LogicLoader™ Command Description Manual

A LogicLoader Command Reference
(LogicLoader Version 2.5)

Logic PD // Products
Published: July 2011
Last revised: April 2012

Revision History

REV	EDITOR	REVISION DESCRIPTION	LoLo VER.	APPROVAL	DATE
A	EN	-Initial Release	2.5.0	JCA	07/28/11
B	SO	-Throughout: Removed all references to config block and legacy syntax; General updates	2.5.1	EN, DH	04/03/12

Table of Contents

1 Introduction to LogicLoader™ Commands

This document briefly explains each LogicLoader command. Each command is described within these categories as they apply:

- Purpose
- Usage
- Arguments
- Examples
- Return Value
- Notes

You may view all of the available commands for your System on Module (SOM) by entering *help all* at the `losh>` prompt. Typing the entry *help* at the `losh>` prompt will print a listing of the available sub-menus. The sub-menu listings are intended as a prompt to the user when needed.

1.1 Differences between Versions 2.4 and 2.5

The following are the major differences between LogicLoader v2.4 and v2.5:

- Updates to the *info* command; see Section 2.21
- New *load-var* command; see Section 2.26
- New *make-var* command; see Section 2.28
- New *pin* command; see Section 2.36
- New *save-var* command; see Section 2.41
- New *video-add* command, see Section 2.51

1.2 LogicLoader User Guide

Before reading this document, please reference the *LogicLoader v2.5 User Guide*.[1] This document provides an overview of LogicLoader usage. It also describes the YAFFS file system, video interface, and boot-time scripts.

1.3 Addendums to LogicLoader User Guide

Logic PD has written a SOM-specific addendum to the *LogicLoader v2.5 User Guide* for each SOM that runs LogicLoader. The *LogicLoader User Guide Addendum* is located under the *LogicLoader Bootloader/Monitor* heading on each product's downloads page.[2]

1.4 Glossary of Abbreviations, Acronyms, and Symbols

?	Built-in shell variable holding return value of last command
@	Built-in shell variable holding auxiliary value
< >	Required argument
[ ]	Optional argument
[[ ]]	Optional argument within another optional argument
ARGS	Arguments

[1] http://support.logicpd.com/downloads/1428/
[2] http://support.logicpd.com/auth/

ARGV	Argument vector; all arguments passed to the command
CF	CompactFlash®
BWH	hexadecimal byte, half-word, or word ▪ *b* = 8-bit value ▪ *h* = 16-bit value ▪ *w* = 32-bit value
DST/DEST	Destination
ERRNO	Error number
FOO	Example name meaning *file name*, or *variable name*; its counterpart is *bar*
GPIO	General Purpose Input Output
LEN	Length
LoLo	LogicLoader
LwIP	Lightweight implementation of the TCP/IP protocol stack
MEM	Memory
NFS	Network File System
ODUX	Octal/decimal hexadecimal
RAM	Random Access Memory
RTC	Real-time clock
SRAM	Static Random Access Memory
SRC	Source
STDIN	Standard input stream
STDOUT	Serial port, typical interface to the machine
TFTP	Trivial File Transfer Protocol
TLB	Translation Look-aside Buffer
YAFFS	Yet Another Flash File System

1.5 Command Usage Overview

The losh command set may vary according to the different features and requirements of the different SOMs. In LogicLoader, all commands in all categories that have been implemented for your SOM are always available from the `losh>` prompt.

1.5.1 Losh Command Annotation

Losh commands are listed in Section 2 with required or optional arguments. Arguments required by a command are noted inside angle brackets < >. Arguments optional to the command are designated by square brackets [ ].

For example, the *load* command requires an argument that specifies the type of file to load, but optionally, a user may also specify an input stream or filename. Therefore, the command's syntax is documented as: load <type> [source].

In most cases, optional arguments are filled with default values if not specified by the user. For example, if a user does not specify the *source* argument to the *load* command, the load is assumed to come from the standard input stream (stdin).

1.5.2 Losh Command Return Values

All losh commands return a value. The returned value from the last command executed is stored in the built-in shell variable '?'. A value of zero indicates that the command executed successfully. A non-zero indicates that an error prevented the command from succeeding.

Some commands return a second value. This value is stored in the built-in shell variable '@'.

For complete information concerning shell environment variables, see the *LogicLoader v2.5 User Guide*.

2 Losh Commands

2.1 bitmap

Purpose

This command displays a bitmap on the screen. Only Windows expanded Device Independent Bitmaps are supported. The bitmap should be a standard 4, 8, or 24bpp with no compression. The [address] parameter may be used to specify an alternate frame buffer address in RAM for drawing the bitmap. The *video-fb* command may then be used to load the alternate frame buffer address into the video controller. The [address] parameter must be placed immediately after the <file_name>, before any other optional parameters. The [tl_x, tl_y] and [br_x, br_y] parameters optionally set the top-left x, top-left y, bottom-right x, and bottom-right y coordinates to specify the bounds of the bitmap on the screen. These comma separated parameters refer to an origin at the top-left of the screen. Specifying a bitmap file 0,0 640,480 will display a bitmap which covers an entire 640x480 screen. The default is to show as much of the bitmap as the screen will allow.

Usage

bitmap <file_name> [address] [tl_x,tl_y [br_x,br_y]]

Examples

```
losh> bitmap /cf_card/TEST_FILE.BMP

losh> bitmap /cf_card/TEST_FILE.BMP 0xa0400000

losh> bitmap /cf_card/TEST_FILE.BMP 0,0 640,480
```

2.2 bootme

Purpose

This command initiates the transfer of a Windows CE image between the device and a host PC running Platform Builder.

Usage

bootme

Examples

```
losh> bootme
```

2.3 burn

Purpose

This command burns a loaded image to a device. If no device is specified, it looks up the device based on the load address that was used with the *load* command. If the command fails, it doesn't burn anything. If the command succeeds, it burns the image into this device from 0 offset. If the device is specified, the offset has to be specified as well.

Usage

- burn
- burn [block_device offset]

Examples

```
losh> burn

losh> burn /dev/flash0 0
```

Notes
This command is not supported on all architectures.

2.4 cache-off

Purpose
This command stops the processor's instruction and data cache. On certain processors the data cache must be enabled to access internal SRAM. In that case, it will turn off the instruction cache but leave the data cache enabled.

Usage
cache-off

Examples

```
losh> cache-off
```

Notes
This command always returns 0.

2.5 cache-on

Purpose
This command starts the processor's cache.

Usage
cache-on

Examples

```
losh> cache-on
```

Notes
This command always returns 0.

2.6 cat

Purpose
This command will print the contents of the specified file to stdout.

Usage
cat <file>

Examples

```
losh> cat /cf/foo.txt
```


Notes

This command will work on any type of file. Use the *hd* command to view the contents of binary files.

2.7 cd

Purpose

This command changes the current working directory.

Usage

cd <directory>

Examples

```
losh> cd /dev
```

2.8 cp

Purpose

This command is used to copy a file.

Usage

cp <src> <dest>

Examples

```
losh> cp /cf/foo /yaffs/bar
```

Notes

- The destination file must be on a writable file system.
- This command does **not** work on directories.
- This command sets '$@' to the number of bytes copied.

2.9 cpu-freq

Purpose

This command changes the frequency and core voltage of the processor.

Usage

cpu-freq [high|nominal|low]

Examples

```
losh> cpu-freq high

losh> cpu-freq nominal
```

Notes

This command will set the CPU core clock frequency and the CPU core voltage. The frequency and core voltage setting this command uses is specific to each platform. However, with each setting, this command will indicate what CPU core clock frequency and CPU core voltage are set to.

2.10 date

Purpose

This command displays the number of seconds since boot.

Usage

date

Examples

```
losh> date
```

2.11 dd

Purpose

This command copies segments of memory from the source device (input interface) to the destination device (output interface).

NOTE: The use of the term block in the description of the *dd* command refers to an arbitrary collection of bytes; its use should not be confused with the precise definition of block when pertaining to a NAND flash device.

Usage (all one line)

dd if:<src_device> of:<dst_device> [count:<number_of_blocks> ibs:<ibs> obs:<obs> is:<is> os:<os> seek:<seek> skip:<skip> skip_bad:<0|1> bs:<bs>]

Arguments

- **if**: The source device
- **of**: The destination device
- **count**: The number of blocks to be copied. **NOTE:** If bad blocks are skipped, those bad blocks are *not* included in this count
- **ibs**: Input block size
- **obs**: Output block size
- **is**: Specifies how many bytes should be skipped after an input block
- **os**: Specifies how many bytes should be skipped after an output block
- **seek**: Specifies the starting position of the copy (in blocks) in the output device
- **skip**: Specifies the starting position of the copy (in blocks) in the input device
- **skip_bad**: If set to 0, the command copies bad blocks; if set to 1, the command skips bad blocks
- **bs**: Use instead of *ibs* and *obs*, where *ibs=obs=bs*
- **ber**: Will perform an additional verification step by testing the source image against the destination image and allowing for the indicated number of bit-errors per block

Examples

```
losh> dd if:/dev/nand0 of:/cf/test.dat count:2 ibs:512 obs:512 is:16
os:16 skip_bad:1
```

NOTE: The example above copies two data blocks from the beginning of the NAND device, skipping spare area and skipping bad blocks.

Return Value

This command sets '$@' to the number of bit errors found when using the *ber* argument

Notes

Only the *if* and *of* arguments are obligatory. The other arguments, if not specified, will be set to the following defaults: count=1, skip_bad=1, ibs/obs=512, and all the other arguments are set to 0.

IMPORTANT NOTE: The current version of the *dd* command requires *ibs* and *obs* values to be equal.

2.12 draw-test

Purpose

This command draws framed red, green, blue, and stipple test patterns.

Usage

draw-test

Examples

```
losh> draw-test
```

2.13 echo

Purpose

This command echoes a string to standard output or to a file.

Usage

echo <string> [filename [offset|.]]

Examples

```
losh> echo "Hello world"

losh> echo "Save this string" /dev/serial_eeprom

losh> echo "Append this string" /dev/serial_eeprom.
```

Notes

When successful, sets '$@' to the number of characters written; otherwise, '$@' will be zero.

2.14 erase

Purpose

This command erases non-volatile NOR flash <device> from start_address for <length> bytes, or for NAND flash <device> from <start block> for <number of blocks> blocks. When using a memory-mapped device (such as NOR flash) the <start address> and <length> parameters indicate the memory address and the length in bytes. When using a block device (such as NAND flash) the <start block> and <number of blocks> indicate the first block number and the number of blocks to erase.

NOTE: Some devices, such as NAND flash, are marked with bad blocks by the manufacturer; these blocks will not be erased and the *erase* command will indicate which blocks have been marked bad.

Usage

erase <device> <start_address> <length>

Examples

```
losh> erase /dev/flash0 0x400c0000 1024

losh> erase /dev/serial_eeprom 0 128

losh> erase /dev/nand0 B10 B502
```

Notes

The *erase* command can be used with any block device, not just NAND and NOR flash.

2.15 exec

Purpose

This command allows the processor to jump to an OS image loaded in memory or to a given address. Before the jump, interrupts, memory caching, and mapping are disabled. *-t* is used to modify the calling sequence with the architecture ID in the second argument and create an ARM Linux ATAGS structure that is passed in as the third argument to the kernel. If *-t* is not specified, then the third argument is the command line string.

Usage

exec [-t] [-i arch_id] [-a atag_addr] [addr -] [kernel command line]

Examples

```
losh> exec

losh> exec -t -i 389 0x400c8000 –

losh> exec -t -a 0xc0001000 0x400c8000 - "root=nfs"

losh> exec 0x400c8000 -

losh> exec 0x400c8000 - "root=nfs"

losh> exec "root=/dev/mtd/2 rootfstype=yaffs"

losh> exec -t "root=/dev/mtd/2 rootfstype=yaffs"
```

Notes

This command returns the return value of the executed code and sets ‘$@’ to the jump address when no losh errors occur. The memory for the command line string is taken out of LogicLoaders's internal heap. If there is no command line, then the pointer is to a null-terminated string.

2.16 hd

Purpose

This command prints the contents of a file to stdout in hex format. This command is useful for looking at the contents of binary files.

Usage

hd <filename> [len [offset]]

Examples

```
losh> hd /dev/serial_eeprom
```

Notes

This command functions similarly for both binary and text files. When looking at text, use the *cat* command.

2.17 help

Purpose

This command provides information about the usage of a specific command or group of commands.

Usage

help <test|file|dir|video|net|thread|misc|all|cmd_name>

Examples

```
losh> help dir

losh> help ifmac
```

2.18 if

Purpose

This command alters control flow of a script.

Usage

- The first form of the *if* statement has only a *then* clause that is executed if the conditional expression is true:

  if <conditional_expr>

  <true_statements>

  endif

- The second form of the *if* statement contains both a *then* clause that is executed if the conditional expression is true and an *else* clause that is executed if the conditional expression is false:

  if <conditional_expr>

  <true_statements>

  else

  <false_statements>

  endif

Examples

```
if ($a != 0)
    echo "A not zero"
          b = -1;
      else
          echo "A is zero"
      endif
```

Return Value

The return value is the return of the last statement executed in the body.

Notes

An *if* statement can be nested in the body of either the *then* or *else* clause of an *if* statement.

2.19 ifconfig

Purpose

This command configures a network interface.

Usage

- ifconfig [interface]
- ifconfig <interface> <ip> <netmask> <gw>
- ifconfig <interface> [up|down|dhcp|def]

Examples

```
losh> ifconfig ( display current status of interfaces )

losh> ifconfig sm0 dhcp ( bring up an interface using DHCP )

losh> ifconfig sm0 1.1.1.1 255.255.255.0 1.1.1.0 ( manual configuration )

losh> ifconfig sm0 down  ( bring an interface down )

losh> ifconfig sm0 up ( bring an interface up )
```

2.20 ifmac

Purpose

This command programs and/or displays the MAC address for a network interface. When the user only provides bytes four, five, and six, the first three bytes will default to LogicLoader's 0x00:0x08:0xEE.

Usage

- ifmac <interface> [byte 4:byte 5:byte 6]
- ifmac <interface> [byte 1:byte 2:byte 3:byte 4:byte 5:byte 6]

Examples

```
losh> ifmac sm0

losh> ifmac sm0 0x01:0x1a:0xee
```

2.21 info

Purpose

This command prints information from a chosen category. The available categories are:

- **arch**: This category provides architecture-specific information such as architecture type and revision.
- **cpu**: This category provides CPU status information; information displayed will vary by processor. An example of information includes cacheable space, processor ID, cache size, and cache type.
- **device**: This category provides information about the named device. Device names can be */dev/nand0, /dev/id0*, or any other device in the */dev* directory. Not all devices support displaying info.
- **id**: This category provides information contained in the ID chip. The SOM includes a ROM ID chip that stores manufacturing information such as serial number, Logic PD part number, MAC address, memory sizes, and product features.
- **intr**: This category provides information regarding what interrupts LogicLoader is currently using.
- **mem**: This category provides all information regarding memory size and use. The information displayed here is dynamic, so, for example, only when a video display is activated, will *info mem* reflect video memory in use.
- **net**: This category provides the Ethernet statistics including transmit errors, receive errors, and total packets transferred. A network connection must be established to use this command.
- **part**: This category provides partition information regarding a specific device.
- **pin**: This category provides a list of all the processor pins being used by LogicLoader, including pin ID, GPIO number (if applicable), and pin mux address. Specifying *info pin <pin ID>* will display more detailed information about a specific pin.
- **prot**: This category provides protected area information. The protected areas of memory are areas that LogicLoader is using or areas where active partitions may exist.
- **var**: This category provides information as to what shell variables are being used and what values are stored in them. Optionally, if a variable name is included, *info var <var name>* will display additional meta-data about that variable.
- **version**: This category displays the loaded LogicLoader version, build, and compiler, as well as the SOM model number, part number, and serial number.
- **video**: This category provides a list of displays supported, along with the display number. Specifying *info video <display number>* will show resolution and display controller register settings for that display number.
- **yaffs**: This category displays YAFFS statistics for a specific mount point, including the free space. A YAFFS mount point must be established to use this command.

Usage

info <arch|cpu|device|id|intr|mem|net|part|pin|prot|var|version|video|yaffs>

Examples

```
losh> info version

losh> info prot /dev/nand0
```

2.22 jump

Purpose

This command allows the processor to jump to an image loaded in memory or to a given address.

Usage

jump [addr]

Examples

```
losh> jump

losh> jump 0x40040000
```

Notes

- Returns the value returned from the executed code and sets '$@' to the jump address when no losh errors occur.
- The *jump* command will not turn off cache and interrupts, as is the case with the *exec* command.

2.23 kill

Purpose

This command stops the specified thread(s) from running. Use the *ps* command to view the list of threads.

Usage

kill <thread_id> [thread_id] ...

Examples

```
losh> kill 2 ( stops thread number 2 )
```

Notes

This command returns the last argument number it could not kill or 0 if it was able to kill the entire argument list.

2.24 load

Purpose

This command opens the requested source file to load and then calls out to various binary format parse functions to perform the load. This command works for files of type .bin, .raw, and .elf.

Usage
- load <bin|raw|elf> [source]
- load <bin|raw|elf> /tftp/<server:filename>
- load raw <dest_addr> <length> [source]
- load <bin|raw|elf> -dhcp

Examples

```
losh> load elf
```

2.25 load-buf

Purpose

This command can be used to display or move the load buffer location. When a .bin or .elf file is loaded, that file is initially loaded to a load buffer, the file headers are parsed, and the final image is relocated to the final destination in memory. The *load-buf* command allows the user to move the load buffer in the event that the load buffer conflicts with a user's needs.

Usage

load-buf <address>

Examples

```
losh> load-buf

losh> load-buf 0x81000000
```

2.26 load-var

Purpose

This command is used to load a variable file into LogicLoader. The file can be created using a simple text editor or the *save-var* command. See the *LogicLoader v2.5 User Guide* for information on syntax of a .var file.

Usage

load-var <file>

Examples

```
losh> load-var /sd/lboot.var

losh> load-var /yaffspart/myvars.var
```

2.27 ls

Purpose

This command will list the contents of the current directory, the directory specified on the command line, or the file specified on the command line.

Usage

ls [dir|file]

Examples

```
losh> ls

losh> ls /dev
```

2.28 make-var

Purpose

Shell variables can be created by simply using the '=' operator to assign a value to a name. Doing so creates a global variable of type identified by the value used in the assignment. However, variables can be created with explicit type and protection by using the *make-var* command. By using the *make-var* command, a user can create variables that have unique properties that could not otherwise be created. For example, creating a variable of type 6 will create a variable whose value is the contents of the memory location specified in the *make-var* command. Creating a variable with protection of 8 will prevent the variable from being shown using *info var*; however, the variable continues to exist and is usable in the shell.

Types are specified by the number listed below. Type 3 is reserved and should not be used.

- 1. Number
- 2. String
- 4. Byte pointer
- 5. Word pointer
- 6. Long pointer

The protection field is a bit mask of the numbers below.

- 1. Read only
- 2. Static
- 4. Global
- 8. Hidden

Usage

make-var <name> <type> <value> [protection]

Examples

```
losh> make-var mynumber 1 12345 0

losh> make-var gp_timer2 6 0x49032028 3
```

2.29 md5sum

Purpose

This command calculates an MD5 hash on a file or memory chunk.

Usage

md5sum <filename | address> [read-size]

Examples

```
losh> md5sum foo.txt

losh> md5sum /dev/serial_eeprom
```



```
losh> md5sum 0x0 512
```

Return Value

This command sets '$@' to the MD5SUM obtained.

2.30 mem-cmp

Purpose

This command compares two memory areas over the <length> of bytes specified.

Usage

mem-cmp <addr1> <addr2> <length>

Examples

```
losh> mem-cmp 0xc0000000 0x40000000 1000
```

Notes

This command sets '$@' to the offset where the match failed.

2.31 mem-copy

Purpose

This command copies memory from <src> address to <dst> address for <count>, in sizes of [bhw] (byte, half-word, word). The default size is a word. A source or destination address that is unaligned with the right width will return an error number indicating an unaligned address.

This command can also be used to map a region of memory to a file by specifying a file as the destination. This is useful when there is a need to treat a region of memory as a file. **NOTE:** No memory copying actually takes place in this instance; rather, a file is created in the file system that is physically mapped to the memory region indicated with the *mem-copy* command.

Usage

mem-copy <src> <dst|file> <count> [/bhw]

Examples

```
losh> mem-copy mem-copy 0xc0000 0xd0000 0x100 /h

losh> mem-copy mem-copy 0xc0000 0xd0000 0x100

losh> mem-copy mem-copy 0xc0000 /myfile 0x100
```

2.32 mem-fill

Purpose

This command fills a memory area with a certain value. If the width [/bhw] is not specified, then /w will be assumed.

Usage

mem-fill <addr> <count> <value> [/bhw]

Examples

```
losh> mem-fill 0x200c0000 100 0xac /b

losh> mem-fill 0xc0000 0x1000 0xabcd /h
```

2.33 mount

Purpose

This command mounts a file system of type <fstype> onto LogicLoaders's root filesystem device at mountpoint <mpoin> with optional arguments [-ro] for read-only and [-rw] for read/write. If the *mount* command is successful, you may use other shell commands to access the new file system.

This command can also be used to partition and mount a device with one command. To do this, specify a device rather than a partition. If partitions already exist on the device, the *mount* command will partition all remaining contiguous space and mount it.

Usage

mount <fstype> <device> <mpoin> [-ro|-rw]

Examples

```
losh> mount fatfs /dev/ata0a /cf

losh> mount yaffs /dev/nand0a /rom

losh> mount yaffs /dev/nand0 /data
```

Notes

If [-ro] is not specified, then the mounted file system will be read/write for the FATFS.

2.34 part-add

Purpose

This command creates a partition on a block device.

Usage

part-add <device> <entry> <start block> <block len>

Examples

```
losh> part-add /dev/nand0 a 1 4095

losh> part-add /dev/flash0 a 5 16
```

Notes

There can be up to four partitions on a device labeled from *a* to *d* and the partitions cannot overlap one another.

2.35 part-rem

Purpose

This command removes a partition.

Usage

part-rem <device> <entry>

Examples

```
losh> part-rem /dev/nand0 a
```

Notes

The <entry> parameter specifies which partition entry to remove from the partition table.

2.36 pin

Purpose

This command is used to re-define processor pin usage. This command can also be used to explicitly inhibit LogicLoader from using a pin.

Usage

- pin redefine <pin id> <mux register address> <gpio number/na> <mux mode> [up/down/disable] [high/nominal/low]
- pin dnu <pin id> [set/clear]

Arguments

- **pin id**: LogicLoader assigns a unique number to each pin it uses. This pin ID is not specific to the hardware in use. For example, LogicLoader uses two LED pins to toggle a heartbeat. Pin ID 1 and pin ID 2 are LED 0 and LED 1 pins. Pin IDs are consistent across all SOM's regardless of what CPU is installed. A list of the pin IDs and the associated processor pin mux addresses can be obtained using the command *info pin* or *info pin <pin id>*. By identifying the pin mux register and using the SOM schematic, a user can identify what pin is represented by the pin ID.
- **mux register address**: This is the address of the pin mux register of the CPU. The pin mux register describes exactly each pin on the processor. The mux address register is used here in LogicLoader because a unique pin mux register is used to configure each pin. This is consistent with any processor that is used.
- **gpio number**: This argument is used to identify what GPIO should be used for the indicated pin. Some processors can have a single GPIO mapped to several pins. Specifying a GPIO here associates a GPIO number to the pin at the indicated mux register address.
- **mux mode**: The argument is specific to the CPU used. Some CPUs support a mode for configuration that can be specified with this argument.
- **up/down/disable**: This argument defines the pull type to be used if the CPU supports configuring a pull for each pin.
- **high/nominal/low**: This argument defines the drive strength to be used if the CPU supports configuring drive strength for each pin.
- **dnu**: "DNU" is short for Do Not Use. This argument is used to indicate to LogicLoader that this pin is off limits. Any function within LogicLoader that attempts to use this pin will be ignored, and no access to the physical pin will be performed.

Examples

```
losh> pin redefine 1 0x480021d8 180 4 disable nominal

losh> pin dnu 1
```



```
losh> pin dnu 1 clear
```

2.37 ping

Purpose
This command pings a remote host via the ICMP network protocol.

Usage
ping <ip-address> [reps]

Examples

```
losh> ping 192.168.1.1 ( pings the host once )

losh> ping 192.168.1.1 10 ( pings the host ten times )
```

Notes
The parameter <ip-address> **must** be quoted.

2.38 ps

Purpose
This command displays a list of the currently executing threads.

Usage
ps

Examples

```
losh> ps
```

2.39 pwd

Purpose
This command prints the current working directory to stdout.

Usage
pwd

Examples

```
losh> pwd
```

2.40 reset

Purpose
This command resets the processor. The different types of reset are specific to each platform.

Usage
reset [cold|warm|hard]

Examples

```
losh> reset

losh> reset hard

losh> reset cold
```

Return Value
0 - success

Notes
'$@' will be zero.

2.41 save-var

Purpose
This command will create a text file of all the variables in the shell. This same file can be loaded using the *load-var* command. See the *LogicLoader v2.5 User Guide* for information on syntax of a .var file.

Usage
save-var <file>

Examples

```
losh> save-var /yaffspart1/myvars.var
```

2.42 rm

Purpose
This command removes a file or files.

Usage
rm <file> [file ...]

Examples

```
losh> rm /yaffs/foo
```

Notes
- The file must exist on a writeable file system.
- This command sets '$@' to the number of files removed.

2.43 set

Purpose
The *set* command can be used to modify several internal variables affecting script execution. These function similarly to the Unix shell scripting analog, where a '-' causes the flags noted below to be set, and a '+' causes them to be unset. It is highly recommended during development that you set the *-w* flag to receive warnings about common scripting errors.

The flags available are:

- c Get user confirmation on warnings

- e Exit script execution immediately when commands fail
- i Interactive shell
- g New variables are globally scoped
- m Mute all script output
- n Read commands, but don't execute; ignored by interactive shells
- q Don't print LogicLoader error messages
- u Exit on expansion of unset variables
- v Echo input lines as they are read
- w Print warnings for possible errors
- x Echo all user commands before executing them

Usage

set [+-][ceigmnquvwx]

Examples

```
losh> set +x

losh> set –x
```

2.44 sleep

Purpose

This command sleeps for the given number of milliseconds.

Usage

sleep <ms>

Examples

```
losh> sleep 10
```

2.45 slide-show

Purpose

This command displays a bitmap slide show on the device's screen using a configuration file with each entry having the following format:

<FILE.BMP>[:tl_x,tl_y:br_x,br_y:Tseconds]
Example entries:

FILE.BMP

FILE.BMP:T5

FILE.BMP:5,5:90,90:T3

(The T parameter must be last; it defaults to 3.)

Usage

slide-show <configuration script file name> [repeat]

Examples

```
losh> slide-show CONFIG.TXT
```

2.46 source

Purpose

This command executes a series of *losh-shell* commands listed in <file>. The *s* and *-varname* parameters to this command are deprecated.

Usage

source <file>

Example

```
losh> source /dev/serial_eeprom

losh> source /cf_card/STARTUP
```

2.47 tsleep

Purpose

This command causes its calling thread to sleep for <ms> number of milliseconds.

Usage

tsleep <ms>

Examples

```
losh> tsleep 1
```

2.48 unmount

Purpose

This command creates a YAFFS checkpoint.

Usage

unmount <mountpoint>

Examples

```
losh> unmount /iocf
```

2.49 unset

Purpose

Delete shell variable [varname] to reclaim LogicLoader heap memory.

Usage

unset [varname]

Examples

```
losh> unset my_var
```

2.50 update

Purpose

This command loads and installs an update image.

Usage

update [filename]

Examples

```
losh> update

losh> update /cf_card/1001234_lolo.upd
```

2.51 video-add

Purpose

This command will capture the display controller registers and assign a name to those settings. To display all the video modes supported, including video modes that have been created with *video-add*, see the *info video* command. User displays added using the *video-add* command are volatile. When the power is lost, the added displays will not exist next time LogicLoader has booted. To preserve custom video displays across power cycles, add the *video-add* command to the *lboot.lol* boot script.

Usage

video-add <name> <x axis size> <y axis size>

Examples

```
losh> video-add mylcd 640 480
```

2.52 video-clear

Purpose

This command clears a device's screen to white or an optional color.

Usage

video-clear [r|g|b|y|l]

Examples

```
losh> video-clear  (clear to white)

losh> video-clear r (clear to red)

losh> video-clear l  (clear to black)
```

2.53 video-close

Purpose

This command closes the default video device. It disables power to the display, disables the video controller, and attempts to restore any GPIO connections to their default values.

Usage
video-close

Examples

```
losh> video-close
```

2.54 video-fb

Purpose
This command sets the address for the current frame buffer. This address will be used for drawing and displaying by the other video commands. A display must have been initialized by using either the *video-open* or *video-init* commands before using this command. When no address is given, it just prints the current frame buffer address.

Usage
video-fb [address]

Examples

```
losh> video-fb 0xa0400000
```

Note
This command sets '$@' to the frame buffer address when a valid display exists.

2.55 video-init

Purpose
This command initializes the default video device. It configures all GPIOs for the video controller but does not power up the display or enable the controller. To display an image, it must be followed by a *video-on* command.

Usage
video-init <display> <bpp>

Examples

```
losh> video-init 5 16

losh> video-init myscr 8
```

2.56 video-off

Purpose
This command powers down and disables the video controller. The controller remains configured for the display and may be turned back on again with the *video-on* command.

Usage
video-off

Examples

```
losh> video-off
```


2.57 video-on

Purpose

This command powers up the display and enables the video controller. The *video-init* or *video-open* commands must have been issued prior to using this command.

Usage

video-on

Examples

```
losh> video-on
```

2.58 video-open

Purpose

This command opens the default video device. It configures all GPIO and enables the video controller. It is equivalent to a *video-init* command followed by a *video-on* command.

Usage

video-open <display> <bpp>

Examples

```
losh> video-open 5 16

losh> video-open myscr 8
```

2.59 w

Purpose

This command writes a hex byte, half-word, or word to a memory location, file, or device. The default is word.

Usage

w [/[bhw]] <addr> <data> [filename]

Examples

```
losh> w /w 0x60000000 0x12345678

losh> w /b 0 0x65 /dev/uart0
```

Notes

This command sets '$@' to the location written.

2.60 while

Purpose

This command repeatedly executes a set of statements until the condition is false. The *while* statement first evaluates the <conditional expression> and if true, executes the commands in <body>. Then control passes back to the top where the *while* statement reevaluates the <conditional expression> and breaks out of the loop if it is false.

Usage
- while <conditional expression>
- <body>
- done

Examples

- The following script example computes the Greatest Common Divisor (GCD) of the variables *A* and *B*, leaving the result in *A* and using *tmp* as a temporary variable:

```
while ($B != 0)
tmp = $B
B = $A % $B
A = $tmp
done
```

- The following script example continually examines a value in memory (say an I/O register) once per second and breaks out of the loop if the low bit is zero. It assumes the register address is in the variable REG_ADDR and the register is 16 bits in size:

```
while !(*(short)$REG_ADDR & 1)
      sleep 1000
done
```

Return Value

The return value is the return of the last statement executed in <body_statements>.

Notes

A *while* statement can be nested in the body of a *while* statement.

2.61 x

Purpose

This command examines memory, a file, or a device, with [width] using [format] at an address for a [length], where the following input arguments represent:

- b - 8-bit byte
- h - 16-bit half word
- w - 32-bit word
- o - Octel radix
- d - Signed decimal radix
- u - Unsigned decimal radix
- x - Hexadecimal radix

Usage

x [/[bhw][odux]] <addr> [length] [filename]

Examples

```
losh> x /h 0x40000000 64

losh> x /b 0x40000000 128

losh> x /b 0 0x90 /dev/id0
```


Notes

This command sets '$@' to the value located at the first address read. Even though '$@' is a 32-bit value, the address is read using a cast to the width specified and the unused bits are set to zero.

Appendix A: Error Codes

NOTE: Gaps in the sequence below are due to error numbers that are not currently used. The error codes below provide a general message about the error; the actual error output may contain more specific information.

- 0 - Success
- 1 - Bad argument 1
- 2 - Bad argument 2
- 3 - Bad argument 3
- 4 - Bad argument 4
- 5 - Bad argument 5
- 6 - Bad argument 6
- 7 - Bad argument 7
- 8 - Bad argument 8
- 9 - Bad argument 9
- 10 - Wrong number of arguments
- 11 – Bad argument formatting

- 20 - Directory doesn't exist
- 21 - Unable to mount
- 22 - File or device could not be opened
- 23 - Not a file
- 24 - Failed to read file
- 25 - Failed to write file
- 26 - Unable to create
- 27 - Unable to remove
- 28 - Unable to seek
- 29 - File exists
- 30 - Found no suitable display driver
- 31 - No open display
- 32 - Failed to get window handle
- 33 - Not a Windows Device Independent Bitmap
- 34 - Invalid window
- 35 - Bad frame buffer

- 40 - Config device is not valid

- 42 - Index or name does not have permission

- 50 - Unable initialize network device
- 51 - No active network interfaces
- 52 - Unable to create socket
- 53 - Unable to bind network socket
- 54 – Ethernet option not installed
- 55 - Unable to find network device
- 56 - Unable to get IP address

- 60 - Out of memory
- 61 - Unaligned memory access
- 64 - Address space invalid

- 80 - Loaded image does not fit in flash device
- 81 - Burn failed
- 82 - Erase failed
- 83 - Verify failed

- 90 - Image failed to load
- 91 - Test failed
- 92 - Test timed out

- 100 - Bad checksum
- 101 - File has bad magic
- 102 - File has bad md5sum
- 103 - File for wrong platform
- 104 - File has wrong version
- 105 - File has too many sections
- 106 - Dhcp inactive
- 107 - General load failure
- 108 - No loaded image found
- 109 - Unknown image type
- 110 - Unable to read master header
- 111 - Unable to read section header
- 112 - Unknown record type
- 113 - Wrong arch type

- 114 - File has too many headers

- 120 - Unable to create thread
- 121 - Unable to find thread

- 140 - Partitions are not supported on this device
- 141 - Partition table signature is corrupt
- 142 - Valid partitions are from *a* to *d*
- 143 - Partition would go past the end of the device
- 144 - Couldn't add partition

- 250 - Skipping erase/burn
- 251 - Bitmap error

Appendix B: Pre-defined Shell Variables

These variables are defined at boot time and each platform may differ slightly. Below is a comprehensive list. To find out exactly what variables are defined on your platform, use the LogicLoader command *info var*.

- @
- ?
- NUMBERFORMAT
- LOLO_DATE
- LOLO_ASSEMBLY
- LOLO_CARDENGINE
- LOLO_VERSION
- LOLO_COMPILER
- SYS_BOOT_DEVICE
- SYS_INTEGRITY_PASS
- SYS_INTEGRITY_FAIL
- SYS_LASTKEY
- SYS_TIME_MSECS
- ID_CHIP_PRESENT
- ID_CHIP_VALID
- ID_PART_NUMBER
- ID_MODEL_NUMBER
- ID_SERIAL_NUMBER
- ID_PLATFORM
- ID_SM0_MAC
- ID_SM1_MAC
- ID_FLASH0_SIZE
- ID_FLASH1_SIZE
- ID_NAND0_SIZE
- ID_NAND1_SIZE
- ID_SDRAM0_SIZE
- ID_SDRAM1_SIZE
- ID_PLATFORM_SPECIFIC_BITS
- ID_HW_REV_ENUM
- NET0_DHCP_OBTAINED
- NET0_LINK_UP
- MEM_NAND0_SIZE
- MEM_NAND1_SIZE

- MEM_FLASH0_BASE
- MEM_FLASH0_SIZE
- MEM_FLASH1_BASE
- MEM_FLASH1_SIZE
- MEM_SDRAM0_BASE
- MEM_SDRAM0_SIZE
- MEM_SDRAM1_BASE
- MEM_SDRAM1_SIZE
- MEM_SRAM0_BASE
- MEM_SRAM0_SIZE
- MEM_TEXT_BASE
- MEM_TEXT_END
- MEM_BSS_START
- MEM_BSS_END
- MEM_STACK_START
- MEM_STACK_END
- MEM_TTABLE_START
- MEM_TTABLE_END
- MEM_HEAP_START
- MEM_HEAP_END

