Contents

List of Figures    vii
List of Tables    ix

Introduction    xi
Typographical Conventions    xii
Software and Hardware    xii
  Software    xii
  Hardware    xiii
Network Configuration    xiv
Product Support    xiv

1. Silicon Graphics' 3270 HLLAPI    1
    IBM 3270 PC API to Silicon Graphics 3270 HLLAPI Changes    2
    Silicon Graphics 3270 HLLAPI Capabilities    3
Contents

Subroutine Descriptions  4
  Determine the Session ID   4
  Initialize the Silicon Graphics 3270 HLLAPI Code   5
  Determine the Value of the Session Parameters   6
  Determine the Current Cursor Position   7
  Connect to a 3270 Presentation Space   7
  Disable Keyboard Input During HLLAPI Activity   8
  Simulate Keyboard Data Entry   9
  Enable Keyboard Input When HLLAPI Activity Is Completed  10
  Disconnect from a 3270 Presentation Space  11
  Copy Data between 3270 Presentation Space and Application
  Data Area  11
  Read OIA Status Group  13
  Determine if the Screen Has Been Updated   13
  Copy Status Line to an Application Data Area   14
  Start Silicon Graphics Message Mode   15
  Stop Silicon Graphics Message Mode   16
  Read Silicon Graphics Message Mode Protocol Data Units   16
  Read a 3270 Structured Field   17
  Write 3270 Structured Field to the Host   18
  Trace Silicon Graphics 3270 HLLAPI Data   19
  Get Silicon Graphics 3270 HLLAPI Information for
  a Specified Session   19
  User Entry Point into the IRIS 3270 Emulator   20

2. Troubleshooting  23
  Troubleshooting Using the Trace Display Utility   23
  Resolving Errors   24
    Error Logging   24
    Error Information Sources   25
    Configuration Errors   25
    Nonconfiguration Errors   25
    SGI HLLAPI Trace Display Interpretation   26
List of Figures

Figure 2-1  3270 Trace Display Window  24
## List of Tables

<table>
<thead>
<tr>
<th>Table B-1</th>
<th>Scan Codes</th>
<th>29</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table B-2</td>
<td>Buffer Codes</td>
<td>31</td>
</tr>
</tbody>
</table>
Introduction

The Silicon Graphics® IRIS® 3270 Emulator products provide a high-speed communications link between an IRIS-4D® Series workstation and an IBM® host computer running either Virtual Machine/Conversational Monitor System (VM/CMS) or Multiple Virtual Storage/Time Sharing Option (MVS/TSO).

With the IRIS 3270 Emulator, you can use an IRIS-4D workstation or server—with Systems Network Architecture (SNA), Transmission Control Protocol (TCP), or 5080 Graphics System Workstation connectivity—to access your IBM mainframe. The list below shows how the various link types of the IRIS 3270 Emulator product connect to an IBM host.

<table>
<thead>
<tr>
<th>IRIS 3270 Emulator Product:</th>
<th>Connects to IBM Host through:</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNA 3270 Emulator</td>
<td>IBM 3705, 3725, or 3745 front end processor</td>
</tr>
<tr>
<td>TCP 3270 Emulator</td>
<td>IBM 8232 Ethernet controller</td>
</tr>
<tr>
<td>3270 for the 5080 Emulator</td>
<td>IRIS Channel Adapter (ICA)</td>
</tr>
</tbody>
</table>

This document describes how to create High-Level Language Application Program Interface (HLLAPI) applications that interact with IBM host applications using Silicon Graphics’ 3270 HLLAPI programming environment and tools. It is organized to provide programming information, including 3270 HLLAPI functionality and troubleshooting.

Chapter 1 explains Silicon Graphics’ 3270 HLLAPI programming environment. Chapter 2 discusses troubleshooting using the trace display utility.

Note: For information on how to use and configure the IRIS 3270 Emulator, refer to the IRIS 3270 Emulator User’s Guide.
Typographical Conventions

These type conventions and symbols are used in this guide:

*Italics* Filenames, variables, IRIX command arguments, command flags, titles of publications, icon names

**Bold** Subroutine names

**UPPERCASE** IBM file names, command names and the names of keys used on IBM systems

*Screen type* Code examples, file excerpts, and screen displays (including error messages)

**Bold Screen type**

User input

() (Parentheses) Following IRIX commands, they surround the reference page (man page) section where the command is described

[] (Brackets) Surrounding optional syntax statement arguments

# IRIX shell prompt for the superuser (*root*)

Software and Hardware

The IRIS 3270 Emulator software and hardware communicate between the IRIS workstation and an IBM-host system running VM/CMS or MVS/TSO. For information on system requirements, refer to the release notes included with your 3270 emulator product.

Software

The IRIS 3270 Emulator software includes these key files and directories:

/opt/3270/bin/ Contains all executable files for 3270
/opt/3270/chest/ Contains Toolchest menu files
/opt/3270/font/ Contains all fonts used by the emulator
/opt/3270/lib Contains libsgi3270.a, the 3270 HLLAPI link library.
/var/opt/3270/example/ Contains examples of 3270 HLLAPI, IRISXFR and IND$FILE file transfer automation, file transfer input redirection, and keyboard input tracing.
/var/opt/3270/file/ Contains all log files used in problem determination
/var/opt/3270/spool/ Default location for files transferred to or from the host and screen captures
/var/opt/3270/lib/ Contains the 3270 configuration files
/usr/lib/X11/app-defaults/Setup3270 Contains color schemes for setup3270(1)

To use Silicon Graphics’ IRISXFR file transfer utility, you must install the IRISXFR program on your IBM host. This software is distributed on a 1/2-inch tape, generated at 1600 bytes per inch (BPI). For information on how to install the host software, refer to the installation instructions included with the IRISXFR distribution.

**Hardware**

**Warning:** Do not attempt to add boards or other upgrades in your system. Hardware upgrades should be installed only by Silicon Graphics-certified personnel. Upgrades performed by noncertified persons void your warranty and may damage your system, or cause injury to improperly trained individuals.
Network Configuration

The IRIS workstation running 3270 emulator software can be connected to an IBM host through these network configurations:

- a leased line using Synchronous Data Link Control (SDLC) via an IBM 37X5 front end processor
- Ethernet, Token Ring, or FDDI using IBM 3172 or equivalent controller
- local non-SNA channel attach using the IRIS Channel Adapter (ICA)

The typical IRIS 3270 Emulator hardware configuration includes:

- an IBM, or IBM compatible, mainframe computer
- an IBM host front-end processor running Network Control Program (NCP)
- an IRIS SNA workstation running the IRIS 3270 Emulator software (SNA mode) through an IRIS SNA gateway
- an IRIS SNA gateway running the IRIS 3270 Emulator software (SNA mode)
- an IRIS Channel Adapter gateway and connections

Product Support

Silicon Graphics provides comprehensive technical hardware and software product support and a maintenance program for IRIS products. For more information, refer to the release notes that accompany this product.
Chapter 1

Silicon Graphics’ 3270 HLLAPI

The Silicon Graphics 3270 high-level language application program interface (HLLAPI) provides a programming environment and tools to create 3270 HLLAPI applications that communicate with IBM mainframe applications. SNA, TCP, and 5080 modes are supported. The Silicon Graphics 3270 HLLAPI supports one session in SNA, TCP, and 5080 mode. Model 2, 3, 4, and 5 terminals are supported.

The Silicon Graphics 3270 HLLAPI also supports 3270 structured field data flows when using an SNA, TCP, or 5080 connection.

Silicon Graphics 3270 HLLAPI applications allow you to:

- create your own customized 3270 user interface for displaying and processing keyboard information
- automate repetitive sequences and run dialogues between the IRIS and IBM host unattended
- achieve faster throughput using structured field and Silicon Graphics Message Mode capabilities

The Silicon Graphics 3270 HLLAPI provides HLLAPI functionality equivalent to IBM 3270 PC application program interface (API) as specified in the IBM 3270 Emulation Program, Version 3.00, Application Program Interface and Host Reference manual, document version SC23-0960-0.

Input and return information is passed in a C structure rather than in registers. The values and meaning of returned codes are left unaltered. The number and type of the input structure members have also been maintained except for modifications introduced to support Silicon Graphics 3270 HLLAPI enhancements.

This chapter assumes that you understand the information contained in the IRIS 3270 Emulator User’s Guide on using and configuring the IRIS 3270 Emulator.
IBM 3270 PC API to Silicon Graphics 3270 HLLAPI Changes

The following five changes map IBM PC Assembler implementation of HLLAPI to Silicon Graphics HLLAPI using C language:

- A one-to-one mapping between service requests and C subroutine calls eliminates the need to map the information in registers AH, AL, BH, BL, CX, and DX. This mapping also eliminates the need for the “system return code” that indicates improper values in these registers.
- The pointer to the parameter list (registers DX and ES) is replaced by a pointer to a structure.
- All instances of segment address/offset address in a parameter list are mapped onto pointers.
- All instances of “Must be zero/Unchanged” or “Reserved/Reserved” in the parameter list format definition are not implemented in the equivalent C structure.
- All elements of each C structure are aligned on 32-bit word boundaries. This protects the user against future changes (a byte field can be expanded to 2 bytes or a short word expanded to a long word).

Three subdirectories under /var/opt/3270/example provide detailed examples of how to use the Silicon Graphics 3270 HLLAPI. All data structures used by the Silicon Graphics HLLAPI are defined in /usr/include/sys/hl_user_struct.h. All constants used as input values or returned values are defined in /usr/include/sys/hl_user_define.h. A short description of the contents of each subdirectory is provided below.

Case 1 provides an example of using hl_entry_point to create a filter routine within the Silicon Graphics-supplied t3279 program. In this case, the keyboard and display processing is done by Silicon Graphics-supplied routines.

Case 2 is obsolete.

Case 3 provides an example of a 3270 emulation with prespecified input and no display processing. The screen descriptors supply the input. A comment file contains the output generated and processes the screen descriptors. To view this file, enter:

```
cat comment_output
```
You can also view a trace file by entering:

cp trace log /usr/3270/file

and then entering:

display_3270trace

Note: All code may be copied and used in your development effort.

Silicon Graphics 3270 HLLAPI Capabilities

The capabilities of Version 7.0 of the Silicon Graphics 3270 HLLAPI are listed here. The features listed in italic type are not provided by the IBM 3270-PC API.

- Determine the session ID
- Determine the value of the session parameters
- Determine the current cursor position
- Connect to the 3270 Presentation Space
- Disable keyboard input during HLLAPI activity
- Simulate keyboard data entry, that is, write data to the host application
- Enable keyboard input when HLLAPI activity is completed
- Disconnect from the 3270 Presentation Space
- Copy between the 3270 Presentation Space and an application data
- Read OIA group status
- Reinitialize Silicon Graphics 3270 HLLAPI after a UNIX® exec call while preserving the host session (for SNA, TCP, and 5080 modes only)
- Determine if the 3270 Presentation Space has been updated without a status line change
- Copy the status line from the 3270 Presentation Space into an application data area
- Start and stop Silicon Graphics Message Mode
- Read and write Silicon Graphics Message Mode Protocol Data Units
Subroutine Descriptions

Each description in this section explains the purpose of the subroutine, how it interacts with other Silicon Graphics 3270 HLLAPI subroutines, and the possible values for each element of the structure passed to the called routine. See `/usr/include/sys/hl_user_struct.h` for the precise definition of each structure used by the service request subroutines. Appendix B lists the error messages that might be written to `/var/opt/3270/file/hllapi_log` in the event that an error is detected.

**Determine the Session ID**

**Calling Sequence**

```c
void hl_query_session_id (struct session_id
 *session_id_struct)
```

**Purpose**
Verify that the specified session is available.

**Usage**
Call this subroutine once for each desired session at the start of an application using the Silicon Graphics 3270 HLLAPI.

**Input Structure Members**

- `session_id` 0x01—indicating a request to connect to Session 1
- `link_type` 0x01—indicating a 5080 connection
  - 0x04—indicating SNA data over a leased line
  - 0x05—indicating a TCP/IP connection
- `lu_name` reserved

**Returned Structure Members**

- `session_ret` 0x00—indicating success
  - 0x02—if Session ID is invalid
  - 0x2b—indicating a 3270 HLLAPI start up problem
Subroutine Descriptions

0x38—indicating that shared memory used by the Silicon Graphics HLLAPI could not be allocated

0x39—indicating that the maximum number of sessions are already running

**shmid** shared memory ID, used for t3279/ps3279 communication

**Initialize the Silicon Graphics 3270 HLLAPI Code**

Calling Sequence

```c
void hl_init (struct init *init_struct)
```

Purpose

Initialize the Silicon Graphics 3270 code at startup time.

Usage

Call this subroutine once, each time the Silicon Graphics 3270 HLLAPI application is started.

**Input Structure Members**

- **session_id**
  - 0x00—for first time initialization unique session number after an `exec` call command
- **command**
  - 0x01—indicates a first time initialization request
  - 0x02—indicates an initialization request after an `exec` call
- **config_name**
  - full path name of configuration file
- **host_name**
  - TCP/IP hostname for SGI gateway or IBM host
- **terminal_pool**
  - LU name used only by SNA
- **signal_rtn**
  - signal catching routine for SIGPOLL

**Returned Structure Members**

- **session_ret**
  - 0x00—indicating success
Determine the Value of the Session Parameters

Calling Sequence

```c
void hl_query_session_params (struct session_params
*session_params_struct)
```

Purpose
Determine the values of session characteristics that affect the operation of a Silicon Graphics 3270 HLLAPI application.

Usage
This subroutine can be called at any time after the Session ID has been determined. The Silicon Graphics 3270 HLLAPI deviates from the IBM 3270-PC API in that alternate screen sizes are supported. Extended Attributes are supported. Programmed Symbols are not supported.

Input Structure Members

- `session_id` unique session number

Returned Structure Members

- `session_type` 0x02—is always returned
- `session_char` 0x00—if no extended attributes and no programmed symbols
  - 0x40—if programmed symbols, but no extended attributes
  - 0x80—if extended attributes, but no programmed symbols
  - 0xc0—if extended attributes and programmed symbols
- `session_rows` number of rows in session’s Presentation Space
- `session_cols` number of columns in session’s Presentation Space
- `session_ret` 0x00—if successful
- 0x02—if Session ID is invalid
Determine the Current Cursor Position

Calling Sequence

```c
void hl_query_session_cursor (struct session_cursor *session_cursor_struct)
```

Purpose
Determine the current cursor position and type.

Usage
This subroutine must be called whenever the Silicon Graphics 3270 HLLAPI application needs to determine the current cursor position and type. Row and column addresses start at 0, not 1.

Input Structure Members

- `session_id` unique session number

Returned Structure Members

- `cursor_type` 0x00—if underscore cursor, always returned
- `cursor_row` current row address of cursor
- `cursor_col` current column address of cursor
- `cursor_pos` returned address, as offset
- `status` 0x00—does not display cursor
- `session_ret` 0x00—if successful
  0x01—if displays cursor
  0x02—if Session ID is invalid

Connect to a 3270 Presentation Space

Calling Sequence

```c
void hl_connect (struct connect *connect_struct)
```

Purpose
Connect a Silicon Graphics 3270 HLLAPI application to an active 3270 Presentation Space.
Usage

This subroutine serializes access to keyboard and copy services. The routine must be called at the start of an Silicon Graphics 3270 HLLAPI application and normally is followed by a call to \texttt{hl\_reserve}, which locks the keyboard, preventing operator input.

Input Structure Members

- \textit{session\_id}\hspace{3em}unique session number
- \textit{query\_reply}\hspace{3em}reserved
- \textit{query\_reply\_len}\hspace{3em}reserved

Returned Structure Members

- \textit{keyboard\_ret}\hspace{3em}0x00—if successful
  - 0x02—if Session ID is invalid
  - 0x04—if session already connected for keyboard services

Disable Keyboard Input During HLLAPI Activity

Calling Sequence

\begin{verbatim}
void hl_reserve (struct reserve *reserve_struct)
\end{verbatim}

Purpose

Causes 3270 emulator to disable keyboard input.

Usage

Normally, this subroutine is called immediately after \texttt{hl\_connect} to prevent the intermingling of operator- and application-generated keyboard input.

Input Structure Members

- \textit{session\_id}\hspace{3em}unique session number

Returned Structure Members

- \textit{keyboard\_ret}\hspace{3em}0x00—if successful
  - 0x02—if Session ID is invalid
  - 0x04—if session not connected for keyboard services
Simulate Keyboard Data Entry

Calling Sequence

```c
void hl_send_key (struct send_key *send_key_struct)
```

Purpose

Sends keystroke data from an Silicon Graphics 3270 HLLAPI application as an operator would enter it from a keyboard.

Usage

Used whenever the Silicon Graphics 3270 HLLAPI application wishes to send data to a host program. The Silicon Graphics 3270 HLLAPI deviates from the IBM specification by allowing an unlimited number of keystrokes to be encoded in the keystroke list.

**Note:** 3270 Buffer Code format is supported for all values of CHAR_SET. ASCII scan codes are supported only if CHAR_SET = US_ENGLISH in your configuration file.

Each scan code and shift state is an 8-bit quantity. See Appendix B, “Scan Code and Buffer Code Tables” for scan code shift state definitions.

<table>
<thead>
<tr>
<th>Scan Code</th>
<th>Shift State</th>
<th>Scan Code</th>
<th>Shift State</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>#1</td>
<td>#n</td>
<td>#n</td>
</tr>
</tbody>
</table>

Input Structure Members

- `session_id` unique session number
- `char_set` character set in use (possible values for `char_set` are defined in `hl_user_define.h`; they range from HL_FRENCH to HL_KATAKANA/english)

**Note:** For Katakana applications, CHAR_SET = HL_KATAKANA implies that Katakana characters will be displayed. If English characters should be displayed, set CHAR_SET = HL_KATAKANA/ENGLISH.

- `keystroke_ptr` pointer to list of scan code/shift state pairs
- `total_keys` total number of keystrokes to be sent to the host
Returned Structure Members

keyboard_ret
- 0x00 — if successful and no AID generated
- 0x02 — if Session ID is invalid
- 0x04 — if session not connected for keyboard services
- 0x10 — if invalid scan code or input inhibited condition
- 0x12 — if successful and AID generated
- 0x37 — if char_set invalid

keys_sent
number of keys sent before processing ended

Enable Keyboard Input When HLLAPI Activity Is Completed

Calling Sequence

```
void hl_release (struct release *release_struct)
```

Purpose
Causes 3270 emulator to enable operator keyboard input.

Usage
If hl_reserve has been called, hl_release can be called to reenable keyboard input. The hl_disconnect subroutine also reenables keyboard input automatically.

Input Structure Members

```
session_id
```
unique session number

Returned Structure Members

keyboard_ret
- 0x00 — if successful
- 0x02 — if Session ID is invalid
- 0x04 — if session not connected for keyboard services
Disconnect from a 3270 Presentation Space

Calling Sequence
```
void hl_disconnect (struct disconnect *disconnect_struct)
```

Purpose
Disconnect a Silicon Graphics 3270 HLLAPI application from an active 3270 Presentation Space.

Usage
This subroutine serializes access to keyboard and copy string services. It must always be called prior to exiting the Silicon Graphics 3270 HLLAPI application code.

Input Structure Members
```
session_id unique session number
```

Returned Structure Members
```
keyboard_ret
0x00—if successful
0x02—if Session ID is invalid
0x04—if session not connected for keyboard services
```

Copy Data between 3270 Presentation Space and Application Data Area

Calling Sequence
```
void hl_copy_data (struct copy_data *copy_data_struct)
```

Purpose
Copy part or all of the data from the 3270 Presentation Space to the Silicon Graphics 3270 HLLAPI application data area or vice versa.

Usage
Called whenever data must be moved from the 3270 Presentation Space to the application data area. It is also used to update the 3270 Presentation Space. Only the 3270 device buffer format is supported.
### Input Structure Members

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>source_id</code></td>
<td>0x00—(application to 3270) unique session number (3270 to application)</td>
</tr>
<tr>
<td><code>source_ptr</code></td>
<td>0x00—(3270 to application) pointer to application data area (application to 3270)</td>
</tr>
<tr>
<td><code>source_beg</code></td>
<td>offset to first source character to be copied (both directions)</td>
</tr>
<tr>
<td><code>source_end</code></td>
<td>offset to last source character to be copied (both directions)</td>
</tr>
<tr>
<td><code>target_id</code></td>
<td>0x00—(3270 to application) unique session number (application to 3270)</td>
</tr>
<tr>
<td><code>target_ptr</code></td>
<td>0x00—(3270 to application) pointer to application data area (application to 3270)</td>
</tr>
<tr>
<td><code>target_beg</code></td>
<td>offset to starting copy position in target buffer (both directions)</td>
</tr>
<tr>
<td><code>copy_mode</code></td>
<td>0x00—3270 field attributes not copied (both directions)</td>
</tr>
<tr>
<td></td>
<td>0x40—3270 field attributes copied (3270 to application)</td>
</tr>
</tbody>
</table>

### Returned Structure Members

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>copy_ret</code></td>
<td>0x00—if successful</td>
</tr>
<tr>
<td></td>
<td>0x02—if Session ID is invalid</td>
</tr>
<tr>
<td></td>
<td>0x03—if target is input inhibited</td>
</tr>
<tr>
<td></td>
<td>0x06—if invalid source definition</td>
</tr>
<tr>
<td></td>
<td>0x07—if invalid target definition</td>
</tr>
<tr>
<td></td>
<td>0x09—if truncation occurred</td>
</tr>
<tr>
<td></td>
<td>0x0e—if some or all of the target is protected</td>
</tr>
<tr>
<td></td>
<td>0x0f—if copy of field attributes not allowed</td>
</tr>
</tbody>
</table>
Read OIA Status Group

Calling Sequence

void hl_read_oia (struct read_oia *read_oia_struct)

Purpose
Determine the current input inhibited state.

Usage
This subroutine determines when the 3270 Presentation Space is ready to accept input. It is designed to be used in a polled mode or blocked mode with a timeout. Each unit of time represents 10 milliseconds.

Input Structure Members

- session_id: unique session number
- mode: 0x01—if polling mode desired
- timeout: 0x02—if blocked mode with timeout desired
- timeout: 1 - 6000—if read_mode = 2; 0 otherwise

Returned Structure Members

- oia_ret: 0x00—if successful
- oia_ret: 0x02—if Session ID is invalid
- oia_ret: 0x23—if timeout value not between 1 and 6000

- oia_buf: 5 bytes containing returned OIAG status (see p. 1-1 to reference document SC23-0960-0, p. 3-73)

Determine if the Screen Has Been Updated

Calling Sequence

void hl_screen_status (struct screen_status *screen_status_struct)

Purpose
Determine if any location on the screen has been updated.

Usage
This subroutine determines when the 3270 Presentation Space or Status Line has been updated. It eliminates the need to call hl_read_oia and hl_copy_data to determine if any location on the screen has been updated. It also
provides a flag that indicates that a structured field has been received from the host and is waiting to be read by the user application. Each unit of time represents 10 milliseconds.

**Input Structure Members**

- **session_id**
  - unique session number

- **mode**
  - 0x01—if polling mode desired
  - 0x02—if blocked mode with timeout desired

- **timeout**
  - 1 - 6000—if read_mode = 2; 0 otherwise

**Returned Structure Members**

- **screen_ret**
  - 0x00—if screen has been updated
  - 0x02—if Session ID is invalid
  - 0x23—if time out value not between 1 and 6000
  - 0x24—if the screen has not been updated
  - 0x25—if Unix has detected an error (error code is in errno)

- **line_change**
  - 44 bytes, one for each line, 0 = unchanged and 1 changed. line_change[0] represents the status line

- **sf_pending**
  - 1—if a structured field is waiting to be read, 0 otherwise

- **write_cmd**
  - 1—if screen updated by Erase Write or Erase Write Alternate command; 0 otherwise

- **partition**
  - reserved

- **filexfr_active**
  - reserved

**Copy Status Line to an Application Data Area**

**Calling Sequence**

```c
void hl_copy_status (struct copy_status *copy_status_struct)
```

**Purpose**

Provides the Silicon Graphics 3270 HLLAPI application with the ability to display and/or analyze all fields of the 3270 status line.
Usage Used in error recovery or when the terminal display program is customer-supplied. If HL_NO_COPY is specified, the target_ptr is not used and it is assumed that stat_pointer from the map data structure will be used to point to the status line. If HL_COPY_TO_APPL is used, target_ptr must be defined.

Input Structure Members

- session_id: unique session number
- mode: HL_NO_COPY and HL_COPY_TO_APPL
- target_ptr: pointer to application data area (size >= 80 bytes)

Returned Structure Members

- copy_ret: 0x00—if the status line has been copied
  0x02—if Session ID is invalid

Start Silicon Graphics Message Mode

Calling Sequence

```c
void hl_start_msg (struct start_msg *start_msg_struct)
```

Purpose Allows the Silicon Graphics 3270 HLLAPI application to start Silicon Graphics Message Mode.

Usage This subroutine must be invoked once prior to calling hl_read_msg or hl_write_msg. (CUT mode only)

Input Structure Members

- session_id: unique session number

Returned Structure Members

- message_ret: 0x00—if successful
  0x02—if Session ID is invalid
Chapter 1: Silicon Graphics' 3270 HLLAPI

Stop Silicon Graphics Message Mode

Calling Sequence

```c
void hl_stop_msg (struct stop_msg *stop_msg_struct)
```

Purpose

Allows the Silicon Graphics 3270 HLLAPI application to terminate Silicon Graphics Message Mode.

Usage

Can be used to switch out of Silicon Graphics Message Mode. It need not be called when disconnecting, since `hl_disconnect` terminates Silicon Graphics Message Mode automatically.

Input Structure Members

- `session_id` unique session number

Returned Structure Members

- `message_ret`
  - 0x00— if successful
  - 0x02— if Session ID is invalid

Read Silicon Graphics Message Mode Protocol Data Units

Calling Sequence

```c
void hl_read_msg (struct read_msg *read_msg_struct)
```

Purpose

Read a Silicon Graphics Message Mode Protocol Data Units sent by the host.

Usage

This subroutine is typically called when the application expects the host to send Message Mode data rather than normal 3270 data. All returned data is encoded in 3270 Buffer Code. It is assumed that each screen is written using the 3270 Erase Write command and that each screen in “unformatted”. Each Set Buffer Address (0,0) is marked by an 0xff character inserted into the data stream. If `HL_NO_COPY` is specified, the `data_ptr` is not used. Instead, the `mba_ptr` from the map data structure is used into point to the 64 Kb Message Mode ring buffer. The map data pointer is initialized by calling `hl_map_data`. The
application must manage its own offset into the ring buffer after initialization. If HL_COPY_TO_APPL is used, data_ptr must be defined.

Input Structure Members

- **session_id**: unique session number
- **mode**: HL_NO_COPY and HL_COPY_TO_APPL
- **data_ptr**: pointer to application data area
- **data_size**: size of the application data area

Returned Structure Members

- **message_ret**: 0x00—if successful
  - 0x02—if Session ID is invalid
  - 0x04—if session not connected for keyboard services
  - 0x21—if the receive time out value has been exceeded
  - 0x26—if the application data size < received data size
  - 0x28—if Silicon Graphics Message Mode has not been invoked
- **data_count**: number of bytes stored in application data area

Read a 3270 Structured Field

Calling Sequence

```c
void hl_read_sf (struct read_sf *read_sf_struct)
```

Purpose

Read a 3270 Structured Field that has been sent by the host.

Usage

This subroutine is called whenever an Silicon Graphics 3270 HLLAPI application expects a Structured Field from the host other than the Read Partition Query and Outbound 3270DS. The application need not call this routine unless it has already detected a structured field pending using the flag word provided by **hl_screen_status**. Each call to **hl_read_sf** returns the contents of a Structured Field. (SDLC)
Chapter 1: Silicon Graphics' 3270 HLLAPI

Input Structure Members

- **session_id**: unique session number
- **data_ptr**: pointer to application data area
- **data_size**: size of the application data area

Returned Structure Members

- **data_count**: number of bytes stored in application data area
- **message_ret**:
  - 0x00—if successful
  - 0x02—if Session ID is invalid
  - 0x04—if session not connected for keyboard services
  - 0x26—if the application data size < received data size

Write 3270 Structured Field to the Host

Calling Sequence

```
void hl_write_sf (struct write_sf *write_sf_struct)
```

Purpose

Send a 3270 Structured Field to the host.

Usage

This subroutine is called whenever the Silicon Graphics 3270 HLLAPI application must send a 3270 Structured Field to the host other than Query Reply and Inbound 3270DS. The Structured Field can contain up to 4K bytes of data. (DFT mode only)

Input Structure Members

- **session_id**: unique session number
- **data_ptr**: pointer to application data area
- **data_count**: number of bytes stored in application data area
Returned Structure Members

message_ret
0x00 if successful
0x02 if session ID is invalid
0x04 if session not connected for keyboard services
0x27 if the application data size > 64K bytes

Trace Silicon Graphics 3270 HLLAPI Data

Calling Sequence
void hl_trace (struct session_id *session_id_struct)

Purpose
Trace keyboard input and host output.

Usage
This routine is called whenever the Silicon Graphics 3270
HLLAPI application wants to trace all data passing through
the Silicon Graphics 3270 HLLAPI. The trace data can be
displayed by invoking /opt/3270/bin/display_3270trace.

Input Structure Members

session_id
unique session number

Returned Structure Members

None

Get Silicon Graphics 3270 HLLAPI Information for a
Specified Session

Calling Sequence
void hl_map_data (struct map_data *map_data_struct)

Purpose
Eliminate need to copy data from Silicon Graphics
Presentation Space image into a local buffer provided by the
Silicon Graphics 3270 HLLAPI application.

Usage
To minimize data copies and allow for different 3270
Presentation Space images, call once for each session at
initialization time.
Chapter 1: Silicon Graphics’ 3270 HLLAPI

Input Structure Members

- `session_id` unique session number
- `ps_ptr` pointer to start of the 3270 Presentation Space
- `stat_ptr` pointer to start of the status line
- `mba_ptr` pointer to start of the Silicon Graphics Message Mode circular buffer
- `ea_ptr` pointer to start of the 3270 Extended Attribute Buffer
- `ps_rows_ptr` pointer to location containing number of rows
- `ps_cols_ptr` pointer to location containing number of columns

Returned Structure Members

- `session_ret` 0x00—if successful

User Entry Point into the IRIS 3270 Emulator

Calling Sequence

```c
void hl_entry_point (struct entry_point *
entry_point_struct)
```

Purpose

Provide an entry point for user-supplied code enhancements to the IRIS 3270 Emulator.

Usage

This routine is called each time through the main processing loop for the IRIS 3270 Emulator. It is designed to provide an inline filter capability for user-specific needs allowing the filter routine to specify when the 3270 Presentation Space should be displayed for user viewing.

Input Structure Members

- `session_id` unique session number
Returned Structure Members

*transfer_ret*

- 0x35—Display the 3270 Presentation Space
- 0x36—Do not display the 3270 Presentation Space
- 0x3c—Poll 3270 Presentation Space for updates
This chapter explains how to gather information and resolve or report problems that might appear when programming the IRIS 3270 Emulator.

The chapter includes information on using the trace utility to resolve HLLAPI errors.

**Troubleshooting Using the Trace Display Utility**

To track errors and more easily troubleshoot problems, you can use 3270 Trace to record all data sent to and received by the 3270 emulation process, then use the Trace Display utility to display the recorded data.

To record the presentation space data, turn on the 3270 Trace option from the *Miscellaneous* box in the Set Up 3270 window, as described in “Selecting Miscellaneous Options,” in Chapter 1 of the *IRIS 3270 Emulator User’s Guide.*

After you use 3270 Trace to record the presentation space, you can display the data using the Trace Display utility by entering:

```
display_3270trace
```

The full-screen window in Figure 2-1 appears.
Figure 2-1  3270 Trace Display Window

Resolving Errors

This section deals with the process for resolving errors. Error messages are listed in Appendix A, “Silicon Graphics HLLAPI Error Messages.”

Error Logging

Errors reported by the IRIS 3270 Emulator are logged in the file /var/opt/3270/file/t3279_log.PID. All errors in this file are also displayed in an error message window. Errors reported by the SGI HLLAPI library are logged in /usr/adm/SYSLOG. These errors are not visually displayed, since it is expected that
each 3270 emulator coded on top of the SGI HLLAPI interface has its own style of user interface.

**Error Information Sources**

For configuration errors, the error information in the log files should be enough. For other errors, additional information can be gathered by setting the 3270 Trace option to Yes using the Set Up 3270 window as described in Chapter 1 of the *IRIS 3270 Emulator User’s Guide*.

**Configuration Errors**

For configuration errors, look up the error message and make the appropriate correction using the Set Up 3270 window. For ASCII terminal users, see Appendix B of the *IRIS 3270 Emulator User’s Guide* for more information on editing the configuration file.

If the error does not go away, refer to your release notes for product support information.

**Nonconfiguration Errors**

To find and correct errors that are not related to configuration problems, perform these steps:

1. Set the 3270 Trace option to Yes using the Set Up 3270 window.
2. Start the terminal emulator from the icon.
3. Re-create the problem.
4. Report the problem. Refer to your release notes for instructions on where to send the files below:
   - all executable files from your `/opt/3270/bin` directory
   - `/var/opt/3270/file/*_log*` (all log files)
5. If the Silicon Graphics 3270 Emulator is terminated by program error, a core file is saved in `/usr/var/opt/spool`. Send this file in addition to those mentioned in step 4.
6. If the problem occurs during a file transfer using IRISXFR, enter \texttt{sgixfr versions} while logged in to your host ID. Report the returned information along with the information collected in steps 1-3.

**SGI HLLAPI Trace Display Interpretation**

If you are a developer working with the SGI HLLAPI, use the Trace Display option in the 3270 icon to display the data contained in the trace file \texttt{/var/opt/3270/file/trace_log}.

These types of information are traced: [BULLET LIST]

- An image of the 3270 Presentation Space. The color code is blue.
- An image of the 3270 Status Line. The color code is white.
- “Line Change Bits.” 1 = changed and 0 = no change. The leftmost bit is for the status line. The other bits are for lines 1 to \( n \) with \( n \) being 24, 27, 32, or 43. The color code is yellow.
- A hexadecimal display of all attributes with their row and column position. Row and column numbers start at 0, not 1. The color code is red.
- Keyboard input. Use Appendix B to interpret the scan codes and shift state representation. The color code is green.
- A hexadecimal display of structured field data read by \texttt{hl_read_sf.c} and written by \texttt{hl_write_sf.c}. The color code for received data is blue and for transmitted data, yellow. The characters are in EBCDIC.
- A hex display of SGI Message Mode data read by \texttt{hl_read_msg.c} and written by \texttt{hl_write_msg.c}. The color code for received data is blue and for transmitted data, yellow. The characters are in 3270 buffer code.
Appendix A

Silicon Graphics HLLAPI Error Messages

The messages listed in this appendix reside in the file /usr/adm/SYSLOG.

These errors are generated by the Silicon Graphics HLLAPI routines described in Chapter 1, “Silicon Graphics’ 3270 HLLAPI.” The messages are divided into two categories: system errors and configuration errors. All system errors should be reported immediately to your SGI service representative. All configuration errors can be eliminated or corrected by changing parameter values in /var/opt/3270/lib/t3279rc.$LOGNAME.

System Errors

In the error list below, \( N \) is the UNIX error number as defined in the intro online man page. The messages are listed below:

If any of these errors occurs, refer to your release notes for product support information.

ERROR: Open of /dev/t3270c0 failed errno = \( N \)
ERROR: Download ioctl call failed errno = \( N \)
ERROR: Open of /var/opt/3270/file/exec.info failed errno = \( N \)
ERROR: Open of /usr/adm/SYSLOG failed errno = \( N \)
ERROR: Open of /var/opt/3270/file/trace_log failed errno = \( N \)
ERROR: Open of 3270 configuration file failed errno = \( N \)
ERROR: Fstat call for /var/opt/3270/lib/t3279uc failed errno = \( N \)
ERROR:Malloc call failed errno = \( N \)
ERROR: Mmap call failed errno = \( N \)
ERROR: Read of /var/opt/3270/lib/t3279uc failed errno = \( N \)
ERROR: SCR_UD ioctl call failed errno = \( N \)
ERROR: SET_DID ioctl call failed errno = \( N \)
Appendix A: Silicon Graphics HLLAPI Error Messages

Configuration Errors

If a configuration error occurs, see Appendix B in the IRIS 3270 Emulator User's Guide for the valid inputs for each configuration parameter. If changing the inputs does not solve the problem, refer to your release notes for product support information.

ERROR: RECV_TIMEOUT must be between 1000 and 65535
ERROR: SEND_TIMEOUT must be between 1 and 2047
ERROR: KEY_TIMEOUT must be between 1 and 31
ERROR: LINK missing or defined improperly
ERROR: CHAR_SET missing or improperly defined
ERROR: MODEL must be between 2 and 5

Errors Reported Using 3270trace Only

Could not get temporary space to read in trace file.
Error detected while reading /var/opt/3270/file/trace_log: errno = N.
Illegal character detected in line number - Please reenter.
Line number too large - Please reenter.
No data in the trace file.
Search string contains zero characters - Please reenter.
### Table B-1 Scan Codes

<table>
<thead>
<tr>
<th>Scancode*</th>
<th>Base</th>
<th>Alt</th>
</tr>
</thead>
<tbody>
<tr>
<td>0B</td>
<td>Erase EOF</td>
<td>Erase Input</td>
</tr>
<tr>
<td>0C</td>
<td>Invalid</td>
<td>Attn</td>
</tr>
<tr>
<td>0D</td>
<td>Tab</td>
<td>Invalid</td>
</tr>
<tr>
<td>0F</td>
<td>PF2</td>
<td>Invalid</td>
</tr>
<tr>
<td>1F</td>
<td>PF4</td>
<td>Invalid</td>
</tr>
<tr>
<td>2F</td>
<td>PF6</td>
<td>Invalid</td>
</tr>
<tr>
<td>3F</td>
<td>PF8</td>
<td>Invalid</td>
</tr>
<tr>
<td>4F</td>
<td>PF10</td>
<td>Invalid</td>
</tr>
<tr>
<td>5A</td>
<td>CR</td>
<td>Invalid</td>
</tr>
<tr>
<td>5E</td>
<td>PF12</td>
<td>Invalid</td>
</tr>
<tr>
<td>5F</td>
<td>PF24</td>
<td>Invalid</td>
</tr>
<tr>
<td>6A</td>
<td>cursor right</td>
<td>Invalid</td>
</tr>
<tr>
<td>6E</td>
<td>PA2</td>
<td>Invalid</td>
</tr>
<tr>
<td>05</td>
<td>Invalid</td>
<td>Sysrq</td>
</tr>
<tr>
<td>06</td>
<td>Clear</td>
<td>Invalid</td>
</tr>
<tr>
<td>07</td>
<td>PF1</td>
<td>Invalid</td>
</tr>
<tr>
<td>08</td>
<td>PF13</td>
<td>Invalid</td>
</tr>
<tr>
<td>10</td>
<td>PF14</td>
<td>Invalid</td>
</tr>
</tbody>
</table>
Appendix B: Scan Code and Buffer Code Tables

Table B-1 (continued)  Scan Codes

<table>
<thead>
<tr>
<th>Scan code&lt;sup&gt;a&lt;/sup&gt;</th>
<th>Base</th>
<th>Alt</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>reset</td>
<td>Invalid</td>
</tr>
<tr>
<td>17</td>
<td>PF3</td>
<td>Invalid</td>
</tr>
<tr>
<td>18</td>
<td>PF15</td>
<td>Invalid</td>
</tr>
<tr>
<td>20</td>
<td>PF16</td>
<td>Invalid</td>
</tr>
<tr>
<td>27</td>
<td>PF5</td>
<td>Invalid</td>
</tr>
<tr>
<td>28</td>
<td>PF17</td>
<td>Invalid</td>
</tr>
<tr>
<td>30</td>
<td>PF18</td>
<td>Invalid</td>
</tr>
<tr>
<td>37</td>
<td>PF7</td>
<td>Invalid</td>
</tr>
<tr>
<td>38</td>
<td>PF19</td>
<td>Invalid</td>
</tr>
<tr>
<td>40</td>
<td>PF20</td>
<td>Invalid</td>
</tr>
<tr>
<td>47</td>
<td>PF9</td>
<td>Invalid</td>
</tr>
<tr>
<td>48</td>
<td>PF21</td>
<td>Invalid</td>
</tr>
<tr>
<td>50</td>
<td>PF22</td>
<td>Invalid</td>
</tr>
<tr>
<td>56</td>
<td>PF11</td>
<td>Invalid</td>
</tr>
<tr>
<td>57</td>
<td>PF23</td>
<td>Invalid</td>
</tr>
<tr>
<td>58</td>
<td>Enter</td>
<td>Invalid</td>
</tr>
<tr>
<td>60</td>
<td>cursor down</td>
<td>Invalid</td>
</tr>
<tr>
<td>61</td>
<td>Backspace</td>
<td>Invalid</td>
</tr>
<tr>
<td>61</td>
<td>cursor left</td>
<td>Invalid</td>
</tr>
<tr>
<td>62</td>
<td>Invalid</td>
<td>home</td>
</tr>
<tr>
<td>63</td>
<td>cursor up</td>
<td>Invalid</td>
</tr>
<tr>
<td>64</td>
<td>Back Tab</td>
<td>Invalid</td>
</tr>
<tr>
<td>65</td>
<td>Insert</td>
<td>Invalid</td>
</tr>
<tr>
<td>66</td>
<td>Delete</td>
<td>Invalid</td>
</tr>
</tbody>
</table>
Table B-1 (continued)  
<table>
<thead>
<tr>
<th>Scancode&lt;sup&gt;a&lt;/sup&gt;</th>
<th>Base</th>
<th>Alt</th>
</tr>
</thead>
<tbody>
<tr>
<td>67</td>
<td>PA1</td>
<td>Invalid</td>
</tr>
<tr>
<td>70</td>
<td>Field Mark</td>
<td>Dev Cncl</td>
</tr>
<tr>
<td>71</td>
<td>Dup</td>
<td>Test</td>
</tr>
<tr>
<td>72</td>
<td>Ident</td>
<td>Invalid</td>
</tr>
<tr>
<td>73</td>
<td>Shift On</td>
<td>Shift Off</td>
</tr>
</tbody>
</table>

<sup>a</sup> All scan codes other than those listed are Invalid. Invalid scan codes are handled as indicated in the Write Keystroke service.

Table B-2  
<table>
<thead>
<tr>
<th>Character Description</th>
<th>ASCII Hex</th>
<th>3270 Device Buffer Hex</th>
<th>Valid Across Languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null</td>
<td>00</td>
<td>00</td>
<td>y</td>
</tr>
<tr>
<td>Invalid</td>
<td>01</td>
<td>01</td>
<td>n</td>
</tr>
<tr>
<td>Invalid</td>
<td>02</td>
<td>02</td>
<td>n</td>
</tr>
<tr>
<td>Invalid</td>
<td>03</td>
<td>03</td>
<td>n</td>
</tr>
<tr>
<td>Invalid</td>
<td>04</td>
<td>04</td>
<td>n</td>
</tr>
<tr>
<td>Invalid</td>
<td>05</td>
<td>05</td>
<td>n</td>
</tr>
<tr>
<td>Invalid</td>
<td>06</td>
<td>06</td>
<td>n</td>
</tr>
<tr>
<td>Invalid</td>
<td>07</td>
<td>07</td>
<td>n</td>
</tr>
<tr>
<td>Greater Than sign</td>
<td>&gt;</td>
<td>3E</td>
<td>08</td>
</tr>
<tr>
<td>Less Than sign</td>
<td>&lt;</td>
<td>3C</td>
<td>09</td>
</tr>
<tr>
<td>Left Bracket</td>
<td>[</td>
<td>5B</td>
<td>0A</td>
</tr>
<tr>
<td>Right Bracket</td>
<td>]</td>
<td>5D</td>
<td>0B</td>
</tr>
<tr>
<td>Right Parenthesis</td>
<td>)</td>
<td>29</td>
<td>0C</td>
</tr>
</tbody>
</table>
### Table B-2 (continued) Buffer Codes

<table>
<thead>
<tr>
<th>Character Description</th>
<th>ASCII Hex</th>
<th>3270 Device Buffer Hex</th>
<th>Valid Across Languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left Parenthesis</td>
<td>(</td>
<td>28 0D</td>
<td>y</td>
</tr>
<tr>
<td>Right Brace</td>
<td>)</td>
<td>7D 0E</td>
<td>n</td>
</tr>
<tr>
<td>Left Brace</td>
<td>[</td>
<td>7B 0F</td>
<td>n</td>
</tr>
<tr>
<td>Blank</td>
<td></td>
<td>20 10</td>
<td>y</td>
</tr>
<tr>
<td>Equal sign</td>
<td>=</td>
<td>3D 11</td>
<td>y</td>
</tr>
<tr>
<td>Apostrophe</td>
<td>'</td>
<td>27 12</td>
<td>y</td>
</tr>
<tr>
<td>Double Quote</td>
<td>&quot;</td>
<td>22 13</td>
<td>n</td>
</tr>
<tr>
<td>Slash</td>
<td>/</td>
<td>2F 14</td>
<td>y</td>
</tr>
<tr>
<td>Reverse Slash</td>
<td>\</td>
<td>5C 15</td>
<td>n</td>
</tr>
<tr>
<td>Vertical Bar</td>
<td></td>
<td></td>
<td>16 11</td>
</tr>
<tr>
<td>Broken Vertical Bar</td>
<td></td>
<td>7C 17</td>
<td>n</td>
</tr>
<tr>
<td>Question Mark</td>
<td>?</td>
<td>3F 18</td>
<td>y</td>
</tr>
<tr>
<td>Exclamation Mark</td>
<td>!</td>
<td>21 19</td>
<td>n</td>
</tr>
<tr>
<td>Dollar sign</td>
<td>$</td>
<td>24 1A</td>
<td>n</td>
</tr>
<tr>
<td>Cent Sign</td>
<td>¢</td>
<td>9B 1B</td>
<td>n</td>
</tr>
<tr>
<td>Lb. sign - Lire</td>
<td>£</td>
<td>9C 1C</td>
<td>n</td>
</tr>
<tr>
<td>Yen sign</td>
<td>¥</td>
<td>9D 1D</td>
<td>n</td>
</tr>
<tr>
<td>Peseta</td>
<td>•</td>
<td>9E 1E</td>
<td>n</td>
</tr>
<tr>
<td>Intl. Monetary sign</td>
<td>□</td>
<td>0F 1F</td>
<td>n</td>
</tr>
<tr>
<td>0 (F0)</td>
<td>0</td>
<td>30 20</td>
<td>y</td>
</tr>
<tr>
<td>1 (F1)</td>
<td>1</td>
<td>31 21</td>
<td>21</td>
</tr>
<tr>
<td>2 (F2)</td>
<td>2</td>
<td>32 22</td>
<td>y</td>
</tr>
<tr>
<td>3 (F3)</td>
<td>3</td>
<td>33 23</td>
<td>y</td>
</tr>
<tr>
<td>Character Description</td>
<td>ASCII Hex</td>
<td>3270 Device Buffer Hex</td>
<td>Valid Across Languages</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------</td>
<td>------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>4 (F4)</td>
<td>4</td>
<td>34</td>
<td>24</td>
</tr>
<tr>
<td>5 (F5)</td>
<td>5</td>
<td>35</td>
<td>25</td>
</tr>
<tr>
<td>6 (F6)</td>
<td>6</td>
<td>36</td>
<td>26</td>
</tr>
<tr>
<td>7 (F7)</td>
<td>7</td>
<td>37</td>
<td>27</td>
</tr>
<tr>
<td>8 (F8)</td>
<td>8</td>
<td>38</td>
<td>28</td>
</tr>
<tr>
<td>9 (F9)</td>
<td>9</td>
<td>39</td>
<td>29</td>
</tr>
<tr>
<td>Beta</td>
<td>ß</td>
<td>F1</td>
<td>2A</td>
</tr>
<tr>
<td>Section Symbol</td>
<td>§</td>
<td>15</td>
<td>2B</td>
</tr>
<tr>
<td>Number sign</td>
<td>#</td>
<td>23</td>
<td>2C</td>
</tr>
<tr>
<td>At sign</td>
<td>@</td>
<td>40</td>
<td>2D</td>
</tr>
<tr>
<td>Percent sign</td>
<td>%</td>
<td>25</td>
<td>2E</td>
</tr>
<tr>
<td>Underscore</td>
<td>_</td>
<td>5F</td>
<td>2F</td>
</tr>
<tr>
<td>Ampersand</td>
<td>&amp;</td>
<td>26</td>
<td>30</td>
</tr>
<tr>
<td>Hyphen</td>
<td>-</td>
<td>2D</td>
<td>31</td>
</tr>
<tr>
<td>Period</td>
<td>.</td>
<td>2E</td>
<td>32</td>
</tr>
<tr>
<td>Comma</td>
<td>,</td>
<td>2C</td>
<td>33</td>
</tr>
<tr>
<td>Colon</td>
<td>:</td>
<td>3A</td>
<td>34</td>
</tr>
<tr>
<td>Plus sign</td>
<td>+</td>
<td>2B</td>
<td>35</td>
</tr>
<tr>
<td>Not Symbol</td>
<td>¬</td>
<td>AA</td>
<td>36</td>
</tr>
<tr>
<td>Overbar</td>
<td>Invalid</td>
<td>37</td>
<td>n</td>
</tr>
<tr>
<td>Degree</td>
<td>Invalid</td>
<td>38</td>
<td>n</td>
</tr>
<tr>
<td>Circumflex</td>
<td>^</td>
<td>5E</td>
<td>3A</td>
</tr>
<tr>
<td>Character Description</td>
<td>ASCII Hex</td>
<td>3270 Device Buffer Hex</td>
<td>Valid Across Languages</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------</td>
<td>------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Tilde</td>
<td>~</td>
<td>7E</td>
<td>3B</td>
</tr>
<tr>
<td>Diaeresis</td>
<td>“</td>
<td>Invalid</td>
<td>3C</td>
</tr>
<tr>
<td>Grave Accent</td>
<td>‘</td>
<td>60</td>
<td>3D</td>
</tr>
<tr>
<td>Acute Accent</td>
<td>‘</td>
<td>27</td>
<td>3E</td>
</tr>
<tr>
<td>Cedilla</td>
<td>ç</td>
<td>Invalid</td>
<td>3F</td>
</tr>
<tr>
<td>a Grave Small</td>
<td>à</td>
<td>85</td>
<td>40</td>
</tr>
<tr>
<td>e Grave Small</td>
<td>è</td>
<td>8A</td>
<td>41</td>
</tr>
<tr>
<td>i Grave Small</td>
<td>ì</td>
<td>8D</td>
<td>42</td>
</tr>
<tr>
<td>o Grave Small</td>
<td>ò</td>
<td>95</td>
<td>43</td>
</tr>
<tr>
<td>u Grave Small</td>
<td>û</td>
<td>97</td>
<td>44</td>
</tr>
<tr>
<td>a Tilde Small</td>
<td>ä</td>
<td>Invalid</td>
<td>45</td>
</tr>
<tr>
<td>o Tilde Small</td>
<td>ô</td>
<td>Invalid</td>
<td>46</td>
</tr>
<tr>
<td>y Diaeresis Small</td>
<td>ý</td>
<td>98</td>
<td>Invalid</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Invalid</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Invalid</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Invalid</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Invalid</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Invalid</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Invalid</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Invalid</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Invalid</td>
</tr>
<tr>
<td>a Diaeresis Small</td>
<td>ä</td>
<td>84</td>
<td>50</td>
</tr>
<tr>
<td>e Diaeresis Small</td>
<td>è</td>
<td>89</td>
<td>51</td>
</tr>
</tbody>
</table>
Table B-2 (continued) Buffer Codes

<table>
<thead>
<tr>
<th>Character Description</th>
<th>ASCII Hex</th>
<th>3270 Device Buffer Hex</th>
<th>Valid Across Languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>i Diaeresis Small</td>
<td>ï</td>
<td>8B 52</td>
<td>n</td>
</tr>
<tr>
<td>o Diaeresis Small</td>
<td>ö</td>
<td>94 53</td>
<td>n</td>
</tr>
<tr>
<td>u Diaeresis Small</td>
<td>ü</td>
<td>81 54</td>
<td>n</td>
</tr>
<tr>
<td>a Circumflex Small</td>
<td>á</td>
<td>83 55</td>
<td>n</td>
</tr>
<tr>
<td>e Circumflex Small</td>
<td>è</td>
<td>88 56</td>
<td>n</td>
</tr>
<tr>
<td>i Circumflex Small</td>
<td>ì</td>
<td>8C 57</td>
<td>n</td>
</tr>
<tr>
<td>o Circumflex Small</td>
<td>õ</td>
<td>93 58</td>
<td>n</td>
</tr>
<tr>
<td>u Circumflex Small</td>
<td>ú</td>
<td>96 59</td>
<td>n</td>
</tr>
<tr>
<td>a Acute Small</td>
<td>á</td>
<td>A0 5A</td>
<td>n</td>
</tr>
<tr>
<td>e Acute Small</td>
<td>é</td>
<td>82 5B</td>
<td>n</td>
</tr>
<tr>
<td>i Acute Small</td>
<td>í</td>
<td>A1 5C</td>
<td>n</td>
</tr>
<tr>
<td>o Acute Small</td>
<td>ö</td>
<td>A2 5D</td>
<td>n</td>
</tr>
<tr>
<td>u Acute Small</td>
<td>ú</td>
<td>A3 5E</td>
<td>n</td>
</tr>
<tr>
<td>n Tilde Small</td>
<td>à</td>
<td>A4 5F</td>
<td>n</td>
</tr>
<tr>
<td>a Grave Capital</td>
<td>Invalid</td>
<td>60</td>
<td>n</td>
</tr>
<tr>
<td>e Grave Capital</td>
<td>Invalid</td>
<td>61</td>
<td>n</td>
</tr>
<tr>
<td>i Grave Capital</td>
<td>Invalid</td>
<td>62</td>
<td>n</td>
</tr>
<tr>
<td>o Grave Capital</td>
<td>Invalid</td>
<td>63</td>
<td>n</td>
</tr>
<tr>
<td>u Grave Capital</td>
<td>Invalid</td>
<td>64</td>
<td>n</td>
</tr>
<tr>
<td>a Tilde Capital</td>
<td>Invalid</td>
<td>65</td>
<td>n</td>
</tr>
<tr>
<td>e Tilde Capital</td>
<td>Invalid</td>
<td>66</td>
<td>n</td>
</tr>
<tr>
<td></td>
<td>Invalid</td>
<td>67</td>
<td>n</td>
</tr>
<tr>
<td></td>
<td>Invalid</td>
<td>68</td>
<td>n</td>
</tr>
<tr>
<td></td>
<td>Invalid</td>
<td>69</td>
<td>n</td>
</tr>
</tbody>
</table>
# Appendix B: Scan Code and Buffer Code Tables

## Table B-2 (continued) Buffer Codes

<table>
<thead>
<tr>
<th>Character Description</th>
<th>ASCII Hex</th>
<th>3270 Device Buffer Hex</th>
<th>Valid Across Languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invalid</td>
<td>6A</td>
<td></td>
<td>n</td>
</tr>
<tr>
<td>Invalid</td>
<td>6B</td>
<td></td>
<td>n</td>
</tr>
<tr>
<td>Invalid</td>
<td>6C</td>
<td></td>
<td>n</td>
</tr>
<tr>
<td>Invalid</td>
<td>6D</td>
<td></td>
<td>n</td>
</tr>
<tr>
<td>Invalid</td>
<td>6E</td>
<td></td>
<td>n</td>
</tr>
<tr>
<td>Invalid</td>
<td>6F</td>
<td></td>
<td>n</td>
</tr>
<tr>
<td>a Diaeresis Capital</td>
<td>Ä</td>
<td>8E</td>
<td>70</td>
</tr>
<tr>
<td>e Diaeresis Capital</td>
<td>Invalid</td>
<td>71</td>
<td></td>
</tr>
<tr>
<td>i Diaeresis Capital</td>
<td>Invalid</td>
<td>72</td>
<td></td>
</tr>
<tr>
<td>o Diaeresis Capital</td>
<td>Ö</td>
<td>99</td>
<td>73</td>
</tr>
<tr>
<td>u Diaeresis Capital</td>
<td>Ü</td>
<td>9A</td>
<td>74</td>
</tr>
<tr>
<td>a Circumflex Capital</td>
<td>Invalid</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>e Circumflex Capital</td>
<td>Invalid</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td>i Circumflex Capital</td>
<td>Invalid</td>
<td>77</td>
<td></td>
</tr>
<tr>
<td>o Circumflex Capital</td>
<td>Invalid</td>
<td>78</td>
<td></td>
</tr>
<tr>
<td>u Circumflex Capital</td>
<td>Invalid</td>
<td>79</td>
<td></td>
</tr>
<tr>
<td>a Acute Capital</td>
<td>Invalid</td>
<td>7A</td>
<td></td>
</tr>
<tr>
<td>e Acute Capital</td>
<td>É</td>
<td>90</td>
<td>7B</td>
</tr>
<tr>
<td>i Acute Capital</td>
<td>Invalid</td>
<td>7C</td>
<td></td>
</tr>
<tr>
<td>o Acute Capital</td>
<td>Invalid</td>
<td>7D</td>
<td></td>
</tr>
<tr>
<td>u Acute Capital</td>
<td>Invalid</td>
<td>7E</td>
<td></td>
</tr>
<tr>
<td>n Tilde Capital</td>
<td>Ñ</td>
<td>A5</td>
<td>7F</td>
</tr>
<tr>
<td>a Small</td>
<td>a</td>
<td>61</td>
<td>80</td>
</tr>
<tr>
<td>Character Description</td>
<td>ASCII Hex</td>
<td>3270 Device Buffer Hex</td>
<td>Valid Across Languages</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------</td>
<td>------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>b Small</td>
<td>b</td>
<td>62</td>
<td>81</td>
</tr>
<tr>
<td>c Small</td>
<td>c</td>
<td>63</td>
<td>82</td>
</tr>
<tr>
<td>d Small</td>
<td>d</td>
<td>64</td>
<td>83</td>
</tr>
<tr>
<td>e Small</td>
<td>e</td>
<td>65</td>
<td>84</td>
</tr>
<tr>
<td>f Small</td>
<td>f</td>
<td>66</td>
<td>85</td>
</tr>
<tr>
<td>g Small</td>
<td>g</td>
<td>67</td>
<td>86</td>
</tr>
<tr>
<td>h Small</td>
<td>h</td>
<td>68</td>
<td>87</td>
</tr>
<tr>
<td>i Small</td>
<td>i</td>
<td>69</td>
<td>88</td>
</tr>
<tr>
<td>j Small</td>
<td>j</td>
<td>6A</td>
<td>89</td>
</tr>
<tr>
<td>k Small</td>
<td>k</td>
<td>6B</td>
<td>8A</td>
</tr>
<tr>
<td>l Small</td>
<td>l</td>
<td>6C</td>
<td>8B</td>
</tr>
<tr>
<td>m Small</td>
<td>m</td>
<td>6D</td>
<td>8C</td>
</tr>
<tr>
<td>n Small</td>
<td>n</td>
<td>6E</td>
<td>8D</td>
</tr>
<tr>
<td>o Small</td>
<td>o</td>
<td>6F</td>
<td>8E</td>
</tr>
<tr>
<td>p Small</td>
<td>p</td>
<td>70</td>
<td>8F</td>
</tr>
<tr>
<td>q Small</td>
<td>q</td>
<td>71</td>
<td>90</td>
</tr>
<tr>
<td>r Small</td>
<td>r</td>
<td>72</td>
<td>91</td>
</tr>
<tr>
<td>s Small</td>
<td>s</td>
<td>73</td>
<td>92</td>
</tr>
<tr>
<td>t Small</td>
<td>t</td>
<td>74</td>
<td>93</td>
</tr>
<tr>
<td>u Small</td>
<td>u</td>
<td>75</td>
<td>94</td>
</tr>
<tr>
<td>v Small</td>
<td>v</td>
<td>76</td>
<td>95</td>
</tr>
<tr>
<td>w Small</td>
<td>w</td>
<td>77</td>
<td>96</td>
</tr>
<tr>
<td>x Small</td>
<td>x</td>
<td>78</td>
<td>97</td>
</tr>
</tbody>
</table>
### Table B-2 (continued) Buffer Codes

<table>
<thead>
<tr>
<th>Character Description</th>
<th>ASCII Hex</th>
<th>3270 Device Buffer Hex</th>
<th>Valid Across Languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>y Small</td>
<td>y</td>
<td>79</td>
<td>98</td>
</tr>
<tr>
<td>z Small</td>
<td>z</td>
<td>7A</td>
<td>99</td>
</tr>
<tr>
<td>Diphthong Small</td>
<td>æ</td>
<td>91</td>
<td>9A</td>
</tr>
<tr>
<td>o Slash Small</td>
<td>Invalid</td>
<td>9B</td>
<td>n</td>
</tr>
<tr>
<td>a Overcircle Small</td>
<td>86</td>
<td>9C</td>
<td>n</td>
</tr>
<tr>
<td>c Cedilla Small</td>
<td>Invalid</td>
<td>9E</td>
<td>y</td>
</tr>
<tr>
<td></td>
<td>Invalid</td>
<td>9F</td>
<td>y</td>
</tr>
<tr>
<td>a Capital</td>
<td>A</td>
<td>41</td>
<td>A0</td>
</tr>
<tr>
<td>b Capital</td>
<td>B</td>
<td>42</td>
<td>A1</td>
</tr>
<tr>
<td>c Capital</td>
<td>C</td>
<td>43</td>
<td>A2</td>
</tr>
<tr>
<td>d Capital</td>
<td>D</td>
<td>44</td>
<td>A3</td>
</tr>
<tr>
<td>e Capital</td>
<td>E</td>
<td>45</td>
<td>A4</td>
</tr>
<tr>
<td>f Capital</td>
<td>F</td>
<td>46</td>
<td>A5</td>
</tr>
<tr>
<td>g Capital</td>
<td>G</td>
<td>47</td>
<td>A6</td>
</tr>
<tr>
<td>h Capital</td>
<td>H</td>
<td>48</td>
<td>A7</td>
</tr>
<tr>
<td>i Capital</td>
<td>I</td>
<td>49</td>
<td>A8</td>
</tr>
<tr>
<td>j Capital</td>
<td>J</td>
<td>4A</td>
<td>A9</td>
</tr>
<tr>
<td>k Capital</td>
<td>K</td>
<td>4B</td>
<td>AA</td>
</tr>
<tr>
<td>l Capital</td>
<td>L</td>
<td>4C</td>
<td>AB</td>
</tr>
<tr>
<td>m Capital</td>
<td>M</td>
<td>4D</td>
<td>AC</td>
</tr>
<tr>
<td>n Capital</td>
<td>N</td>
<td>4E</td>
<td>AD</td>
</tr>
<tr>
<td>o Capital</td>
<td>O</td>
<td>4F</td>
<td>AE</td>
</tr>
<tr>
<td>Character Description</td>
<td>ASCII Hex</td>
<td>3270 Device Buffer Hex</td>
<td>Valid Across Languages</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------</td>
<td>------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>p Capital</td>
<td>P</td>
<td>50</td>
<td>AF</td>
</tr>
<tr>
<td>q Capital</td>
<td>Q</td>
<td>51</td>
<td>B0</td>
</tr>
<tr>
<td>r Capital</td>
<td>R</td>
<td>52</td>
<td>B1</td>
</tr>
<tr>
<td>s Capital</td>
<td>S</td>
<td>53</td>
<td>B2</td>
</tr>
<tr>
<td>t Capital</td>
<td>T</td>
<td>54</td>
<td>B3</td>
</tr>
<tr>
<td>u Capital ;</td>
<td>U</td>
<td>55</td>
<td>B4</td>
</tr>
<tr>
<td>v Capital</td>
<td>V</td>
<td>56</td>
<td>B5</td>
</tr>
<tr>
<td>w Capital</td>
<td>W</td>
<td>57</td>
<td>B6</td>
</tr>
<tr>
<td>x Capital</td>
<td>X</td>
<td>58</td>
<td>B7</td>
</tr>
<tr>
<td>y Capital</td>
<td>Y</td>
<td>59</td>
<td>B8</td>
</tr>
<tr>
<td>z Capital</td>
<td>Z</td>
<td>5A</td>
<td>B9</td>
</tr>
<tr>
<td>Diphthong Capital</td>
<td>Æ</td>
<td>92</td>
<td>BA</td>
</tr>
<tr>
<td>O Slash Capital</td>
<td>Invalid</td>
<td></td>
<td>BB</td>
</tr>
<tr>
<td>a Overcircle Capital</td>
<td>8F</td>
<td>BC</td>
<td>n</td>
</tr>
<tr>
<td>c Cedilla Capital</td>
<td>80</td>
<td>BD</td>
<td>n</td>
</tr>
<tr>
<td>; Semicolon</td>
<td>3B</td>
<td>BE</td>
<td>y</td>
</tr>
<tr>
<td>* Asterisk</td>
<td>2A</td>
<td>BF</td>
<td>y</td>
</tr>
</tbody>
</table>
Index

B
buffer codes, 29

D
display_3270trace, 23

E
error messages
  3270trace, 28
  HLLAPI configuration errors, 28
  HLLAPI system errors, 27

H
hl_connect, 7
hl_copy_data, 11
hl_copy_status, 14
hl_disconnect, 11
hl_entry_point, 20
hl_init, 5
hl_map_data, 19
hl_query_session, 6
hl_query_session_cursor, 7
hl_query_session_id, 4
hl_read_msg, 16
hl_read_oiag, 13
hl_read_sf, 17
hl_release, 10
hl_reserve, 8
hl_screen_status, 13
hl_send_key, 9
hl_start_msg, 15
hl_stop_msg, 16
hl_trace, 19
hl_write sf, 18
HLLAPI, 26
  capabilities, 3
  features, 2
  subroutines, 4
HLLAPI subroutines
  hl_query_session_id, 4

I
IRIS 3270 Emulator
  introduction, xi
  network configuration, xiv
  product support, xiv
  software and hardware requirements, xii

P
presentation space, 23
product support, xiv
Index

R

resolving errors, 24

T

trace display, 23
trace display option, 26
troubleshooting configuration errors, 25
troubleshooting nonconfiguration errors, 25
troubleshooting using the trace display utility, 23
Tell Us About This Manual

As a user of Silicon Graphics products, you can help us to better understand your needs and to improve the quality of our documentation.

Any information that you provide will be useful. Here is a list of suggested topics:

- General impression of the document
- Omission of material that you expected to find
- Technical errors
- Relevance of the material to the job you had to do
- Quality of the printing and binding

Please send the title and part number of the document with your comments. The part number for this document is 007-1410-020.

Thank you!

Three Ways to Reach Us

- To send your comments by electronic mail, use either of these addresses:
  - On the Internet: techpubs@sgi.com
  - For UUCP mail (through any backbone site): [your_site]!sgi!techpubs
- To fax your comments (or annotated copies of manual pages), use this fax number: 650-965-0964
- To send your comments by traditional mail, use this address:
  
  Technical Publications
  Silicon Graphics, Inc.
  2011 North Shoreline Boulevard, M/S 535
  Mountain View, California 94043-1389