

OpenBoot 3.x

Quick Reference

Sun Microsystems Computer Corporation
A Sun Microsystems, Inc. Business
2550 Garcia Avenue
Mountain View, CA 94043 U.S.A.
415 960-1300 FAX 415 969-9131
Part No: 802-3240-10
Revision A, November 1995

Syntax

Enter commands at the `ok` prompt. They are executed left-to-right after a carriage-return. Separate all commands by one or more spaces.

Help Commands

<code>help</code>	List main help categories.
<code>help category</code>	Show help for all commands in the category. Use only the first word of the category description.
<code>help command</code>	Show help for individual command (where available).

Examining and Creating Device Aliases

<code>devalias</code>	Display all current device aliases.
<code>devalias alias</code>	Display the device path name corresponding to alias.
<code>devalias alias device-path</code>	Define an alias representing the device path. If an alias with the same name already exists, the new value supersedes the old.

Device Tree Browsing Commands

<code>.properties</code>	Display the names and values of the current node's properties.
<code>dev node-name</code>	Search for a node with the given name in the subtree below the current node, and choose the first such node found.
<code>dev ..</code>	Choose the device node that is the parent of the current node.
<code>dev /</code>	Choose the root machine node.
<code>device-end</code>	Leave the device tree.
<code>ls</code>	Display the names of the current node's children.
<code>pwd</code>	Display the device path name that names the current node.
<code>show-devs [device-path]</code>	Display all the devices directly under the specified device in the device tree. <code>show-devs</code> used alone shows the entire device tree.
<code>words</code>	Display the names of the current node's methods.

Common Options for the boot Command

<code>boot [device-specifier][filename] [option]</code>	
[device-specifier]	The name (full path name or alias) of a device. Examples: <code>cdrom</code> (CD-ROM drive) <code>disk</code> (hard disk) <code>floppy</code> (3-1/2" diskette drive) <code>net</code> (Ethernet) <code>tape</code> (SCSI tape)
[filename]	The name of the program to be booted (for example, <code>stand/diag</code>). If specified, filename is relative to the root of the selected device and partition. If not, the boot program uses the value of the <code>boot-file</code> parameter.

[option\$	-a - Prompt interactively for the device and name of the boot file.
	-h - Halt after loading the program.

(OS-specific options may differ from system to system)

Dagnostic Test Commands

probe-scsi	Identify devices attached to the built-in SCSI bus.
test device-specifier	Execute the specified device's self-test method. For example: test floppy - test the floppy drive, if installed test net - test the network connection
test-all [device-specifier]	Test all devices (that have a built-in self-test method) below the specified node. (If device-specifier is absent, the root node is used.)
watch-clock	Test the clock function.
watch-net	Monitor the network connection.

System Information Display Commands

banner	Display the power-on banner.
.version	Display the version and date of the boot PROM.
.speed	Display CPU and bus speeds.

Emergency Keyboard Commands

Hold down keys during power-on sequence.

Stop	Bypass POST. This command does not depend on security-mode. (Note: some systems bypass POST as a default; in such cases, use <code>stop-D</code> to start POST.)
Stop-A	Abort.
Stop-D	Enter diagnostic mode (set <code>diag-switch?</code> to true).
Stop-F	Enter Forth on TTYA instead of probing. Use <code>!exit</code> to continue with the initialization sequence. (Useful if hardware is broken.)
Stop-N	Reset NVRAM contents to default values.

File Loading Commands

boot [specifier \$ -h	(--)	Load file from specified source.
byte-load	(adr xt--)	Interpret a loaded FCode binary file. xt is usually 1.
dl	(--)	Load a Forth file over a serial line with TIP and interpret. Type: ~C cat filename ^~D
dlbin	(--)	Load a binary file over a serial line with TIP. Type: ~C cat filename
dload filename	(adr --)	Load specified file over Ethernet to given address.
go	(--)	Begin executing a previously-loaded binary program, or resume executing an interrupted program.

init-program	(--)	Initialize to execute a binary file.
load [<i>specifier</i>]	(--)	Load data from specified device into memory at the address given by load-base. (See boot format.)
load-base	(-- adr)	Address at which load places the data it reads from a device.

SPARC™ Register Commands

%g0 through %g7	(-- value)	Return the value in the given register.
%i0 through %i7	(-- value)	Return the value in the given register.
%l0 through %l7	(-- value)	Return the value in the given register.
%o0 through %o7	(-- value)	Return the value in the given register.
%pc %npc	(-- value)	Return the value in the given register.
.fregisters	(--)	Display values in %f0 through %f31.
.locals	(--)	Display the values in the i, l and o registers.
.registers	(--)	Display values in %g0 through %g7, plus %pc, %npc, %psr, %y, %wim, %tbr.
.window	(window# --)	Display the desired window.
ctrace	(--)	Display the return stack showing C subroutines.
set-pc	(value --)	Set %pc to the given value, and set %npc to (value+4).
to <i>regname</i>	(value --)	Change the value stored in any of the above registers. Use in the form: value to <i>regname</i>
w	(window# --)	Set the current window for displaying %ix %lx or %ox.

SPARC V9 Register Commands

%fprs %asi %pstate %tl-c %pil %tstate %tt %tba %cwp %cansave %canrestore %otherwin %wstate %cleanwin	(-- value)	Return the value in the specified register.
.pstate	(--)	Formatted display of the processor state register.
.ver	(--)	Formatted display of the version register.
.ccr	(--)	Formatted display of the ccr register.
.trap-registers	(--)	Display trap-related registers.

Breakpoint Commands

<code>+bp</code>	(<code>adr --</code>)	Add a breakpoint at the given address.
<code>-bp</code>	(<code>adr --</code>)	Remove the breakpoint at the given address.
<code>--bp</code>	(<code>--</code>)	Remove the most-recently-set breakpoint.
<code>.bp</code>	(<code>--</code>)	Display all currently set breakpoints.
<code>.breakpoint</code>	(<code>--</code>)	Perform a specified action when a breakpoint occurs (Example, ['] <code>.registers</code> to <code>.breakpoint</code>).
<code>.instruction</code>	(<code>--</code>)	Display the address, opcode for the last-encountered breakpoint.
<code>.step</code>	(<code>--</code>)	Perform a specified action when a single step occurs (see <code>.breakpoint</code>).
<code>bpoff</code>	(<code>--</code>)	Remove all breakpoints.
<code>finish-loop</code>	(<code>--</code>)	Execute until the end of this loop.
<code>go</code>	(<code>--</code>)	Continue from a breakpoint. This can be used to go to an arbitrary address by setting up the processor's program counter before issuing <code>go</code> .
<code>gos</code>	(<code>n --</code>)	Execute <code>go</code> <code>n</code> times.
<code>hop</code>	(<code>--</code>)	(Like the <code>step</code> command.) Treats a subroutine call as a single instruction.
<code>hops</code>	(<code>n --</code>)	Execute <code>hop</code> <code>n</code> times.
<code>return</code>	(<code>--</code>)	Execute until the end of this subroutine.
<code>returnl</code>	(<code>--</code>)	Execute until the end of this leaf subroutine.
<code>skip</code>	(<code>--</code>)	Skip (do not execute) the current instruction.
<code>step</code>	(<code>--</code>)	Single-step one instruction.
<code>steps</code>	(<code>n --</code>)	Execute <code>step</code> <code>n</code> times.
<code>till</code>	(<code>adr --</code>)	Execute until the given address is encountered. Equivalent to <code>+bp go</code> .

Disassembler Commands

<code>+dis</code>	(<code>--</code>)	Continue disassembling where the last disassembly left off.
<code>dis</code>	(<code>adr --</code>)	Begin disassembling at the given address.

Miscellaneous Operations

<code>eject-floppy</code>	(<code>--</code>)	Eject the diskette from the drive.
<code>firmware-version</code>	(<code>-- n</code>)	Return major/minor CPU firmware version (that is, <code>0x00020001</code> = firmware version 2.1).
<code>ftrace</code>	(<code>--</code>)	Show calling sequence when exception occurred.
<code>get-msecs</code>	(<code>-- ms</code>)	Return the approximate current time in milliseconds.
<code>ms</code>	(<code>n --</code>)	Delay for <code>n</code> milliseconds. Resolution is 1 millisecond.
<code>reset-all</code>	(<code>--</code>)	Reset the entire system (similar to a power cycle).

sync	(--)	Call the operating system to write any pending information to the hard disk.
------	--------	--

NVRAM Configuration Parameters

auto-boot?	true	If true, boot automatically after power-on or reset.
boot-command	boot	Executed when auto-boot? is true.
boot-device	disk net	Device from which to boot.
boot-file	empty string	File to boot (an empty string lets secondary booter choose default).
diag-device	net	Diagnostic boot source device.
diag-file	empty string	File from which to boot in diagnostic mode.
diag-level	min	Level of diagnostics to run (min or max).
diag-switch?	false	If true, run in diagnostic mode.
fcode-debug?	false	If true, include name fields for plug-in device FCodes.
input-device	keyboard	Power-on input device (usually keyboard, ttya, or ttyb).
keyboard-click?	false	If true, enable keyboard click.
keymap	no default	Keymap for custom keyboard.
nvrामrc	empty string	Contents of NVRAMRC.
oem-banner	empty string	Custom OEM banner (enabled by oem-banner? true).
oem-banner?	false	If true, use custom OEM banner.
oem-logo	no default	Byte array custom OEM logo (enabled by oem-logo? true). Displayed in hex.
oem-logo?	false	If true, use custom OEM logo (else, use Sun logo).
output-device	screen	Power-on output device (usually screen, ttya, or ttyb).
sbus-probe-list	01	Which SBus slots are probed and in what order.
scsi-initiator-id	7	SCSI bus address of host adapter, range 0-f.
security-mode	none	Firmware security level (none, command, or full).
security-password	no default	Firmware security password (never displayed).
ttya-mode	9600,8,n,1,-	TTYA (baud, #bits, parity, #stop, handshake).
ttyb-mode	9600,8,n,1-	TTYB (baud, #bits, parity, #stop, handshake).
ttya-ignore-cd	true	If true, OS ignores TTYA carrier-detect.
ttyb-ignore-cd	true	If true, OS ignores TTYB carrier-detect.

<code>ttya-rts-dtr-off</code>	<code>false</code>	If true, OS does not assert DTR and RTS on TTYA.
<code>ttyb-rts-dtr-off</code>	<code>false</code>	If true, OS does not assert DTR and RTS on TTYB.
<code>use-nvramrc?</code>	<code>false</code>	If true, execute commands in NVRAMRC during system start-up.
<code>watchdog-reboot?</code>	<code>false</code>	If true, reboot after watchdog reset.

Viewing and Changing Configuration Parameters

<code>password</code>		Set <code>security-password</code> .
<code>printenv</code> [parameter]		Display all current parameters and current default values (numbers are usually shown as decimal values). <code>printenv</code> parameter shows the current value of the named parameter.
<code>setenv</code> parameter value		Set the parameter to the given decimal or text value. (Changes are permanent, but usually only take effect after a reset).
<code>set-default</code> parameter		Reset the value of the named parameter to the factory default.
<code>set-defaults</code>		Reset parameter values to the factory defaults.

NVRAMRC Editor Commands

<code>nvalias</code> alias device-path		Store the command " <code>devalias</code> alias device-path in NVRAMRC. (The alias persists until the <code>nvunalias</code> or <code>set-defaults</code> commands are executed.)
<code>nvedit</code>		Enter the NVRAMRC editor. If data remains in the temporary buffer from a previous <code>nvedit</code> session, resume editing those previous contents. If not, read the contents of NVRAMRC into the temporary buffer and begin editing it.
<code>nvquit</code>		Discard the contents of the temporary buffer, without writing it to NVRAMRC.
<code>nvrecover</code>		Recover the contents of NVRAMRC if they have been lost as a result of the execution of <code>set-defaults</code> ; then enter the editor as with <code>nvedit</code> . <code>nvrecover</code> fails if <code>nvedit</code> is executed between the time that the NVRAMRC contents were lost and the time that <code>nvrecover</code> is executed.
<code>nvstore</code>		Copy the contents of the temporary buffer to NVRAMRC; discard the contents of the temporary buffer.
<code>nvunalias</code> alias		Delete the corresponding alias from NVRAMRC.

Editor Commands (for Command Lines and NVRAMRC)

	Prev. Line	Beg. Line	Prev. Word	Prev. Char	Next Char	Next Word	End Line	Next Line
Move	^ P	^ A	escB	^ B	^ F	escF	^ E	^ N
Delete		^ U	^ W	Del	^ D	escD	^ K	
			Re-type line		^R			
			Show all lines		^L			
			Paste after ^K		^Y			
			Complete command		^ space			
			Show all matches		^/ or ^?}			

esc = Press and release Escape key first; ^ = Press and hold Control key

Using the NVRAMRC Editor

```

ok nvedit
:
(use editor commands)
:
^C (get back took prompt)
ok nvstore (save changes)
ok setenv use-nvramrc? true (enable NVRAMRC)

```

Numeric Usage and Stack Comments

- Numeric I/O defaults to hexadecimal.
- Switch to decimal with `decimal`, switch to hexadecimal with `hex`.
- Use `10 .d` to see which base is currently active.

A numeric stack is used for all numeric parameters. Typing any integer puts that value on top of the stack. (Previous values are pushed down.) The right-hand item in a set always indicates the topmost stack item.

- The command `.` removes and displays the top stack value.
- The command `.s` non-destructively shows the entire stack contents.

A stack comment such as `(n1 n2 -- n3)` or `(adr len --)` or `(--)` listed after each command name shows the effect on the stack of executing that command. Items **before** the `--` are used by the command and removed from the stack. These items **must** be present on the stack **before** the command can properly execute. Items **after** the `--` are left on the stack after the command completes execution, and are available for use by subsequent commands.

	Alternate stack results. Example: (input -- adr len false result true).
?	Unknown stack items (changed from ???).
???	Unknown stack items.
adr	Memory address (generally a virtual address).
adr16	Memory address, must be 16-bit aligned.
adr32	Memory address, must be 32-bit aligned.
adr64	Memory address, must be 64-bit aligned.
byte xxx	8-bit value (smallest byte in a 32-bit word).
char	7-bit value (smallest byte), high bit unspecified.
cnt/len/size	Count or length.

flag xxx?	0 = false; any other value = true (usually -1).
long lxxx	32-bit value.
n n1 n2 n3	Normal signed values.
+n u	Unsigned, positive values.
phys	Physical address (actual hardware address).
pstr	Packed string (adr len means unpacked string).
virt	Virtual address (address used by software).
word wxxx	16-bit value.
xt	Execution token.

Changing the Number Base

decimal	(--)	Set the number base to 10.
d# number	(-- n)	Interpret the next number in decimal; base is unchanged.
hex	(--)	Set the number base to 16.
h# number	(-- n)	Interpret the next number in hex; base is unchanged.
.d	(n --)	Display n in decimal without changing base.
.h	(n --)	Display n in hex without changing base.

Basic Number Display

.	(n --)	Display a number in the current base.
.s	(--)	Display contents of data stack.
showstack	(--)	Execute .s automatically before each ok prompt.

Stack Manipulation Commands

-rot	(n1 n2 n3 -- n3 n1 n2)	Inversely rotate three stack items.
>r	(n --)	Move a stack item to the return stack.
?dup	(n -- n n 0)	Duplicate the top stack item if non-zero.
2drop	(n1 n2 --)	Remove two items from the stack.
2dup	(n1 n2 -- n1 n2 n1 n2)	Duplicate two stack items.
2over	(n1 n2 n3 n4 -- n1 n2 n3 n4 n1 n2)	Copy second two stack items.
2swap	(n1 n2 n3 n4 -- n3 n4 n1 n2)	Exchange two pairs of stack items.
clear	(??? --)	Empty the stack.
depth	(??? -- ??? +n)	Return the number of items on the stack.
drop	(n --)	Remove the top item from the stack.
dup	(n -- n n)	Duplicate the top stack item.
over	(n1 n2 -- n1 n2 n1)	Copy the second stack item to the top of the stack.

<code>pick</code>	<code>(??? +n -- ??? n2)</code>	Copy +n-th stack item (1 <code>pick</code> = <code>over</code>).
<code>r></code>	<code>(-- n)</code>	Move a return stack item to the stack.
<code>r@</code>	<code>(-- n)</code>	Copy the top of the return stack to the stack.
<code>roll</code>	<code>(??? +n -- ?)</code>	Rotate +n stack items (2 <code>roll</code> = <code>rot</code>).
<code>rot</code>	<code>(n1 n2 n3 -- n2 n3 n1)</code>	Rotate three stack items.
<code>swap</code>	<code>(n1 n2 -- n2 n1)</code>	Exchange the top two stack items.
<code>tuck</code>	<code>(n1 n2 -- n2 n1 n2)</code>	Copy the top stack item below the second item.

Arithmetic Functions

<code>*</code>	<code>(n1 n2 -- n3)</code>	Multiply <code>n1 * n2</code> .
<code>+</code>	<code>(n1 n2 -- n3)</code>	Add <code>n1 + n2</code> .
<code>-</code>	<code>(n1 n2 -- n3)</code>	Subtract <code>n1 - n2</code>
<code>/</code>	<code>(n1 n2 -- quot)</code>	Divide <code>n1 / n2</code> ; remainder is discarded.
<code>lshift</code>	<code>(n1 +n -- n2)</code>	Left-shift <code>n1</code> by +n bits.
<code>rshift</code>	<code>(n1 +n -- n2)</code>	Right-shift <code>n1</code> by +n bits.
<code>>>a</code>	<code>(n1 +n -- n2)</code>	Arithmetic right-shift <code>n1</code> by +n bits.
<code>abs</code>	<code>(n -- u)</code>	Absolute value.
<code>and</code>	<code>(n1 n2 -- n3)</code>	Bitwise logical AND.
<code>bounds</code>	<code>(startadr len -- endadr startadr)</code>	Convert <code>startadr len</code> to <code>endadr startadr</code> for <code>do</code> loop.
<code>bljoin</code>	<code>(b.low b2 b3 b.hi -- long)</code>	Join four bytes to form a 32-bit value.
<code>bwjoin</code>	<code>(b.low b.hi -- word)</code>	Join two bytes to form a 16-bit value.
<code>lbsplit</code>	<code>(long -- b.low b2 b3 b.hi)</code>	Split a 32-bit value into four bytes.
<code>lwsplit</code>	<code>(long -- w.low w.hi)</code>	Split a 32-bit value into two 16-bit words.
<code>max</code>	<code>(n1 n2 -- n3)</code>	<code>n3</code> is maximum of <code>n1</code> and <code>n2</code> .
<code>min</code>	<code>(n1 n2 -- n3)</code>	<code>n3</code> is minimum of <code>n1</code> and <code>n2</code> .
<code>mod</code>	<code>(n1 n2 -- rem)</code>	Remainder of <code>n1 / n2</code> .
<code>negate</code>	<code>(n1 -- n2)</code>	Change the sign of <code>n1</code> .
<code>invert</code>	<code>(n1 -- n2)</code>	Bitwise ones complement.
<code>or</code>	<code>(n1 n2 -- n3)</code>	Bitwise logical OR.
<code>wbsplit</code>	<code>(word -- b.low b.hi)</code>	Split 16-bit value into two bytes.
<code>wljoin</code>	<code>(w.low w.hi -- long)</code>	Join two 16-bit values to form a 32-bit value.
<code>xor</code>	<code>(n1 n2 -- n3)</code>	Bitwise exclusive OR.

Memory Access Commands

!	(n adr32 --)	Store a number at adr32, must be 32-bit aligned.
+!	(n adr32 --)	Add n to the number stored at adr32, must be 32-bit aligned.
@	(adr32 -- n)	Fetch a number from adr32, must be 32-bit aligned.
c!	(n adr --)	Store low byte of n at adr.
c@	(adr -- byte)	Fetch a byte from adr.
cpeek	(adr -- false byte true)	Fetch the byte at adr. Return the data and true if the access was successful. Return false if a read access error occurred. (Also lpeek, wpeek.)
cpoke	(byte adr -- okay?)	Store the byte to adr. Return true if the access was successful. Return false if a write access error occurred. (Also lpoke, wpoke.)
comp	(adr1 adr2 len -- n)	Compare two byte arrays, n = 0 if arrays are identical, n = 1 if first byte that is different is greater in array#1, n = -1 otherwise.
dump	(adr len --)	Display len bytes of memory starting at adr.
fill	(adr size byte --)	Set size bytes of memory to byte.
l!	(n adr32 --)	Store a 32-bit number at adr32.
l@	(adr32 -- long)	Fetch a 32-bit number from adr32.
move	(adr1 adr2 u --)	Copy u bytes from adr1 to adr2, handle overlap properly.
w!	(n adr16 --)	Store a 16-bit number at adr16, must be 16-bit aligned.
w@	(adr16 -- word)	Fetch a 16-bit number from adr16, must be 16-bit aligned.

Memory Mapping Commands

<code>alloc-mem</code>	(size -- virt)	Allocate and map size bytes of available memory; return the virtual address. Unmap with <code>free-mem</code> .
<code>free-mem</code>	(virt size --)	Free memory allocated by <code>alloc-mem</code> .
<code>free-virtual</code>	(virt size --)	Undo mappings created with <code>memmap</code> .
<code>map?</code>	(virt --)	Display memory map information for the virtual address.
<code>memmap</code>	(phys space size -- virt)	Map a region of physical addresses; return the allocated virtual address. Unmap with <code>free-virtual</code> .
<code>obio</code>	(-- space)	Specify the device address space for mapping.
<code>obmem</code>	(-- space)	Specify the onboard memory address space for mapping.
<code>pgmap!</code>	(pmentry virt --)	Store a new page map entry for the virtual address.
<code>pgmap?</code>	(virt --)	Display the decoded page map entry corresponding to the virtual address.
<code>pgmap@</code>	(virt -- pmentry)	Return the page map entry for the virtual address.
<code>pagesize</code>	(-- size)	Return the size of a page (often 4K).
<code>sbus</code>	(-- space)	Specify the SBus address space for mapping.

Defining Words

<code>:</code>	name (--) Usage: (??? -- ?)	Start creating a new colon definition.
<code>;</code>	(--)	Finish creating a new colon definition.
<code>buffer:</code>	name (size --) Usage: (-- adr64)	Create a named array in temporary storage.
<code>constant</code>	name (n --) Usage: (-- n)	Define a constant (for example, 3 constant bar).
<code>create</code>	name (--) Usage: (-- adr16)	Generic defining word.
<code>defer</code>	name (--) Usage: (??? -- ?)	Define forward reference or execution vector.
<code>value</code>	name (n --) Usage: (-- n)	Create a changeable, named 32-bit quantity.
<code>variable</code>	name (--) Usage: (-- adr16)	Define a variable.

Dictionary Searching Commands

' name	(-- xt)	Find the named word in the dictionary. (Returns the execution token. Use outside definitions.)
['] name	(-- xt)	Similar to ' but is used inside definitions.
.calls	(xt --)	Display a list of all words that call the word whose execution token is xt.
\$find	(adr len -- adr len false xt n)	Find a word. n = 0 if not found, n = 1 if immediate, n = -1 otherwise.
see thisword	(--)	Decompile the named command.
(see)	(xt --)	Decompile the word indicated by the execution token.
sifting ccc	(--)	Display names of all dictionary entries containing the sequence of characters. ccc contains no spaces.
words	(--)	Display visible words in the dictionary.

Dictionary Compilation Commands

,	(n --)	Place a number in the dictionary.
c,	(byte --)	Place a byte in the dictionary.
w,	(word --)	Place a 16-bit number in the dictionary.
l,	(long --)	Place a 32-bit number in the dictionary.
allot	(n --)	Allocate n bytes in the dictionary.
forget name	(--)	Remove word from dictionary and all subsequent words.
here	(-- adr)	Address of top of dictionary.
to name	(n --)	Install a new action in a defer word or value.
patch new-word old-word word-to-patch	(--)	Replace old-word with new-word in word-to-patch.
(patch)	(new-n old-n xt --)	Replace old-n with new-n in word indicated by xt.

Controlling Text Input

(ccc)	(--)	Begin a comment.
\ rest-of-line	(--)	Skip the rest of the line.
ascii ccc	(-- char)	Get numerical value of first ASCII character of next word.
key	(-- char)	Read a character from the assigned input device's keyboard.

key?	(-- flag)	True if a key has been typed on the input device's keyboard.
------	-------------	--

Displaying Text Output

cr	(--)	Terminate a line on the display and go to the next line.
emit	(char --)	Display the character.
type	(adr +n --)	Display n characters.

Manipulating Text Strings

" ccc	(-- adr len)	Collect an input stream string, either interpreted or compiled.
." ccc	(--)	Compile a string for later display.
bl	(-- char)	ASCII code for the space character; decimal 32.
count	(pstr -- adr +n)	Unpack a packed string.
p" ccc	(-- pstr)	Collect a string from the input stream; store as a packed string.

Redirecting I/O

input	(device --)	Select device (ttya, ttyb, keyboard, or " devicespecifier) for subsequent input.
io	(device --)	Select device for subsequent input and output.
output	(device --)	Select device (ttya, ttyb, screen, or " devicespecifier) for subsequent output.

Comparison Commands

<	(n1 n2 -- flag)	True if n1 < n2.
<=	(n1 n2 -- flag)	True if n1 <= n2.
<>	(n1 n2 -- flag)	True if n1 <> n2.
=	(n1 n2 -- flag)	True if n1 = n2.
>	(n1 n2 -- flag)	True if n1 > n2.
>=	(n1 n2 -- flag)	True if n1 >= n2.
between	(n min max -- flag)	True if min <= n <= max.
u<	(u1 u2 -- flag)	True if u1 < u2, unsigned.
u<=	(u1 u2 -- flag)	True if u1 <= u2, unsigned.
u>	(u1 u2 -- flag)	True if u1 > u2, unsigned.
u>=	(u1 u2 -- flag)	True if u1 >= u2, unsigned.
within	(n min max -- flag)	True if min <= n < max.

if-else-then Commands

else	(--)	Execute the following code if if failed.
if	(flag --)	Execute the following code if flag is true.
then	(--)	Terminate if...else...then.

begin (Conditional) Loop Commands

again	(--)	End a begin...again infinite loop.
begin	(--)	Begin a begin...while...repeat, begin...until, or begin...again loop.
repeat	(--)	End a begin...while...repeat loop.
until	(flag --)	Continue executing a begin...until loop until flag is true.
while	(flag --)	Continue executing a begin...while...repeat loop while flag is true.

do (Counted) Loop Commands

+loop	(n --)	End a do...+loop construct; add n to loop index and return to do (if n < 0, index goes from start to end inclusive).
?do	(end start --)	Begin ?do...loop to be executed 0 or more times. Index goes from start to end-1 inclusive. If end = start, loop is not executed.
do	(end start --)	Begin a do...loop. Index goes from start to end-1 inclusive. Example: 10 0 do i . loop (prints 0 1 2...d e f).
i	(-- n)	Loop index.
j	(-- n)	Loop index for next enclosing loop.
leave	(--)	Exit from do...loop.
loop	(--)	End of do...loop.

case Statement

```
( value )
case
2 of ." it was two" endof
0 of ." it was zero" endof
." it was " dup . (optional default clause)
endcase
```

Cache Manipulation Commands

clear-cache	(--)	Invalidate all cache entries.
cache-off	(--)	Disable the cache.
cache-on	(--)	Enable the cache.
flush-cache	(--)	Write back any pending data from the cache.

Alternate Address Space Access Commands

spacec!	(byte adr asi --)	Store the byte at asi and address.
spacec@	(adr asi -- byte)	Fetch the byte from asi and address.
spaced!	(n1 n2 adr asi --)	Store the two values at asi and address. Order is implementation-dependent.
spaced@	(adr asi -- n1 n2)	Fetch the two values from asi and address. Order is implementation-dependent.
space1!	(long adr asi --)	Store the 32-bit word at asi and address.

space1@	(adr asi -- long)	Fetch the 32-bit word from asi and address.
spacew!	(word adr asi --)	Store the 16-bit word at asi and address.
spacew@	(adr asi -- word)	Fetch the 16-bit word from asi and address.
spacex!	(x adr asi --)	Store the 64-bit word at asi and address.
spacex@	(adr asi -- x)	Fetch the 64-bit word from asi and address.

Multiprocessor Commands

switch-cpu	(cpu# --)	Switch to indicated CPU.
------------	-------------	--------------------------

Program Execution Control Commands

abort	(--)	Abort current execution and interpret keyboard commands.
abort" ccc "	(abort? --)	If flag is true, abort and display message.
eval	(adr len --)	Interpret Forth source from an array.
execute	(xt--)	Execute the word whose execution token is on the stack.
exit	(--)	Return from the current word. (Cannot be used in counted loops.)
quit	(--)	Same as abort, but leave stack intact.

© 1995, Sun Microsystems, Inc.—Printed in the United States of America.

Sun, Sun Microsystems, the Sun logo, and OpenBoot are trademarks or registered trademarks of Sun Microsystems, Inc. All SPARC trademarks are trademarks or registered trademarks of SPARC International, Inc. in the United States and other countries. THIS PUBLICATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. THIS PUBLICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS.