MSDD_STATE_TWO	The good state is equal to two (2).
MSDD_STATE_THREE	The good state is equal to three (3).
rs	The requested state value of the MSDD point.
MSDD_STATE_ZERO	The requested state is equal to zero (0).
MSDD_STATE_ONE	The requested state is equal to one (1).
MSDD_STATE_TWO	The requested state is equal to two (2).
MSDD_STATE_THREE	The requested state is equal to three (3).

Table E-16. st_device_driver Structure

Parameter	Description
DEVICE_DRIVER_STATUS st_device_driver	Status table entry for analog points.
q	Quality of the device driver point type.
GOOD_QUALITY	Device driver is in good quality.
BAD_QUALITY	Device driver is in bad quality.
alm	The alarm status of the device driver point.
NORMAL	Device driver is in the normal mode of operation (no alarms).
ALARM	Device driver is in the alarm mode of operation.
tag	Red tag status of the device driver point.
NOT_TAGGED	The device driver is not red tagged.
RED_TAGGED	The device driver is red tagged.
ov	Output value of the device driver point.
ZERO	Value of the device driver is zero (0).
ONE	Value of the device driver is one (1).
fb1	Input one feedback state equal to one status of the device driver point.
NO	Input one feedback state is not equal to one.
YES	Input one feedback state is equal to one.
fb2	Input two feedback state equal to one status of the device driver point.
NO	Input two feedback state is not equal to one.
YES	Input two feedback state is equal to one.
fs	The feedback status of the device driver point.
GOOD_QUALITY	The feedback status is in good quality.
BAD_QUALITY	The feedback status is in bad quality.
or	The override value equal to one status of the device driver point.
NO	The override value is not equal to one.
YES	The override value is equal to one.

Table E-16. st_device_driver Structure (continued)

Parameter	Description
mode	The operating mode of the device driver point.
DD_AUTOMATIC_MODE	The device driver is in the automatic mode of operation.
DD_REMOTE_MODE	The device driver is in the remote mode of operation.
DD_MANUAL_MODE	The device driver is in the manual mode of operation.

Table E-17. st_remote_motor Structure

Parameter	Description
REMOTE_MOTOR_STATUS st_remote_motor	Status table entry for analog points.
q	The quality of the RMC point.
GOOD_QUALITY	The RMC is in good quality.
BAD_QUALITY	The RMC is in bad quality.
alm	The alarm status of the RMC.
NORMAL	RMC is in the normal mode of operation.
ALARM	RMC is in alarm.
fb1	Input one feedback state equal to one status of the RMC point.
NO	Input one feedback is not equal to one.
YES	Input one feedback is equal to one.
fb2	Input two feedback state equal to one status of the RMC point.
NO	Input two feedback is not equal to one.
YES	Input two feedback is equal to one.
tag	Red tag status of the RMC point.
NOT_TAGGED	RMC is not red tagged.
RED_TAGGED	RMC is red tagged.
ov	The output value of the RMC point.
STOPPED	The value of the RMC is stopped (0).
RUNNING	The value of the RMC is running (1).
bs	Bad start status of the RMC point.
NO	RMC did not have a bad start status.
YES	RMC had a bad start status.
f	Fault status of the RMC.
NO	RMC is not in fault.
YES	RMC is in fault.
sp1	Start permissive one state status of the RMC point.
NO	Start permissive one state is not allowed.
YES	Start permissive one state is allowed.
sp2	Start permissive two state status of the RMC point.
NO	Start permissive two state is not allowed.
YES	Start permissive two state is allowed.

Table E-17. st_remote_motor Structure (continued)

Parameter	Description
rmc_err	These are the error codes for the bad start (bs) or fault (f) conditions.
RMC_NO_ERROR	No errors in the RMC.
RMC_STOPPED	RMC is stopped.
RMC_INTERLOCK1_ZERO	RMC has interlock one equal to zero (0).
RMC_INTERLOCK2_ZERO	RMC has interlock two equal to zero (0).
RMC_INTERLOCK3_ZERO	RMC has interlock three equal to zero (0).
RMC_INTERLOCK4_ZERO	RMC has interlock four equal to zero (0).
RMC_FEEDBACK1_ZERO	RMC has feedback one equal to zero (0).
RMC_FEEDBACK2_ZERO	RMC has feedback two equal to zero (0).
RMC_FEEDBACK1_ONE	RMC has feedback one equal to one (1).
RMC_FEEDBACK2_ONE	RMC has feedback two equal to one (1).

Table E-18. st_dadig Structure

Parameter	Description
DADIG_STATUS st_dadig	Status table entry for dadig points.
q	Quality of the DADIG point.
GOOD_QUALITY	DADIG is in good quality.
BAD_QUALITY	DADIG is in bad quality.
alm	Alarm status of the DADIG point.
NORMAL	DADIG is in the normal mode of operation (no alarms).
ALARM	DADIG is in alarm.
realm	Re-alarm status of the DADIG point.
NO	DADIG is not in the re-alarm condition.
YES	DADIG is in the re-alarm condition.
sup	Alarms suppressed status of the DADIG point.
NO	DADIG does not have the alarms suppressed.
YES	DADIG has the alarms suppressed.
os	Output suspect status of the DADIG point.
NO	DADIG output does not have a status that is suspect.
YES	DADIG output has a status that is suspect.
nr	Off scan or no report status of the DADIG point.
NO	DADIG is on scan (exception reporting).
YES	DADIG is off scan (not exception reporting).
tag	Red tag status of the DADIG point.
NOT_TAGGED	DADIG is not red tagged.
RED_TAGGED	DADIG is red tagged.
ov	Output value of the DADIG point.
ZERO	Value of the DADIG is zero (0).
ONE	Value of the DADIG is one (1).

Table E-18. st_dadig Structure (continued)

Parameter	Description
lat	Extended status transition latched status of the DADIG point.
NO	Extended status is not latched.
YES	Extended status is latched.
qo	Quality override status of the DADIG point.
NO	Quality of the DADIG is not being overridden.
YES	Quality of the DADIG is being overridden.
sp	Set permissive status of the DADIG point.
NO	DADIG does not have the set permissive.
YES	DADIG has the set permissive.
pi	Primary input select status of the DADIG point.
NO	DADIG does not have the primary input selected.
YES	DADIG has the primary input selected.
ai	Alternate/user input selected status of the DADIG point.
NO	DADIG does not have the alternate (user) input selected.
YES	DADIG has the alternate (user) input selected.

Table E-19. st_transaction Structure

Parameter	Description
TRANSACTION_STATUS st_transaction	Status table entry for transaction points.
uc_trans	Status of the control transaction. When control is requested of the block in the INFI 90 OPEN system the computer interface will return to the user with a return code of ICI_OK when the control message has been sent to the INFI 90 OPEN system. To determine the actual status of the control message another read is required which will pass back the status of the control command here in the uc_trans member. Refer to Section 9 for an explanation of the valid values for the uc_trans member.

Table E-20. st_text_selector Structure

Parameter	Description
TEXT_SELECTOR_STATUS st_text_selector	Status table entry for text selector points.
q	The quality of the text selector point.
GOOD_QUALITY	The text selector point is in good quality.
BAD_QUALITY	The text selector point is in bad quality.

APPENDIX F - TUNING A MODULE

INTRODUCTION

This appendix describes how to tune function blocks from the host computer. The computer interface point table is not used when tuning function blocks.

MODULE TUNING

The computer interface provides two functions that the host computer uses in order to tune a function code block. These two functions are **READ BLOCK** and **TUNE BLOCK**.

Figure F-1 details the general block data format required for configuration functions.

1. From the host computer, issue the **READ BLOCK** function to read the actual block data for the function code. This function will return the block data in a stream of bytes.

2. Use the general block data diagram in Figure F-1 to parse through the stream of bytes returned by **READ BLOCK**. Modify the desired tunable specifications in the byte stream. None of the other bytes in the byte stream should be modified as the entire block data stream must be written back to the module when using the **TUNE BLOCK** function.

After the general block data stream has been modified, issue the **TUNE BLOCK** function to actually modify the tunable specifications of the function code block.

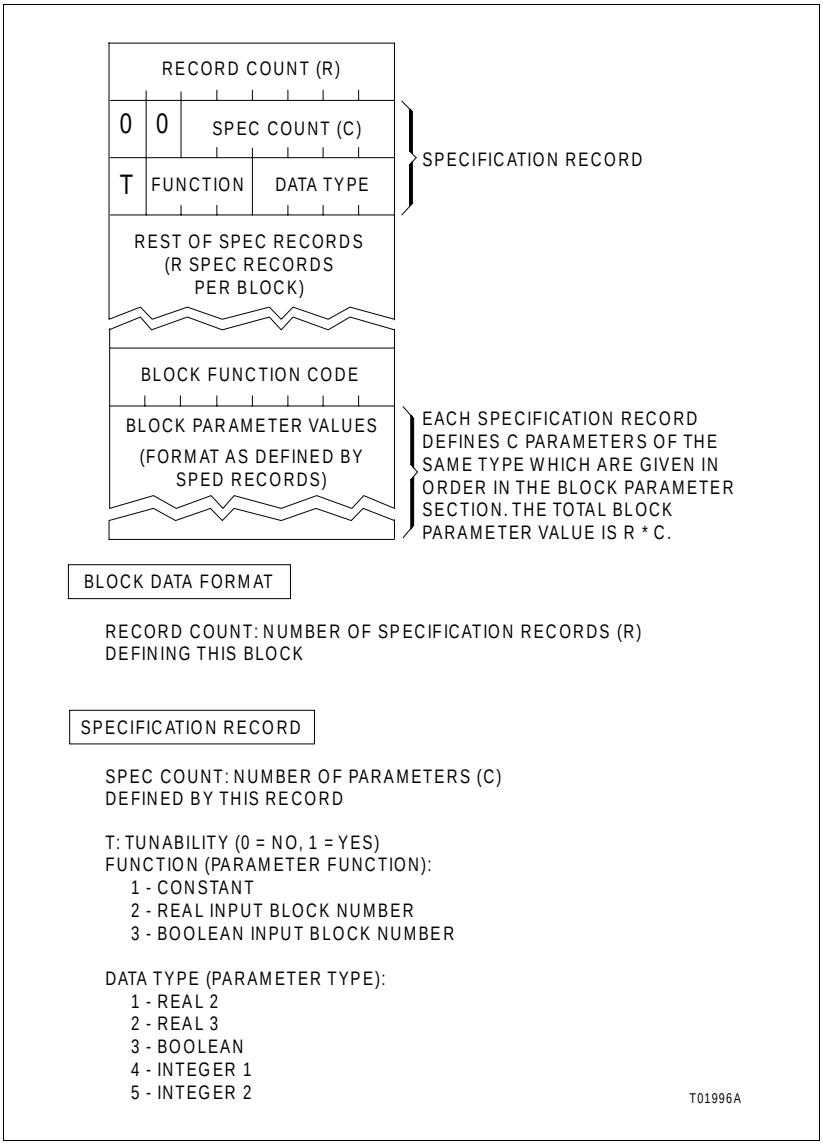


Figure F-1. General Block Data Format

APPENDIX G - SPECIFICATION TABLE DESCRIPTION

INTRODUCTION

The Strategic Enterprise Management Application Programming Interface (semAPI) functions return point specifications in a structure called a SPEC_TABLE. This structure is a union of various specification types and is a key into the union. The specification structure is defined in the file ***l3stable.h***.

SPEC_TABLE

The key member s_point_type will indicate which member in the structure has valid data. The **READ DATA SPECS** function returns a SPEC_TABLE structure. The following is a description of the SPEC_TABLE structure.

s_point_type

This is the key to the union. The value of this member will indicate which structure member in the union to access for data.

SPECS_ANALOG_POINTS
Specifications for Real-3 analog points. Access the st_analog structure member. Valid point types are:

PT_PROCESS_VARIABLE
PT_ANALOG

SPECS_STATION_SPECS_POINTS
Specifications for station points. Access the st_station_specs structure member. Valid point type is:

PT_SET_POINT

SPECS_DIGITAL_POINTS
Specifications for digital points. Access the st_digital structure member. Valid point type is:

PT_DIGITAL

SPECS_RCM_POINTS
Specifications for remote control memory (RCM) points. Access the st_rcm structure member. Valid point type is:

PT_RCM

SPECS_STATION_READ_POINTS
Specifications for station read points. Access the st_station_read structure member. Valid point type is:

PT_STATION

SPECS_RMSC_POINTS
Specifications for remote manual set constant (RMSC) points. Access the st_rmsc structure member. Valid point type is:

PT_RMSC

SPECS_REAL4_POINTS
Specifications for Real-4 analog points. Access the st_real4 structure member. Valid point types are:

PT_REAL4_ANALOG_READ

SPECS_DAANG_POINTS
Specifications for data acquisition analog (DAANG) points. Access the st_daang structure member. Valid point type is:

PT_DAANG

SPECS_USER_POINTS
Specifications for ASCII string points. Access the st_user structure member. Valid point type is:

PT_ASCII_STRING

SPECS_RMC_POINTS
Specifications for remote motor control (RMC) points. Access the st_rmc structure member. Valid point type is:

PT_RMC

SPECS_DD_POINTS
Specifications for device driver (DD) points. Access the st_dd structure member. Valid point type is:

PT_DD

SPECS_MSDD_POINTS
Specifications for multi-state device driver (MSDD) points. Access the st_msdd structure member. Valid point type is:

PT_MSDD

SPECS_DADIG_POINTS
Specifications for data acquisition digital (DADIG) points. Access the st_dadig structure member. Valid point type is:

PT_DADIG

un_points

The union of various point specification types. The member s_point_type will indicate which structure in the union to access.

ANALOG_SPECS
st_analog

Structure for analog specifications.

uc_engineering_units
Engineering units descriptor index number for analog points. For a station this is the engineering unit index of the process variable.

f_variable_zero
Floating zero point value for an analog point. For a station this is the zero of the process variable.

f_variable_span
Floating point value for the span of an analog point. For a station this is the span of the process variable.

f_high_alarm
Floating point value for the high alarm limit of an analog point.

f_low_alarm
Floating point value for the low alarm limit of an analog point.

STATION_SPECS
st_station_specs

Structure for station specifications.

uc_engineering_units
The engineering units descriptor index number for the set point.

f_point_zero
Floating point value for the zero of a set point.

f_point_span
Floating point value of the span for a set point.

f_deviation
Floating point value of the deviation alarm for a set point.

DIGITAL_SPECS
st_digital

Structure for digital specifications.

uc_alarm_spec
Alarm specification for a digital point. Valid values include:

LOGIC_ZERO_ALARM
Logic zero (0) is the alarm state for the digital point.

LOGIC_ONE_ALARM
Logic one (1) is the alarm state for the digital point.

NO_ALARM_STATE
No alarm state is defined for the digital point.

RCM_SPECS
st_rcm

Structure for remote control memory (RCM) specifications.

uc_switch_type
Display type for an RCM point. The value of SPEC 8 in the block (function code 62).

STATION_READ_SPECS
st_station_read

Structure for station specifications.

uc_engineering_units
Engineering units descriptor index number for the station.

f_high_alarm
Floating point value for the high alarm limit of a station point.

f_low_alarm
Floating point value for the low alarm limit of a station point.

f_deviation
Floating point value for the deviation alarm of a station point.

f_span
Floating point value of the set point (SP) and process variable (PV) span.

f_variable_zero
Floating point zero value of the process variable.

f_point_zero
Floating point zero value of the set point.

uc_station_type
The station type. Valid values include:

BASIC_WITH_SETPOINT
Basic station with a set point variable.

RATIO_INDEX
Ratio index station.

CASCADE
Cascade type station.

BASIC_WITHOUT_SETPOINT
Basic station without the set point variable.

BASIC_WITH_BIAS
Basic station with BIAS.

RMSC_SPECS
st_rmsc

Structure for remote manual set constant (RMSC) specifications.

uc_engineering_units
The engineering units descriptor index number for RMSC points.

f_zero
Floating point zero value for a RMSC.

f_span
Floating point span value for an RMSC point (high limit minus low limit).

f_high_alarm
Floating high alarm limit point value for an RMSC point.

f_low_alarm
Floating point low alarm limit value for an RMSC point.

REAL4_SPECS
st_real4

Structure for real-4 analog specifications.

uc_engineering_units
Engineering units descriptor index number for Real-4 Analog points.

f_zero
Floating point zero value for a Real-4 Analog point.

f_span
Floating point span (high limit minus low limit) value for a Real-4 Analog point.

f_high_alarm
Floating point high alarm limit value for a Real-4 Analog point.

f_low_alarm
Floating point low alarm limit value for a Real-4 Analog point.

DAANG_SPECS
st_daang

Structure for data acquisition analog (DAANG) specifications.

uc_engineering_units
The engineering units descriptor index number for DAANG points.

f_high_alarm
Floating point high alarm limit value for a DAANG point.

f_low_alarm
Floating point low alarm limit value for a DAANG point.

f_user
Floating point user inserted value for a DAANG point.

f_high_display
Floating point high display reference value for a DAANG point.

f_center_display
Floating point center display reference value for a DAANG point.

f_low_display
Floating point low display reference value for a DAANG point.

uc_switch_type
DAANG type (mode setting). Valid values include:

DAANG_MANUAL
DAANG is in the manual mode.

DAANG_AUTOMATIC_INP
DAANG is in the automatic mode (Input).

DAANG_AUTOMATIC_CALC
DAANG is in the automatic mode (Calculated).

DAANG_SUPPRESS_ALM
DAANG has alarms suppressed.

DAANG_UNSUPPRESS_ALM
DAANG does not have alarms suppressed.

DAANG_OFF_SCAN
DAANG is OFF scan.

DAANG_ON_SCAN
DAANG is ON scan.

DAANG_FORCE_XR
DAANG should force an exception report.

USER_SPECS st_user

Structure for the ASCII string specification.

uc_data_class
Class of data that is stored in the ASCII string.

uc_data_format
Format of data in the ASCII string.

uc_status_size
Number bytes in the status of the ASCII string.

uc_data_size
Number of bytes in the data of the ASCII string.

uc_alarm_code
Alarm code of the ASCII string block.

uc_byte_count
Number of specification bytes that are filled in the uc_data array. The maximum number of bytes is equal to MAX_UDXR_SPEC_BYTES.

uc_data [MAX_UDXR_SPEC_BYTES]
Array of specification bytes. The actual number of bytes in the array is determined using the uc_byte_count parameter.

RMC_SPECS
st_rmc

Structure for RMC specifications.

uc_display_type
The display type for an RMC point. The value of SPEC 14 in the RMC block (function code 136).

DD_SPECS
st_dd

Structure for device driver specifications.

uc_display_type
Display type for an device driver point. The value of SPEC 10 in the DD block (function code 123).

MSDD_SPECS
st_msdd

Structure for MSDD (multi-state device driver) specifications.

uc_display_type
The display type for an MSDD point. The value of SPEC 18 in the MSDD block (function code 129).

DADIG_SPECS
st_dadig

Structure for DADIG specifications.

uc_display_type
The display type for an DADIG point. The value of SPEC 17 in the DADIG block (function code 211).

APPENDIX H - TIME STRUCTURE DESCRIPTION (ICI_TIME.H)

INTRODUCTION

The Strategic Enterprise Management Application Programming Interface (semAPI) functions may require the user to pass a TIME_STRUCT structure as input to the function. The semAPI function library handles times throughout the system using this TIME_STRUCT definition. The time structure is defined in the header file (***ici_time.h***)

TIME STRUCTURE DEFINITIONS

The following represents valid values for the time units (uc_units) member of the TIME_STRUCT structure. Table H-1 lists and describes valid values. Table H-2 provides time structure examples.

Table H-1. Valid Values for Time Units

Parameter	Description
TIME_STRUCT	The structure that defines the time values that are passed to the API function.
uc_units	Time units.
ICI_MS	Interpret the s_number as milliseconds.
ICI_TMS	Interpret the s_number as 10 milliseconds.
ICI_HMS	Interpret the s_number as 100 milliseconds.
ICI_SEC	Interpret the s_number as seconds.
ICI_MIN	Interpret the s_number as minutes.
ICI_HRS	Interpret the s_number as hours.
s_number	The number of the time units (refer to Table H-2).

Table H-2. Time Structure Examples

Example	Description
TIME_STRUCT st_time={ICI_SEC,5}	Represents 5 seconds.
TIME_STRUCT st_time={ICI_MIN,10}	Represents 10 minutes.

APPENDIX I - ERROR STRUCTURE DESCRIPTION

INTRODUCTION

Each and every Strategic Enterprise Management Application Programming Interface (semAPI) function returns an ERR_STRUCT structure and a return status as return parameters. This appendix explains the various levels of the error structure. The error structure is defined in the header file (***ici_err.h***).

NOTE: Always zero the error structure before each use.

FUNCTION RETURN STATUS

The following defines valid returns from the semAPI functions. This return parameter declares as a short data type. Valid values include:

ICI_OK The function completed successfully. The user data is filled in with valid data.

ICI_FATAL A fatal error was encountered while trying to issue the function. Check the error structure for specifics about the error. The user data area is not filled in.

ICI_WARNING A warning was encountered while trying to issue the function. Check the error structure for specifics about the warning. An application would typically continue on warning messages. The user data is filled in with data. The user may decide that the warning has caused the data to be suspect.

ICI_CONTINUE The return status is only valid on s_retrieve_quick_reply using the quick access method. It indicates that the reply to a quick function has not been received. The application should try to receive the reply again later. The user data area is filled in.

ERR_STRUCT DEFINITIONS

ERR_STRUCT structure indicates the success or failure of the function. The error structure is organized in levels in order to pinpoint the point of failure. Refer to ***TROUBLESHOOTING*** in Section 9 for more details and corrective action. The following is the format of an error structure and return status.

s_level Indicates the overall type of error. If several errors occur while processing a function, the Post severe error will be reflected in the s_level (i.e. if two warnings and a fatal error occurs, then the s_level will be set to ICI_FATAL since that is more severe than a warning).

Valid values include:

ICI_FATAL
A fatal error has occurred while processing the function. If several errors have occurred, only one error per level will be saved. Check the bits in the s_err_layer member to determine which levels have encountered errors.

ICI_WARNING
A warning has occurred while processing the function. If several warnings have occurred, only one per level will be saved. Check the bits in the s_err_layer member to determine which levels have encountered warnings.

s_err_layer This is a bit mask, which will indicate which levels in the error structure have been filled in with error messages. The following bit mask values exist to check the s_err_layer member:

ICI_FATAL_ERR
A fatal computer interface error has occurred. Check the ici structure and the s_fatal member.

DD_FATAL_ERR
A fatal device driver error has occurred. Check the dd structure and the s_fatal member.

MD_FATAL_ERR
A fatal message driver error has occurred. Check the md structure and the s_fatal member.

SUB_FATAL_ERR
A fatal function error has occurred. Check the sub structure and the s_fatal member.

GEN_FATAL_ERR
A fatal general error has occurred. Check the gen structure and the s_fatal member.

ICI_WARN_ERR
An ICI warning has occurred. Check the ici structure and the s_warn member.

DD_WARN_ERR
A device driver warning has occurred. Check the dd structure and the s_warn member.

MD_WARN_ERR
A message driver warning has occurred. Check the md structure and the s_warn member.

SUB_WARN_ERR
A function warning has occurred. Check the sub structure and the s_warn member.

GEN_WARN_ERR
A general warning has occurred. Check the gen structure and the s_warn member.

ici Structure containing the computer interface specific error returns.

s_fatal
Value of the fatal error. Refer to Section 9 for error descriptions.

s_warn
Value of the warning. Refer to Section 9 for error descriptions.

dd Structure containing the device driver specific error returns.

s_fatal
Value of the fatal error. Refer to Section 9 for error descriptions.

s_warn
Value of the warning. Refer to Section 9 for error descriptions.

md Structure containing the message driver specific error returns.

s_fatal
Value of the fatal error. Refer to Section 9 for error descriptions.

s_warn
Value of the warning. Refer to Section 9 for error descriptions.

subs Structure containing the function specific error returns.

s_fatal
Value of the fatal error. Refer to Section 9 for error descriptions.

s_warn
Value of the warning. Refer to Section 9 for error descriptions.

s_stat
Value of additional status information. This may be used if the s_fatal and s_warn are not sufficient to report errors back to the user.

gen General errors are miscellaneous and do not fit into the other categories.

s_fatal
Value of the fatal error. Refer to Section 9 for error descriptions.

s_warn
Value of the warning. Refer to Section 9 for error descriptions.

Refer to ***I*** in Section 6 for details on converting numeric error numbers into text strings.

APPENDIX J - INFI 90 OPEN ADDRESS STRUCTURE

INTRODUCTION

The Strategic Enterprise Management Application Programming Interface (semAPI) functions may require the user to pass an INFI 90 OPEN address. A generic structure handles INFI 90 OPEN addresses called INFI90ADR.

INFI90ADR

Table J-1 describes the address structure of INFI90ADR structure. This structure is defined in ***ici_user.h***.

> **NOTE:** On a Plant Loop computer interface, the s_loop parameter is ignored. Remember also that on the Plant Loop computer interface there is a different MAX and MIN for the s_node parameter.

Table J-1. INFI 90 OPEN Address Structure

Parameter	Description
INFI90ADR	The combination of the loop, node, module, and block define the location of a function code block in the INFI 90 OPEN system.
s_loop	The INFI 90 OPEN loop number. The loop number must be greater than or equal to MIN_LOOP_ADR and less than or equal to MAX_LOOP_ADR.
s_node	The INFI 90 OPEN node or PCU number. For INFI-NET the PCU number must be greater than or equal to MIN_INFI_NODE_ADR and less than or equal to MAX_INFI_NODE_ADR. For Plant Loop the PCU number must be greater than or equal to MIN_NET_NODE_ADR and less than or equal to MAX_NET_NODE_ADR.
s_module	The INFI 90 OPEN module number. The module number must be greater than or equal to MIN_MODULE_ADR and less than or equal to MAX_MODULE_ADR.
s_block	The INFI 90 OPEN block number. The block number must be greater than or equal to MIN_BLOCK_ADR and less than or equal to MAX_BLOCK_ADR.

APPENDIX K - QUICK MESSAGE IDENTIFIER

INTRODUCTION

The *Quick Access* version of the Strategic Enterprise Management Application Programming Interface (semAPI) Functions require the user to pass the address of a quick message identifier called MSG_ID.

MSG_ID

Table K-1 describes the quick access message identifier structure. This structure is defined in the header file ***ici_user.h***.

Table K-1. MSG_ID Structure

Parameter	Description
MSG_ID	The combination of elements that make up the identification of a quick message.
s_msg_num	The message number returned from the semAPI function. This number is used in the call to s_retrieve_quick_reply to get the response for a quick message.
p_callback	This member is reserved for future use. Set to null.
vp_data	This member is reserved for future use. Set to null.

APPENDIX L - TAG NAME ACCESS

TAG NAME ACCESS

Tag name access is only available through the INOSM01 module. All other computer interface modules return a DD (device driver) fatal error (INVAL_MSG_TYPE) when attempting to use tag name access.

The official ICI_TAGNAME structure and the #define for USE_TAGNAME is defined in the header file ***ici_user.h***.

> **NOTE:** Using tag names as opposed to indices reduces performance (longer execution time). The semAPI software must convert all tag names to indices when issuing the commands to the computer interface module.

The following describes the available referencing options to points in the computer interface module.

- Refer to points using the index only. This is the fastest method, however not very flexible. As tag names change in the system, application programs may need to be re-written.
- Refer to points using tag names during start-up. Use tag names in the application at start-up, and then use **READ TAG INDICES** to convert all tag names to indices. This method allows for application flexibility by using tag names during start-up and using indices for fast access during run time.
- Refer to points using tag names. This method is the slowest but is most flexible, because as tag names change, the application programs do not need to be re-written.

User Parameter Area

For each instance of an index, an ICI_TAGNAME structure is added to the user parameter area. The user can reference indices or tag names. For index reference, fill in the index value and ignore the ICI_TAGNAME structure. For tag name reference, fill in the ICI_TAGNAME structure and set the index value equal to USE_TAGNAME.

User Data Area

For each instance of an index, ICI_TAGNAME structure is added to the user parameter area. The user can receive tag names or not receive tag names. To receive tag names in the user data area, the application must set the

c_return_tagnames member to YES in the **CONNECT TO LOGICAL COMPUTER INTERFACE** function. When this member is YES, the ICI_TAGNAME structure is returned along with the index. When this member is NO, ignore the ICI_TAGNAME structure.

APPENDIX M - TIME SYNC ACCURACY

INTRODUCTION

This appendix explains how to set the INICI03 INFI-NET to Computer Interface (ICI) time synchronization accuracy and lists typical values used for time synchronization accuracy.

TALK90 UTILITIES SETUP

To set the system time and date of all nodes in a system (loop), the ICI interface must have the highest time synchronization accuracy. To set the time synchronization accuracy, invoke the firmware version of the TALK90 utilities on the ICI interface. Refer to ***INFI-NET to Computer Interfaces (INICI01/03)*** for more information on using the TALK90 utilities.

The following hardware is required to invoke the ICI firmware version of the TALK 90 utilities:

- Diagnostic terminal or a personal computer with a terminal emulation software package.
- Serial cable with a male DB-9 connector for the INICT03A INFI-NET to Computer Transfer Module. The other end of the serial cable must have the correct type of connector (number of pins and gender) for the computer or terminal serial port.

To set up the required hardware to use TALK90 utilities:

1. Connect the male DB-9 connector end of the serial cable to the female DB-9 connector (P4) located on the side of the INICT03A module.

2. Connect the other end of the serial cable to the computer or terminal.

3. Set pole 4 of dipswitch SW4 (UUB0) to one (open or off) to enable the port one utility option.

4. The INICT03A serial port one communication speed should be set for 9600 baud. To set the correct communication rate, set dipswitch SW1 pole 5 to 0 (closed) and poles 6 through 8 to 1 (open).

5. Set the INICIT03A serial port communication parameters to eight data bits, 1 stop bit, and no parity. Set poles 2 and 3 of dipswitch SW4 to 0 or (closed).

To use the TALK90 utilities:

1. Press **Enter** to access the utilities menu.

2. From the utilities menu select option **9** and the following prompts are displayed.

Default Time Sync Accuracy is currently n.
Change Accuracy (Y?N)?
Enter New Accuracy (0 to 15):

Type **Y** to change the time synchronization accuracy, then enter the desired accuracy at the appropriate prompt. Table M-1 lists typical values used to set time synchronization accuracy.

Table M-1. Time Synchronization Accuracy Values

Value	Description
0	Low accuracy
3	Low accuracy battery backed
6	High accuracy
9	OIS 20 console
11	ODMS system
12	Satellite clock system

APPENDIX N - HARDWARE CONFIGURATION

INTRODUCTION

The Strategic Enterprise Management Application Programming Interface (semAPI) software requires a software key (provided with the software) and proper jumper settings for operation. This section explains how to install the software key and the required jumper settings. Included is a wiring diagram showing how to interconnect the INICI03 interface modules.

INSTALLING THE SOFTWARE KEY

The software key is used with the NTMP01 termination unit or the NIMP01 termination module. Use the appropriate procedure to install the software key and to set the jumpers.

NTMP01 Termination Unit

If using an NTMP01 termination unit:

1. For the software key, set jumpers J1, J3, J8, J9, J10 and J18 on the NTMP01 termination unit as shown in Figure N-1.

2. Connect the male end of the software key to the port labeled P6 (printer port) on the NTMP01 termination unit. Refer to Figure N-1 for the location of P6.

3. For the computer port, set jumpers J4, J5, J6, and J7 as shown in Figure N-1.

4. Determine the requirements of the application and set up the termination unit as DTE or DCE equipment by setting J2 accordingly. Refer to Figure N-1 for J2 jumper settings.

5. Connect the appropriate end of the serial communication cable to P5 (computer port).

NIMP01 Termination Module

If using an NIMP01 termination module:

1. For the software key, set jumpers J1, J8, J9, J10, J13 and J20 on the NIMP01 termination module as shown in Figure N-2.

2. Connect the male end of the software key to the 25 pin end of the 9-to-25 pin adapter.

3. Connect the 9 pin end of the 9-to-25 pin adaptor to the port labeled P6 (printer port) on the NIMP01 termination module. Refer to Figure N-2 for the location of the port.

4. For the computer port, set jumpers J6, J5, J7, and J20 as shown in Figure N-2.

5. Determine the requirements of the application and set up the termination unit as DTE or DCE equipment by setting J2 accordingly. Refer to Figure N-2 for J2 jumper settings.

6. Connect the appropriate end of the serial communication cable to P5 (computer port).

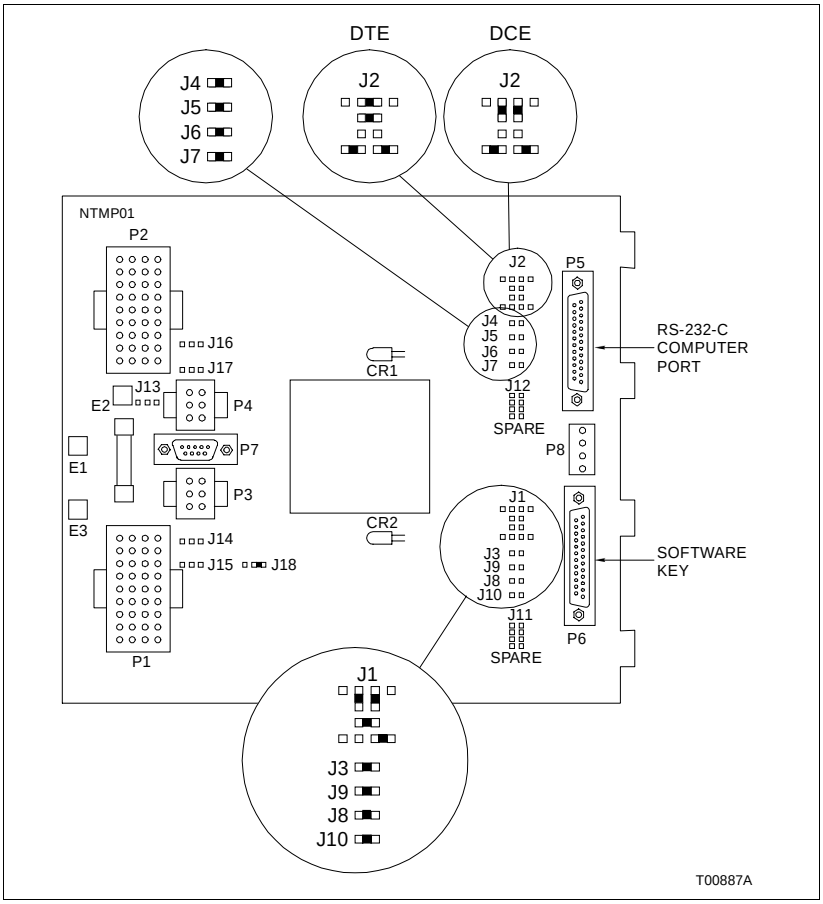


Figure N-1. NTMP01 Connector Assignments and Jumper Settings

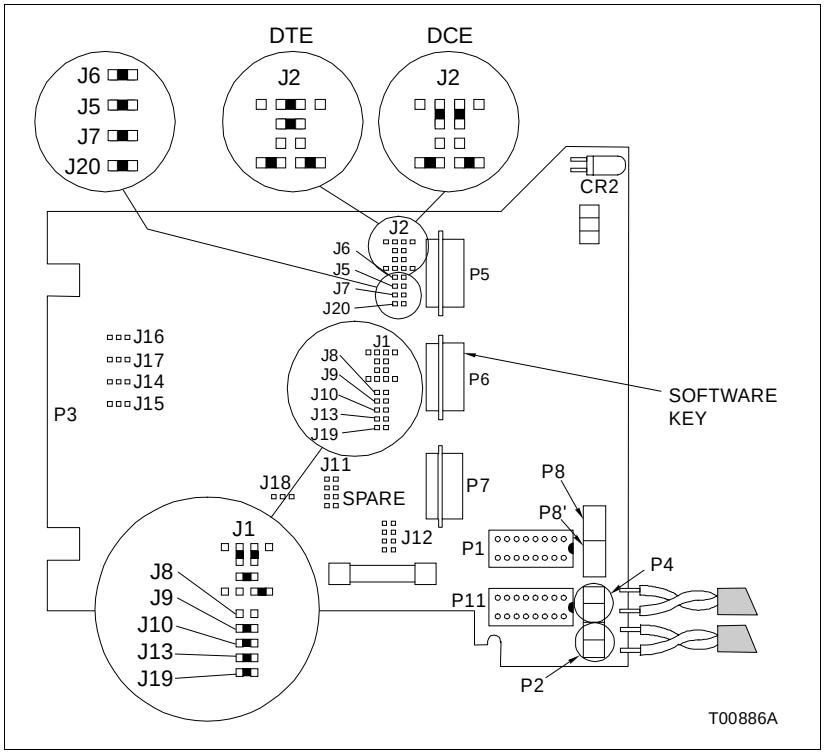


Figure N-2. NIMP01 Connector Assignments and Jumper Settings

INICI03 INTERFACE WIRING

Figure N-3 shows how to wire the INICI03 interface. Refer to **INFI-NET to Computer Interfaces (INICI01/03)** to set the interface jumpers and dipswitches before installing the interface.

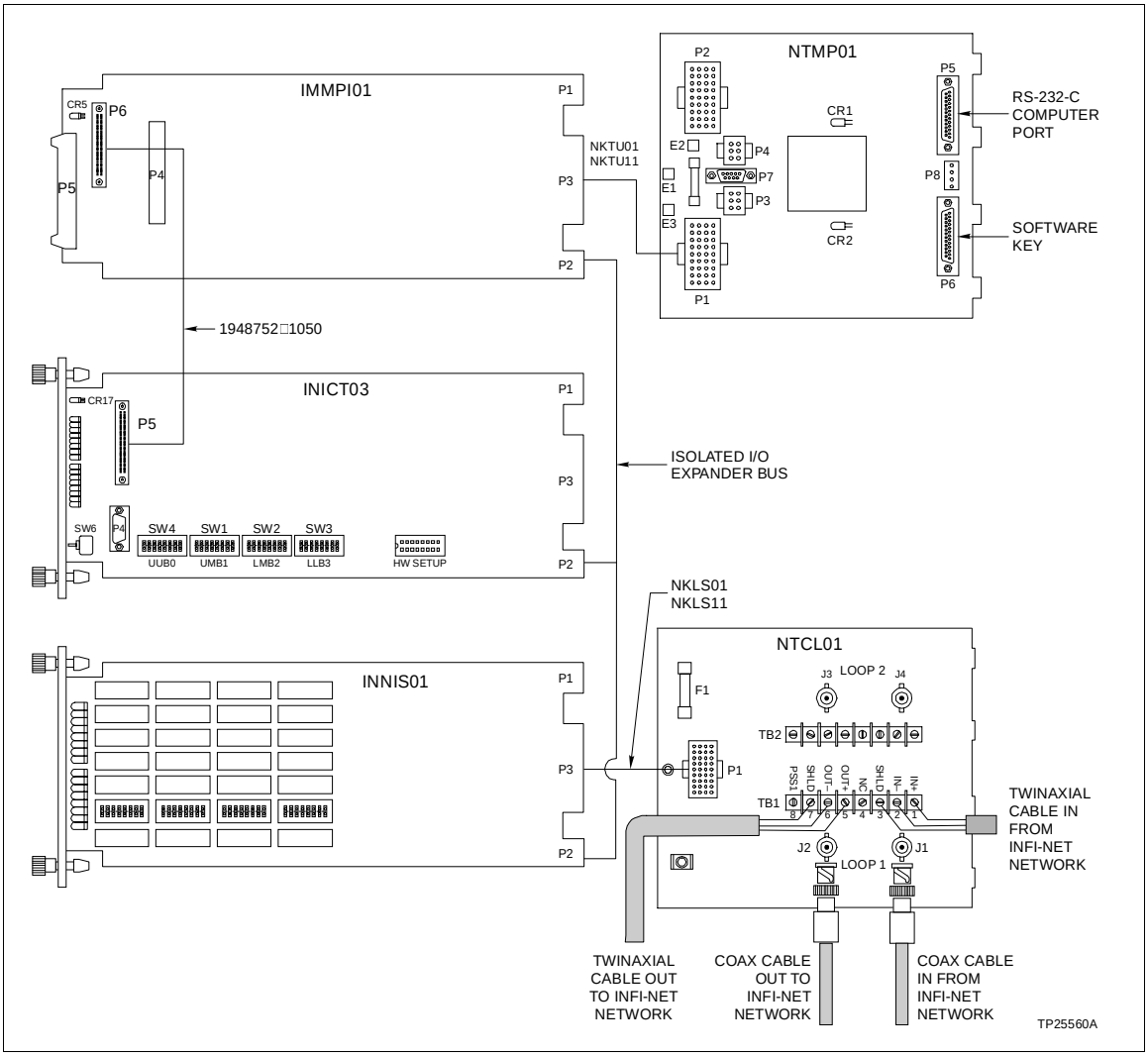


Figure N-3. Wiring Diagram, INICI03 Interface

Index

Index (continued)

Visit Elsag Bailey on the World Wide Web at http://www.ebpa.com

Our worldwide staff of professionals is ready to meet *your* needs for process automation.
For the location nearest you, please contact the appropriate regional office.

AMERICAS
29801 Euclid Avenue
Wickliffe, Ohio USA 44092
Telephone 1-216-585-8500
Telefax 1-216-585-8756

ASIA/PACIFIC
152 Beach Road
Gateway East #20-04
Singapore 189721
Telephone 65-391-0800
Telefax 65-292-9011

EUROPE, AFRICA, MIDDLE EAST
Via Puccini 2
16154 Genoa, Italy
Telephone 39-10-6582-943
Telefax 39-10-6582-941

GERMANY
Graefstrasse 97
D-60487 Frankfurt Main
Germany
Telephone 49-69-799-0
Telefax 49-69-799-2406

Form WBPEEUI350252B0 Litho in U.S.A. 0997

Elsag Bailey
Process Automation