Appendix A

Silicon Graphics Driver/Kernel API

This appendix summarizes the Silicon Graphics Driver/Kernel Authorized Programming Interface in tabular form. The data structures, entry points, and kernel functions are listed alphabetically with cross-references to the pages where they are discussed. The tables also show which functions and structures are compatible with SVR4 and which are unique to IRIX.

The tables in this appendix are based on the reference pages in volume D. The reference pages in volume D constitute the formal, engineering definition of the Driver/Kernel API. When discussion in this book disagrees with the contents of a reference page, the reference page takes precedence (however, any such disagreement should be reported by email to techpubs@sgi.com).

- "Driver Exported Names" on page 564 tabulates the names of data and functions that a driver must export.
- "Kernel Data Structures and Declarations" on page 565 tabulates the objects used in the interface.
- "Kernel Functions" on page 566 tabulates the IRIX kernel services used by drivers.

Each table in this appendix has a column headed "Versions." The codes in this column have the following meanings:

SV	Syntactically and semantically portable from SVR4 UNIX, as documented in the *UNIX SVR4.2 Device Driver Reference*.
SV*	Syntactically portable from UNIX SVR4, but semantics may differ. Read the discussion and reference page carefully when porting.
5.3	Portable from IRIX version 5.3.
5.3*	Portable from IRIX 5.3, but interface has changed in some detail or new ability has been added.
6.2	Introduced in IRIX version 6.2.

Driver Exported Names

A kernel driver is required to export certain names of static data and functional entry points. These exported names are summarized in Table A-1.

Table A-1 Driver Exported Names

Name	Summary	Discussed	Versions
close(D2)	Notify driver of final close of minor device.	page 149	SV, 5.3
devflag(D1)	Show driver attributes to *lboot*.	page 140	SV*, 5.3*
halt(D2)	Notify driver of system shutdown.	page 168	SV, 5.3
info(D1)	Show driver entries to STREAMS interface.	page 542	SV, 5.3
init(D2)	Initialize driver early in system startup.	page 144	SV*, 5.3
intr(D2)	Notify driver of device interrupt.	page 164	SV, 5.3
ioctl(D2)	Call driver to implement ioctl() call.	page 150	SV*, 5.3
map(D2)	Call driver to implement IRIX mmap().	page 159	5.3
mmap(D2)	Call driver to implement mmap().	page 161	SV*, 5.3
open(D2)	Call driver to open a device.	page 146	SV, 5.3
print(D2)	Call block driver to display filesystem error.	page 169	SV, 5.3
put(D2)	Call STREAMS driver to receive message.	page 544	SV, 5.3
read(D2)	Call character driver to read data.	page 152	SV, 5.3
size(D2)	Call block driver to get device capacity.	page 169	SV, 5.3
srv(D2)	Call driver to service queued messages.	page 545	SV, 5.3
start(D2)	Initialize driver late in system startup.	page 145	SV, 5.3
strategy(D2)	Call block driver to read or write data.	page 154	SV*, 5.3
unload(D2)	Call loadable driver prior to unloading it.	page 167	5.3
unmap(D2)	Call driver to notify it of unmap() call.	page 162	5.3
write(D2)	Call character driver to write data.	page 152	SV, 5.3

The following reference pages have overview information on exported names: intro(D1), intro(D2), and prefix(D1).

Note: The following SVR4 exported names are not used in IRIX drivers: chpoll, _load, and _unload. The latter is replaced by *pfx***load()** without the leading underscore.

Kernel Data Structures and Declarations

The driver/kernel interface is based on shared use of certain data types and defined constant values. For general information on these interface objects, see the intro(D4) and intro(D5) reference pages.

The interface objects used by device drivers are summarized in Table A-2. .

Table A-2 Device Driver Interface Objects

Name	Summary	Discussed	Versions
buf(D4)	Block read/write request structure.	page 183	SV*, 5.3*
eisa_dma_cb(D4)	DMA command block for EISA slave DMA.	page 446	5.3
eisa_dma_buf(D4)	DMA command buffer for EISA slave DMA.	page 446	5.3
errnos(D5)	Error numbers valid for driver use.		SV*, 5.3
iovec(D4)	Describes an I/O buffer segment to the read or write entry points.	page 182	SV, 5.3
signals(D5)	Lists signal numbers valid for driver use.		SV*, 5.3
uio(D4)	Describes an I/O request to the read or write entry points.	page 182	SV*, 5.3

Note: The following data structures used in SVR4 drivers are not used in IRIX: *dma_buf* and *dma_cb*. The *eisa_dma_buf* and *eisa_dma_cb* structures are similar but are used only in EISA drivers.

The interface objects used by STREAMS drivers are summarized in Table A-3

Table A-3 STREAMS Driver Interface Objects

Name	Summary	Discussed	Versions
copyreq(D4)	Copy request structure.		SV, 5.3
copyresp(D4)	Copy response structure.		SV, 5.3
datab(D4)	Message data block.		SV, 5.3
free_rtn(D4)	Describes a message-free routine.		SV, 5.3
iocblk(D4)	Describes ioctl() data or response.		SV, 5.3
linkblk(D4)	Describes multiplexed link.		SV, 5.3
module_info(D4)	Describes module attributes.		SV, 5.3
msgb(D4)	Describes all or part of a message.		SV, 5.3
qinit(D4)	Points to handlers and parameters for a queue.		SV, 5.3
queue(D4)	Describes a queue of messages.		SV, 5.3
streamtab(D4)	Points to the queues handled by a driver.		SV, 5.3
stroptions(D4)	Lists stream-head options.		SV, 5.3

Kernel Functions

The IRIX kernel makes available the Table A-4 functions summarized in .

Table A-4 Kernel Functions

Name	Summary	Discussed	Versions
adjmsg(D3)	Trim bytes from a message.		SV, 5.3
allocb(D3)	Allocate a message block.		SV, 5.3
ASSERT(D3)	Debugging macro designed for use in the kernel (compare to assert(3X)).	page 252	5.3
badaddr(D3)	Test physical address for input.	page 195	5.3

Table A-4 (continued) Kernel Functions

Name	Summary	Discussed	Versions
badaddr_val(D3)	Test physical address for input and return the input value received.	page 195	6.2
bcanput(D3)	Test for flow control in a specified priority band.		SV, 5.3
bcanputnext(D3)	Test for flow control in a specified priority band.		SV, 5.3
bcmp(D3)	Compare data between kernel locations.	page 193	SV, 5.3
bcopy(D3)	Copy data between locations in the kernel.	page 193	SV, 5.3
biodone(D3)	Mark a *buf_t* as complete and wake any process waiting for it.	page 217	SV, 5.3
bioerror(D3)	Manipulate error fields within a *buf_t*.	page 217	SV, 5.3
biowait(D3)	Suspend process pending completion of block I/O.	page 217	SV, 5.3
bp_mapin(D3)	Map buffer pages into kernel virtual address space.	page 199	SV, 5.3
bp_mapout(D3)	Release mapping of buffer pages.	page 199	SV, 5.3
bptophys(D3)	Get physical address of buffer data.	page 198	5.3
brelse(D3)	Return a buffer to the system's free list.	page 190	SV, 5.3
btod(D3)	Return number of 512-byte "sectors" in a byte count (round up).	page 197	5.3
btop(D3)	Return number of I/O pages in a byte count (truncate).	page 197	SV, 5.3
btopr(D3)	Return number of I/O pages in a byte count (round up).	page 197	SV, 5.3
bufcall(D3)	Call a function when a buffer becomes available.		SV, 5.3
bzero(D3)	Clear kernel memory for a specified size.	page 192	SV, 5.3
canput(D3)	Test for room in a message queue.		SV, 5.3

Table A-4 (continued) Kernel Functions

Name	Summary	Discussed	Versions
canputnext(D3)	Test for room in a message queue.		SV, 5.3
clrbuf(D3)	Erase the contents of a buffer desribed by a buf_t.	page 199	SV, 5.3
cmn_err(D3)	Display an error message or panic the system.	page 249	SV*, 5.3
copyb(D3)	Copy a message block.		SV, 5.3
copyin(D3)	Copy data from user address space.	page 192	SV, 5.3
copymsg(D3)	Copy a message.		SV, 5.3
copyout(D3)	Copy data to user address space.	page 192	SV, 5.3
cpsema(D3)	Conditionally decrement a semaphore's state.	page 222	5.3
cvsema(D3)	Conditionally increment a semaphore's state	page 222	5.3
datamsg(D3)	Test whether a message is a data message.		SV, 5.3
delay(D3)	Delay for a specified number of clock ticks.	page 214	SV, 5.3
disable_sysad_parity()	Disable memory parity checking on SysAD bus.	page 526	
dki_dcache_inval(D3)	Invalidate the data cache for a given range of virtual addresses.	page 200	5.3
dki_dcache_wb(D3)	Write back the data cache for a given range of virtual addresses.	page 200	5.3
dki_dcache_wbinval(D3)	Write back and invalidate the data cache for a given range of virtual addresses.	page 200	5.3
dma_map(D3)	Load DMA mapping registers for an imminent transfer.	page 330	5.3
dma_mapbp(D3)	Load DMA mapping registers for an imminent transfer.	page 330	5.3

Table A-4 (continued) Kernel Functions

Name	Summary	Discussed	Versions
dma_mapaddr(D3)	Return the “bus virtual” address for a given map and address.	page 330	5.3
dma_mapalloc(D3)	Allocate a DMA map.	page 330	5.3
dma_mapfree(D3)	Free a DMA map.	page 330	5.3
drv_getparm(D3)	Retrieve kernel state information.	page 202	SV*, 5.3
drv_hztousec(D3)	Convert clock ticks to microseconds	page 214	SV, 5.3
drv_priv(D3)	Test for privileged user.	page 202	SV, 5.3
drv_setparm(D3)	Set kernel state information.	page 202	SV, 5.3
drv_usectohz(D3)	Convert microseconds to clock ticks.	page 214	SV, 5.3
drv_usecwait(D3)	Busy-wait for a specified interval.	page 214	SV, 5.3
dtimeout(D3)	Schedule a function execute on a specified processor after a specified length of time.	page 214	5.3
dupb(D3)	Duplicate a message block.		SV, 5.3
dupmsg(D3)	Duplicate a message.		SV, 5.3
eisa_dma_disable(D3)	Disable recognition of hardware requests on a DMA channel.	page 446	5.3
eisa_dma_enable(D3)	Enable recognition of hardware requests on a DMA channel.	page 446	5.3
eisa_dma_free_buf(D3)	Free a previously allocated DMA buffer descriptor.	page 446	5.3
eisa_dma_free_cb(D3)	Free a previously allocated DMA command block.	page 446	5.3
eisa_dma_get_buf(D3)	Allocate a DMA buffer descriptor.	page 446	5.3
eisa_dma_get_cb(D3)	Allocate a DMA command block.	page 446	5.3
eisa_dma_prog(D3)	Program a DMA operation for a subsequent software request.	page 446	5.3

Table A-4 (continued) Kernel Functions

Name	Summary	Discussed	Versions
eisa_dma_stop(D3)	Stop software-initiated DMA operation and release channel.	page 446	5.3
eisa_dma_swstart(D3)	Initiate a DMA operation via software request.	page 446	5.3
eisa_dmachan_alloc()	Allocate a DMA channel for EISA slave DMA.	page 444	5.3
eisa_ivec_alloc()	Allocate an IRQ level for EISA.	page 442	5.3
eisa_ivec_set()	Associate a handler with an EISA IRQ.	page 442	5.3
enableok(D3)	Allow a queue to be serviced.		SV, 5.3
enable_sysad_parity()	Reenable parity checking on SysAD bus.	page 526	
esballoc(D3)	Allocate a message block using an externally-supplied buffer.		SV, 5.3
esbbcall(D3)	Call a function when an externally-supplied buffer can be allocated.		SV, 5.3
etoimajor(D3)	Convert external to internal major device number.	page 181	SV, 5.3
fast_itimeout(D3)	Same as itimeout() but takes an interval in "fast ticks."	page 214	6.2
fasthzto(D3)	Returns the value of a *struct timeval* as a count of "fast ticks."	page 214	6.2
flushband(D3)	Flush messages in a specified priority band.		SV, 5.3
flushbus(D3)	Make sure contents of the write buffer are flushed to the system bus	page 200	5.3
flushq(D3)	Flush messages on a queue.		SV, 5.3
freeb(D3)	Free a message block.		SV, 5.3
freemsg(D3)	Free a message.		SV, 5.3
freerbuf(D3)	Free a buf_t with no buffer.	page 190	SV, 5.3

Table A-4 (continued) Kernel Functions

Name	**Summary**	**Discussed**	**Versions**
freesema(D3)	Free the resources associated with a semaphore.	page 222	5.3*
freezestr(D3)	Freeze the state of a stream.		SV, 5.3
fubyte(D3)	Load a byte from user space.	page 192	5.3
fuword(D3)	Load a word from user space.	page 192	5.3
geteblk(D3)	Get a buf_t with no buffer.	page 190	SV, 5.3
getemajor(D3)	Get external major device number.	page 181	SV, 5.3
geteminor(D3)	Get external minor device number.	page 181	SV, 5.3
geterror(D3)	retrieve error number from a buffer header	page 217	SV, 5.3
getmajor(D3)	Get internal major device number.	page 181	SV, 5.3
getminor(D3)	Get internal minor device number.	page 181	SV, 5.3
getnextpg(D3)	Return *pfdat* structure for next page.	page 199	5.3
getq(D3)	Get the next message from a queue.		SV, 5.3
getrbuf(D3)	Allocate a *buf_t* with no buffer.	page 190	SV, 5.3
hwcpin(D3)	Copy data from device registers to kernel memory.	page 192	5.3
hwcpout(D3)	Copy data from kernel memory to device registers.	page 192	5.3
initnsema(D3)	Initialize a semaphore to a specified count.	page 222	5.3
initnsema_mutex(D3)	Initialize a semaphore to a count of 1.	page 222	5.3
insq(D3)	Insert a message into a queue.		SV, 5.3
ip26_enable_ucmem(D3)	Change memory mode on IP26 processor.	page 27	6.2
ip26_return_ucmem(D3)	Change memory mode on IP26 processor.	page 27	SV, 5.3
is_sysad_parity_enabled()	Test for parity checking on SysAD bus.	page 526	5.3

Table A-4 (continued) Kernel Functions

Name	Summary	Discussed	Versions
itimeout(D3)	Schedule a function to be executed after a specified number of clock ticks.	page 214	SV, 5.3
itoemajor(D3)	Convert internal to external major device number.	page 181	SV, 5.3
kern_calloc(D3)	Allocate and clear space from kernel memory.	page 187	5.3
kern_free(D3)	Free kernel memory space.	page 187	5.3
kern_malloc(D3)	Allocate kernel virtual memory.	page 187	5.3
kmem_alloc(D3)	Allocate space from kernel free memory.	page 187	SV, 5.3
kmem_free(D3)	Free previously allocated kernel memory.	page 187	SV, 5.3
kmem_zalloc(D3)	Allocate and clear space from kernel free memory.	page 187	SV, 5.3
kvtophys(D3)	Get physical address of kernel data.	page 199	5.3
linkb(D3)	Concatenate two message blocks.		SV*, 5.3*
LOCK(D3)	Acquire a basic lock, waiting if necessary.	page 205	SV*, 5.3*
LOCK_ALLOC(D3)	Allocate and initialize a basic lock.	page 205	SV*, 5.3*
LOCK_DEALLOC(D3)	Deallocate an instance of a basic lock.	page 205	SV*, 5.3*
LOCK_INIT(D3)	Initialize a basic lock that was allocated statically, or reinitialize an allocated lock.	page 205	6.2
LOCK_DESTROY(D3)	Uninitialize a basic lock that was allocated statically.	page 205	6.2
makedevice(D3)	Make device number from major and minor numbers.	page 181	SV, 5.3
max(D3)	Return the larger of two integers.		SV, 5.3
min(D3)	Return the lesser of two integers.		SV, 5.3
msgdsize(D3)	Return number of bytes of data in a message.		SV, 5.3

Table A-4 (continued) Kernel Functions

Name	Summary	Discussed	Versions
msgpullup(D3)	Concatenate bytes in a message.		SV, 5.3
MUTEX_ALLOC(D3)	Allocate and initialize a mutex lock.	page 207	6.2
MUTEX_DEALLOC(D3)	Deinitialize and free a dynamically allocated mutex lock.	page 207	6.2
MUTEX_DESTROY(D3)	Deinitialize a mutex lock.	page 207	6.2
MUTEX_INIT(D3)	Initialize an existing mutex lock.	page 207	6.2
MUTEX_ISLOCKED(D3)	Test if a mutex lock is owned.	page 207	6.2
MUTEX_LOCK(D3)	Claim a mutex lock.	page 207	6.2
MUTEX_MINE(D3)	Test if a mutex lock is owned by this process.	page 207	6.2
MUTEX_TRYLOCK(D3)	Conditionally claim a mutex lock.	page 207	6.2
MUTEX_UNLOCK(D3)	Release a mutex lock.	page 207	6.2
MUTEX_WAITQ(D3)	Get the number of processes blocked by mutex lock.	page 207	6.2
ngeteblk(D3)	Allocate a *buf_t* and a buffer of specified size.	page 190	SV, 5.3
noenable(D3)	Prevent a queue from being scheduled.		SV, 5.3
OTHERQ(D3)	Get a pointer to queue's partner queue.		SV, 5.3
pcmsg(D3)	Test whether a message is a priority control message.		SV, 5.3
phalloc(D3)	Allocate and initialize a pollhead structure.	page 189	SV, 5.3
phfree(D3)	Free a pollhead structure.	page 189	SV, 5.3
physiock(D3)	Validate and issue a raw I/O request	page 217	SV, 5.3
pio_andb_rmw(D3)	Byte read-and-write.	page 325	5.3
pio_andh_rmw(D3)	16-bit read-and-write.	page 325	5.3
pio_andw_rmw(D3)	32-bit read-and-write.	page 325	5.3

Table A-4 (continued) Kernel Functions

Name	Summary	Discussed	Versions
pio_badaddr(D3)	Check for bus error when reading an address.	page 325	5.3
pio_badaddr_val(D3)	Check for bus error when reading an address and return the value read.	page 325	5.3
pio_bcopyin(D3)	Copy data from a bus address to kernel's virtual space.	page 325	5.3
pio_bcopyout(D3)	Copy data from kernel's virtual space to a bus address.	page 325	5.3
pio_mapaddr(D3)	Convert a bus address to a virtual address.	page 325	5.3
pio_mapalloc(D3)	Allocate a PIO map.	page 325	5.3
pio_mapfree(D3)	Free a PIO map.	page 325	5.3
pio_orb_rmw(D3)	Byte read-or-write.	page 325	5.3
pio_orh_rmw(D3)	16-bit read-or-write.	page 325	5.3
pio_orw_rmw(D3)	32-bit read-or-write.	page 325	5.3
pio_wbadaddr(D3)	Check for bus error when writing to an address.	page 325	5.3
pio_wbadaddr_val(D3)	Check for bus error when writing a specified value to an address.	page 325	5.3
pollwakeup(D3)	Inform polling processes that an event has occurred.	page 156	SV, 5.3
pptophys(D3)	Convert page pointer to physical address.	page 199	SV, 5.3
proc_ref(D3)	Obtain a reference to a process for signaling.	page 202	SV, 5.3
proc_signal(D3)	Send a signal to a process.	page 202	SV, 5.3
proc_unref(D3)	Release a reference to a process.	page 202	SV, 5.3
psema(D3)	Perform a “P” or wait semaphore operation.	page 222	SV, 5.3
ptob(D3)	Convert size in pages to size in bytes.	page 197	SV, 5.3
pullupmsg(D3)	Concatenate bytes in a message.		SV, 5.3

Table A-4 (continued) Kernel Functions

Name	Summary	Discussed	Versions
putbq(D3)	Place a message at the head of a queue.		SV, 5.3
putctl(D3)	Send a control message to a queue.		SV, 5.3
putctl1(D3)	Send a control message with a one-byte parameter to a queue.		SV, 5.3
putnext(D3)	Send a message to the next queue.		SV, 5.3
putnextctl(D3)	Send a control message to a queue.		SV, 5.3
putnextctl1(D3)	Send a control message with a one-byte parameter to a queue.		SV, 5.3
putq(D3)	Put a message on a queue.		SV, 5.3
qenable(D3)	Schedule a queue's service routine to be run.		SV, 5.3
qprocsoff(D3)	Enable put and service routines.		SV, 5.3
qprocson(D3)	Disable put and service routines		SV, 5.3
qreply(D3)	Send a message in the opposite direction in a stream.		SV, 5.3
qsize(D3)	Find the number of messages on a queue.		SV, 5.3
RD(D3)	Get a pointer to the read queue.		SV, 5.3
rmalloc(D3)	Allocate space from a private space management map.	page 191	SV, 5.3
rmallocmap(D3)	Allocate and initialize a private space management map.	page 191	SV, 5.3
rmalloc_wait(D3)	Allocate resources from a space management map.	page 191	SV, 5.3
rmfree(D3)	Release resources into a space management map.	page 191	SV, 5.3
rmfreemap(D3)	Free a private space management map.	page 191	SV, 5.3
rmvb(D3)	Remove a message block from a message.		SV, 5.3
rmvq(D3)	Remove a message from a queue.		SV, 5.3

Table A-4 (continued) Kernel Functions

Name	Summary	Discussed	Versions
RW_ALLOC(D3)	Allocate and initialize a reader/writer lock.	page 211	SV*, 5.3*
RW_DEALLOC(D3)	Deallocate a reader/writer lock.	page 211	SV*, 5.3*
RW_DESTROY(D3)	Deinitialize an existing reader/writer lock.	page 211	6.2
RW_INIT(D3)	Initialize an existing reader/writer lock.	page 211	6.2
RW_RDLOCK(D3)	Acquire a reader/writer lock as reader, waiting if necessary.	page 211	SV*, 5.3*
RW_TRYRDLOCK(D3)	Try to acquire a reader/writer lock as reader, returning a code if it is not free.	page 211	SV*, 5.3*
RW_TRYWRLOCK(D3)	Try to acquire a reader/writer lock as writer, returning a code if it is not free.	page 211	SV*, 5.3*
RW_UNLOCK(D3)	Release a reader/writer lock as reader or writer.	page 211	SV*, 5.3*
RW_WRLOCK(D3)	Acquire a reader/writer lock as writer, waiting if necessary.	page 211	SV*, 5.3*
SAMESTR(D3)	Test if next queue is of the same type.		SV, 5.3
scsi_abort()	Transmits a SCSI ABORT command.	page 357	5.3*
scsi_alloc(D3)	Open a connection between a driver and a target device.	page 357	5.3*
scsi_command(D3)	Transmit a SCSI command on the bus and return results.	page 357	5.3*
scsi_free(D3)	Release connection to target device.	page 357	5.3*
scsi_info(D3)	Issue the SCSI Inquiry command and return the results.	page 357	5.3*
scsi_reset()	Resets the SCSI adapter or bus.	page 357	5.3*
setgiovector()	Register a GIO interrupt handler.	page 515	5.3
setgioconfig()	Prepare a GIO slot for use.	page 516	5.3
sgset(D3)	Assign physical addresses to a vector of software scatter-gather registers.	page 199	5.3

Table A-4 (continued) Kernel Functions

Name	Summary	Discussed	Versions
sleep(D3)	Suspend process execution pending occurrence of an event.	page 219	SV, 5.3
SLEEP_ALLOC(D3)	Allocate and initialize a sleep lock.	page 209	SV*, 5.3*
SLEEP_DEALLOC(D3)	Deinitialize and deallocate a dynamically allocated sleep lock.	page 209	SV*, 5.3*
SLEEP_DESTROY	Deinitialize a sleep lock.	page 209	6.2
SLEEP_INIT(D3)	Initialize an existing sleep lock.	page 209	6.2
SLEEP_LOCK(D3)	Acquire a sleep lock, waiting if necessary until the lock is free.	page 209	SV*, 5.3*
SLEEP_LOCKAVAIL(D3)	Query whether a sleep lock is available.	page 209	SV*, 5.3*
SLEEP_LOCK_SIG(D3)	Acquire a sleep lock, waiting if necessary until the lock is free or a signal is received.	page 209	SV*, 5.3*
SLEEP_TRYLOCK(D3)	Try to acquire a sleep lock, returning a code if it is not free.	page 209	SV*, 5.3*
SLEEP_UNLOCK(D3)	Release a sleep lock.	page 209	SV*, 5.3*
splbase(D3)	Block no interrupts.	page 213	SV, 5.3
spltimeout(D3)	Block only timeout interrupts.	page 213	SV, 5.3
spldisk(D3)	Block disk interrupts.	page 213	SV, 5.3
splstr(D3)	Block STREAMS interrupts.	page 213	SV, 5.3
spltty(D3)	Block disk, VME, serial interrupts.	page 213	SV, 5.3
splhi(D3)	Block all I/O interrupts.	page 213	SV, 5.3
spl0(D3)	Same as **splbase**().	page 213	SV, 5.3
splx(D3)	Restore previous interrupt level.	page 213	SV, 5.3
strcat(D3)	Append one string to another.		SV, 5.3
strcpy(D3)	Copy a string.		SV, 5.3

Table A-4 (continued) Kernel Functions

Name	Summary	Discussed	Versions
streams_interrupt(D3)	Synchronize interrupt-level function with STREAMS mechanism.		5.3
STREAMS_TIMEOUT(D3)	Synchronize timeout with STREAMS mechanism.		5.3
strlen(D3)	Return length of a string.		SV, 5.3
strlog(D3)	Submit messages to the log driver.		SV, 5.3
strncmp(D3)	Compare two strings for a specified length.		SV, 5.3
strncpy(D3)	Copy a string for a specified length.		SV, 5.3
strqget(D3)	Get information about a queue or band of the queue.		SV, 5.3
strqset(D3)	Change information about a queue or band of the queue.		SV, 5.3
subyte(D3)	Store a byte to user space.	page 192	5.3
suword(D3)	Store a word to user space.	page 192	5.3
SV_ALLOC(D3)	Allocate and initialize a synchronization variable.	page 220	SV*, 5.3*
SV_BROADCAST(D3)	Wake all processes sleeping on a synchronization variable.	page 220	SV*, 5.3*
SV_DEALLOC(D3)	Deinitialize and deallocate a synchronization variable.	page 220	SV*, 5.3*
SV_DESTROY	Deinitialize a synchronization variable.	page 220	6.2
SV_INIT	Initialize an existing synchronization variable.	page 220	6.2
SV_SIGNAL(D3)	Wake one process sleeping on a synchronization variable.	page 220	SV*, 5.3*
SV_WAIT(D3)	Sleep until a synchronization variable is signalled.	page 220	SV*, 5.3*

Table A-4 (continued) Kernel Functions

Name	Summary	Discussed	Versions
SV_WAIT_SIG(D3)	Sleep until a synchronization variable is signalled or a signal is received.	page 220	SV*, 5.3*
timeout(D3)	Schedule a function to be executed after a specified number of clock ticks.	page 214	SV, 5.3
TRYLOCK(D3)	Try to acquire a basic lock, returning a code if the lock is not currently free.	page 205	SV*, 5.3*
uiomove(D3)	Copy data using *uio_t*.	page 194	SV, 5.3
uiophysio(D3)	Validate a raw I/O request and pass to a strategy function.	page 217	5.3
unbufcall(D3)	Cancel a pending bufcall request.		SV, 5.3
undma(D3)	Unlock physical memory in user space.	page 217	5.3
unfreezestr(D3)	Unfreeze the state of a stream.		SV, 5.3
unlinkb(D3)	Remove a message block from the head of a message.		SV, 5.3
UNLOCK(D3)	Release a basic lock.	page 205	SV*, 5.3*
untimeout(D3)	Cancel a previous itimeout or fast_itimeout request.	page 214	SV*, 5.3*
ureadc(D3)	Copy a character to space described by *uio_t*.	page 194	SV, 5.3
userdma(D3)	Lock physical memory in user space.small number of	page 217	5.3
userabi()	Get data sizes for the ABI of the user process (32- or 64-bit).	page 170	6.2
uwritec(D3)	Return a character from space described by *uio_t*.	page 194	SV, 5.3
v_getaddr(D3)	Get the user virtual address associated with a *vhandl_t*.	page 196	5.3
v_gethandle(D3)	Get a unique identifier associated with a *vhandl_t*.	page 196	5.3

Table A-4 (continued) Kernel Functions

Name	Summary	Discussed	Versions
v_getlen(D3)	Get the length of user address space associated with a *vhandl_t*.	page 196	5.3
v_mapphys(D3)	Map kernel address space into user address space.	page 196	5.3
valusema(D3)	Return the value associated with a semaphore.	page 222	5.3
vme_adapter(D3)	Determine VME adapter that corresponds to a given memory address.	page 330	5.3
vme_ivec_alloc(D3)	Allocate a VME bus interrupt vector.	page 332	5.3
vme_ivec_free(D3)	Free a VME bus interrupt vector.	page 332	5.3
vme_ivec_set(D3)	Register a VME bus interrupt vector.	page 332	5.3
vsema(D3)	Perform a "V" or signal semaphore operation.	page 222	5.3
wakeup(D3)	Waken a process waiting for an event.	page 219	SV, 5.3
wbadaddr(D3)	Test physical address for output.	page 195	SV, 5.3
wbadaddr_val(D3)	Test physical address for output of specific value.	page 195	SV, 5.3
WR(D3)	Get a pointer to the write queue.		SV, 5.3

The following SVR4 kernel functions are not implemented in IRIX: bioreset, dma_disable, dma_enable, dma_free_buf, dma_free_cb, dma_get_best_mode, dma_get_buf, dma_get_cb, dma_pageio, dma_prog, dma_swstart, dma_swsetup, drv_gethardware, hat_getkpfnum, hat_getppfnum, inb, inl, inw, kvtoppid, mod_drvattach, mod_drvdetach, outb, outl, outw, physmap, physmap_free, phystoppid, psignal, rdma_filter, repinsb, repinsd, repinsw, repoutsb, repoutsd, repoutsw, rminit, rmsetwant, SLEEP_LOCKOWNED, strncat, vtop.