

Open On-Chip Debugger (openocd)

Edition 1.0 for openocd version 1.0
29 December 2007

Table of Contents

OpenOCD	1
About	2
1 Developers	3
2 Building	4
3 Running	6
4 Configuration	7
4.1 Daemon configuration	7
4.2 JTAG interface configuration	7
4.3 parport options	8
4.4 amt_jtagaccel options	9
4.5 ft2232 options	9
4.6 ep93xx options	10
4.7 Target configuration	11
4.7.1 arm7tdmi options	12
4.7.2 arm720t options	12
4.7.3 arm9tdmi options	12
4.7.4 arm920t options	12
4.7.5 arm966e options	12
4.7.6 xscale options	12
4.8 Flash configuration	12
4.8.1 lpc2000 options	12
4.8.2 cfi options	12
4.8.3 at91sam7 options	13
4.8.4 str7 options	13
4.8.5 str9 options	13
4.8.6 str9 options (str9xpec driver)	13
4.8.7 stellaris (LM3Sxxx) options	13
4.8.8 stm32x options	13
5 Commands	14
5.1 Daemon	14
5.1.1 Target state handling	14
5.1.2 Memory access commands	14
5.1.3 Flash commands	15
5.2 Target Specific Commands	17
5.2.1 AT91SAM7 specific commands	17

5.2.2	STR9 specific commands	17
5.2.3	STR9 configuration	17
5.2.4	STR9 option byte configuration	17
5.2.5	STM32x specific commands	17
5.3	Architecture Specific Commands	19
5.3.1	ARMV4/5 specific commands	19
5.3.2	ARM7/9 specific commands	19
5.3.3	ARM920T specific commands	19
5.4	Debug commands	20
5.5	JTAG commands	21
6	Sample Scripts	22
6.1	OMAP5912 Flash Debug	22
6.1.1	Openocd config	22
6.1.2	Openocd init	23
6.2	STR71x Script	23
6.3	STR750 Script	24
6.4	STR912 Script	24
6.5	STR912 comstick	25
6.6	STM32x Script	25
6.7	STM32x Performance Stick	26
6.8	LPC2129 Script	27
6.9	LPC2148 Script	27
6.10	LPC2294 Script	28
6.11	AT91R40008 Script	28
6.12	AT91SAM7s Script	29
6.13	XSCALE IXP42x Script	30
6.14	Cirrus Logic EP9301 Script	30
6.15	Hilscher netX 100 / 500 Script	31
6.16	Marvell/Intel PXA270 Script	32
7	GDB and Openocd	33
7.1	Connecting to gdb	33
7.2	Programming using gdb	33
8	FAQ	34
Appendix A GNU Free Documentation License		
	36
	ADDENDUM: How to use this License for your documents	41
Index	42

OpenOCD

This is edition 1.0 of the openocd manual for version 1.0, 29 December 2007

About

The Open On-Chip Debugger (openocd) aims to provide debugging, in-system programming and boundary-scan testing for embedded target devices. The targets are interfaced using JTAG (IEEE 1149.1) compliant hardware, but this may be extended to other connection types in the future.

Openocd currently supports Wiggler (clones), FTDI FT2232 based JTAG interfaces, the Amontec JTAG Accelerator, and the Gateworks GW1602. It allows ARM7 (ARM7TDMI and ARM720t), ARM9 (ARM920t, ARM922t, ARM926ej-s, ARM966e-s), XScale (PXA25x, IXP42x) and Cortex-M3 (Luminary Stellaris LM3 and ST STM32) based cores to be debugged.

Flash writing is supported for external CFI compatible flashes (Intel and AMD/Spansion command set) and several internal flashes (LPC2000, AT91SAM7, STR7x, STR9x, LM3 and STM32x). Preliminary support for using the LPC3180's NAND flash controller is included.

1 Developers

Openocd has been created by Dominic Rath as part of a diploma thesis written at the University of Applied Sciences Augsburg (<http://www.fh-augsburg.de>). Others interested in improving the state of free and open debug and testing technology are welcome to participate.

Other developers have contributed support for additional targets and flashes as well as numerous bugfixes and enhancements. See the AUTHORS file for regular contributors.

2 Building

You can download the current SVN version with SVN client of your choice from the following repositories:

([svn://svn.berlios.de/openocd/trunk](http://svn.berlios.de/openocd/trunk))

or

(<http://svn.berlios.de/svnroot/repos/openocd/trunk>)

Using the SVN command line client, you could use the following command to fetch the latest version (make sure there is no (non-svn) directory called "openocd" in the current directory):

```
svn checkout svn://svn.berlios.de/openocd/trunk
```

Building the OpenOCD requires a recent version of the GNU autotools. On my build system, I'm using autoconf 2.13 and automake 1.9. For building on Windows, you have to use Cygwin. Make sure that your PATH environment variable contains no other locations with Unix utils (like UnxUtils) - these can't handle the Cygwin paths, resulting in obscure dependency errors (This is an observation I've gathered from the logs of one user - correct me if I'm wrong).

You further need the appropriate driver files, if you want to build support for a FTDI FT2232 based interface:

ftdi2232 libftdi (<http://www.intra2net.com/opensource/ftdi/>)

ftd2xx libftd2xx (<http://www.ftdichip.com/Drivers/D2XX.htm>)

When using the Amontec JTAGkey, you have to get the drivers from the Amontec homepage (www.amontec.com), as the JTAGkey uses a non-standard VID/PID.

Please note that the ftdi2232 variant (using libftdi) isn't supported under Cygwin. You have to use the ftd2xx variant (using FTDI's D2XX) on Cygwin.

In general, the D2XX driver provides superior performance (several times as fast), but has the draw-back of being binary-only - though that isn't as worse, as it isn't a kernel module, only a user space library.

To build OpenOCD (on both Linux and Cygwin), use the following commands:

```
./bootstrap
```

Bootstrap generates the configure script, and prepares building on your system.

```
./configure
```

Configure generates the Makefiles used to build OpenOCD

```
make
```

Make builds the OpenOCD, and places the final executable in ./src/

The configure script takes several options, specifying which JTAG interfaces should be included:

```
-enable-parport
-enable-parport_ppdev
-enable-amtjtagaccel
-enable-ft2232_ftd2xx1
```

¹ Using the latest D2XX drivers from FTDI and following their installation instructions, I had to use '--enable-ft2232_libftd2xx' for the OpenOCD to build properly


```
-enable-ft232rlibftdi  
-with-ftd2xx=/path/to/d2xx/
```

If you want to access the parallel port using the PPDEV interface you have to specify both the ‘`--enable-parport`’ AND the ‘`--enable-parport_ppdev`’ option since the ‘`--enable-parport_ppdev`’ option actually is an option to the parport driver (see <http://forum.sparkfun.com/viewtopic.php?t=3795> for more info).

Cygwin users have to specify the location of the FTDI D2XX package. This should be an absolute path containing no spaces.

Linux users should copy the various parts of the D2XX package to the appropriate locations, i.e. `/usr/include`, `/usr/lib`.

3 Running

The OpenOCD runs as a daemon, waiting for connections from clients (Telnet or GDB). Run with `--help` or `-h` to view the available command line arguments.

It reads its configuration by default from the file `openocd.cfg` located in the current working directory. This may be overwritten with the `-f <configfile>` command line switch.

To enable debug output (when reporting problems or working on OpenOCD itself), use the `-d` command line switch. This sets the `debug_level` to "3", outputting the most information, including debug messages. The default setting is "2", outputting only informational messages, warnings and errors. You can also change this setting from within a telnet or gdb session (`debug_level <n>`).

You can redirect all output from the daemon to a file using the `-l <logfile>` switch.

4 Configuration

The Open On-Chip Debugger (OpenOCD) runs as a daemon, and reads its current configuration by default from the file `openocd.cfg` in the current directory. A different configuration file can be specified with the `-f <conf.file>` given at the `openocd` command line.

The configuration file is used to specify on which ports the daemon listens for new connections, the JTAG interface used to connect to the target, the layout of the JTAG chain, the targets that should be debugged, and connected flashes.

4.1 Daemon configuration

telnet_port *<number>* Port on which to listen for incoming telnet connections

gdb_port *<number>* First port on which to listen for incoming GDB connections. The GDB port for the first target will be `gdb_port`, the second target will listen on `gdb_port + 1`, and so on.

gdb_detach *<resume|reset|halt|nothing>* Configures what `openocd` will do when `gdb` detaches from the daemon. Default behaviour is *<resume>*

gdb_memory_map *<enable|disable>* Set to *<enable>* so that `openocd` will send the memory configuration to `gdb` when requested. `gdb` will then know when to set hardware breakpoints, and program flash using the `gdb load` command. `'gdb_flash_program enable'` will also need enabling for flash programming to work. Default behaviour is *<disable>*

gdb_flash_program *<enable|disable>* Set to *<enable>* so that `openocd` will program the flash memory when a `vFlash` packet is received. Default behaviour is *<disable>*

daemon_startup *<mode>* either `'attach'` or `'reset'` Tells the OpenOCD whether it should reset the target when the daemon is launched, or if it should just attach to the target.

4.2 JTAG interface configuration

interface *<name>* Use the interface driver *<name>* to connect to the target. Currently supported interfaces are

`parport` PC parallel port bit-banging (Wigglers, PLD download cable, ...)

`amt_jtagaccel` Amontec Chameleon in its JTAG Accelerator configuration connected to a PC's EPP mode parallel port

`ft2232` FTDI FT2232 based devices using either the open-source `libftdi` or the binary only `FTD2XX` driver. The `FTD2XX` is superior in performance, but not available on every platform. The `libftdi` uses `libusb`, and should be portable to all systems that provide `libusb`.

`ep93xx` Cirrus Logic EP93xx based single-board computer bit-banging (in development)

jtag_speed *<number>* Limit the maximum speed of the JTAG interface. Usually, a value of zero means maximum speed. The actual effect of this option depends on the JTAG interface used.

reset_config *<signals>* [*combination*] [*trst_type*] [*srst_type*] The configuration of the reset signals available on the JTAG interface AND the target. If the JTAG interface provides SRST, but the target doesn't connect that signal properly, then OpenOCD can't use it. *<signals>* can be 'none', 'trst_only', 'srst_only' or 'trst_and_srst'. [*combination*] is an optional value specifying broken reset signal implementations. 'srst_pulls_trst' states that the testlogic is reset together with the reset of the system (e.g. Philips LPC2000, "broken" board layout), 'trst_pulls_srst' says that the system is reset together with the test logic (only hypothetical, I haven't seen hardware with such a bug, and can be worked around).

The [*trst_type*] and [*srst_type*] parameters allow the driver type of the reset lines to be specified. Possible values are 'trst_push_pull' (default) and 'trst_open_drain' for the test reset signal, and 'srst_open_drain' (default) and 'srst_push_pull' for the system reset. These values only affect JTAG interfaces with support for different drivers, like the Amontec JTAGkey and JTAGAccelerator.

jtag_device *<IR length>* *<IR capture>* *<IR mask>* *<IDCODE instruction>* Describes the devices that form the JTAG daisy chain, with the first device being the one closest to TDO. The parameters are the length of the instruction register (4 for all ARM7/9s), the value captured during Capture-IR (0x1 for ARM7/9), and a mask of bits that should be validated when doing IR scans (all four bits (0xf) for ARM7/9). The IDCODE instruction will in future be used to query devices for their JTAG identification code. This line is the same for all ARM7 and ARM9 devices. Other devices, like CPLDs, require different parameters. An example configuration line for a Xilinx XC9500 CPLD would look like this:

```
jtag_device 8 0x01 0x0e3 0xfe
```

The instruction register (IR) is 8 bits long, during Capture-IR 0x01 is loaded into the IR, but only bits 0-1 and 5-7 should be checked, the others (2-4) might vary. The IDCODE instruction is 0xfe.

jtag_nsrst_delay *<ms>* How long (in milliseconds) the OpenOCD should wait after deasserting nSRST before starting new JTAG operations.

jtag_ntrst_delay *<ms>* How long (in milliseconds) the OpenOCD should wait after deasserting nTRST before starting new JTAG operations.

The jtag-n[st]rst_delay options are useful if reset circuitry (like a reset supervisor, or on-chip features) keep a reset line asserted for some time after the external reset got deasserted.

4.3 parport options

parport_port *<number>* Either the address of the I/O port (default: 0x378 for LPT1) or the number of the '/dev/parport' device

When using PPDEV to access the parallel port, use the number of the parallel port: 'parport_port 0' (the default). If 'parport_port 0x378' is specified you may encounter a problem.

parport_cable *<name>* The layout of the parallel port cable used to connect to the target. Currently supported cables are

- wiggler Original Wiggler layout, also supported by several clones, such as the Olimex ARM-JTAG

old_amt_wiggler The Wiggler configuration that comes with Amontec's Chameleon Programmer. The new version available from the website uses the original Wiggler layout (**'wiggler'**)

chameleon Describes the connection of the Amontec Chameleon's CPLD when operated in configuration mode. This is only used to program the Chameleon itself, not a connected target.

dlc5 Xilinx Parallel cable III.

triton The parallel port adapter found on the 'Karo Triton 1 Development Board'. This is also the layout used by the HollyGates design (see <http://www.lartmaker.nl/projects/jtag/>).

flashlink ST Parallel cable.

parport_write_on_exit **<on|o >** This will configure the parallel driver to write a known value to the parallel interface on exiting openocd

4.4 amt_jtagaccel options

parport_port **<number>** Either the address of the I/O port (default: 0x378 for LPT1) or the number of the '/dev/parport' device

4.5 ft2232 options

ft2232_device_desc **<description>** The USB device description of the FTDI FT2232 device. If not specified, the FTDI default value is used. This setting is only valid if compiled with FTD2XX support.

ft2232_layout **<name>** The layout of the FT2232 GPIO signals used to control output-enables and reset signals. Valid layouts are

usbjtag The "USBTAG-1" layout described in the original OpenOCD diploma thesis

jtagkey Amontec JTAGkey and JTAGkey-tiny

signalyzer Signalyzer

olimex-jtag Olimex ARM-USB-OCD

m5960 American Microsystems M5960

evb_lm3s811 Luminary Micro EVB-LM3S811 as a JTAG interface (not onboard processor), no TRST or SRST signals on external connector

comstick Hitex STR9 comstick

stm32stick Hitex STM32 Performance Stick

ft2232_vid_pid **<vid> <pid>** The vendor ID and product ID of the FTDI FT2232 device. If not specified, the FTDI default values are used. This command is not available on Windows.

ft2232_latency **<ms>** On some systems using ft2232 based JTAG interfaces the FT_Read function call in ft2232_read() fails to return the expected number of bytes. This can be caused by USB communication delays and has proved hard to reproduce and debug. Setting the FT2232 latency timer to a larger value increases delays for short USB packages but it also reduces the risk of timeouts before receiving the expected number

of bytes. The OpenOCD default value is 2 and for some systems a value of 10 has proved useful.

4.6 ep93xx options

Currently, there are no options available for the ep93xx interface.

4.7 Target configuration

target *<type>* *<endianess>* *<reset_mode>* *<JTAG pos>* *<variant>* Defines a target that should be debugged. Currently supported types are:

```
arm7tdmi
arm720t
arm9tdmi
arm920t
arm922t
arm926ejs
arm966e
cortex_m3
xscale
```

If you want to use a target board that is not on this list, see Adding a new target board. Endianess may be ‘little’ or ‘big’.

The `reset_mode` specifies what should happen to the target when a reset occurs:

`reset_halt` Immediately request a target halt after reset. This allows targets to be debugged from the very first instruction. This is only possible with targets and JTAG interfaces that correctly implement the reset signals.

`reset_init` Similar to ‘`reset_halt`’, but executes the script file defined to handle the ‘reset’ event for the target. Like ‘`reset_halt`’ this only works with correct reset implementations.

`reset_run` Simply let the target run after a reset.

`run_and_halt` Let the target run for some time (default: 1s), and then request halt.

`run_and_init` A combination of ‘`reset_init`’ and ‘`run_and_halt`’. The target is allowed to run for some time, then halted, and the ‘reset’ event script is executed.

On JTAG interfaces / targets where system reset and test-logic reset can’t be driven completely independent (like the LPC2000 series), or where the JTAG interface is unavailable for some time during startup (like the STR7 series), you can’t use ‘`reset_halt`’ or ‘`reset_init`’.

target_script *<target#>* *<event>* *<script_ le>* Event is either ‘reset’, ‘post_halt’, ‘pre_resume’ or ‘gdb_program_config’

TODO: describe exact semantic of events

run_and_halt_time *<target#>* *<time_in_ms>* The amount of time the debugger should wait after releasing reset before it asserts a debug request. This is used by the ‘`run_and_halt`’ and ‘`run_and_init`’ reset modes.

working_area *<target#>* *<address>* *<size>* *<backup|nobackup>* Specifies a working area for the debugger to use. This may be used to speed-up downloads to target memory and flash operations, or to perform otherwise unavailable operations (some coprocessor operations on ARM7/9 systems, for example). The last parameter decides whether the memory should be preserved *<backup>*. If possible, use a working_area that doesn’t need to be backed up, as that slows down operation.

4.7.1 arm7tdmi options

target arm7tdmi *<endianess>* *<reset_mode>* *<jtag#>* The arm7tdmi target definition requires at least one additional argument, specifying the position of the target in the JTAG daisy-chain. The first JTAG device is number 0. The optional [*variant*] parameter has been removed in recent versions. The correct feature set is determined at runtime.

4.7.2 arm720t options

ARM720t options are similar to ARM7TDMI options.

4.7.3 arm9tdmi options

ARM9TDMI options are similar to ARM7TDMI options. Supported variants are ‘arm920t’, ‘arm922t’ and ‘arm940t’. This enables the hardware single-stepping support found on these cores.

4.7.4 arm920t options

ARM920t options are similar to ARM9TDMI options.

4.7.5 arm966e options

ARM966e options are similar to ARM9TDMI options.

4.7.6 xscale options

Supported variants are ‘ixp42x’, ‘ixp45x’, ‘ixp46x’, ‘pxa250’, ‘pxa255’, ‘pxa26x’.

4.8 Flash configuration

flash bank *<driver>* *<base>* *<size>* *<chip_width>* *<bus_width>* *<target#>* [*driver_options* ...] Configures a flash bank at *<base>* of *<size>* bytes and *<chip_width>* and *<bus_width>* bytes using the selected flash *<driver>*.

flash autoerase *<‘on’|‘off’>* auto erase flash banks prior to writing. Currently only works when using ‘flash write_image’ command. Default is ‘off’.

4.8.1 lpc2000 options

flash bank lpc2000 *<base>* *<size>* 0 0 *<target#>* *<variant>* *<clock>* [*calc_checksum*] LPC flashes don’t require the chip and bus width to be specified. Additional parameters are the *<variant>*, which may be *lpc2000_v1* (older LPC21xx and LPC22xx) or *lpc2000_v2* (LPC213x, LPC214x, LPC210[123], LPC23xx and LPC24xx), the number of the target this flash belongs to (first is 0), the frequency at which the core is currently running (in kHz - must be an integral number), and the optional keyword *calc_checksum*, telling the driver to calculate a valid checksum for the exception vector table.

4.8.2 cfi options

flash bank cfi *<base>* *<size>* *<chip_width>* *<bus_width>* *<target#>* CFI flashes require the number of the target they’re connected to as an additional argument. The CFI driver makes use of a working area (specified for the target) to significantly speed up operation.

4.8.3 at91sam7 options

flash bank at91sam7 0 0 0 0 <target#> AT91SAM7 flashes only require the target#, all other values are looked up after reading the chip-id and type.

4.8.4 str7 options

flash bank str7x <base> <size> 0 0 <target#> <variant> variant can be either STR71x, STR73x or STR75x.

4.8.5 str9 options

flash bank str9x <base> <size> 0 0 <target#> The str9 needs the flash controller to be configured prior to Flash programming, eg.

str9x flash_config 0 4 2 0 0x80000

This will setup the BBSR, NBBSR, BBADR and NBBADR registers respectively.

4.8.6 str9 options (str9xpec driver)

flash bank str9xpec <base> <size> 0 0 <target#> Before using the flash commands the turbo mode will need enabling using str9xpec 'enable_turbo' <num>.

Only use this driver for locking/unlocking the device or configuring the option bytes. Use the standard str9 driver for programming.

4.8.7 stellaris (LM3Sxxx) options

flash bank stellaris <base> <size> 0 0 <target#> stellaris flash plugin only require the target#.

4.8.8 stm32x options

flash bank stm32x <base> <size> 0 0 <target#> stm32x flash plugin only require the target#.

5 Commands

The Open On-Chip Debugger (OpenOCD) allows user interaction through a telnet interface (default: port 4444) and a GDB server (default: port 3333). The command line interpreter is available from both the telnet interface and a GDB session. To issue commands to the interpreter from within a GDB session, use the ‘**monitor**’ command, e.g. use ‘**monitor poll**’ to issue the ‘**poll**’ command. All output is relayed through the GDB session.

5.1 Daemon

sleep <*msec*> Wait for *n* milliseconds before resuming. Useful in connection with script files (**script** command and **target_script** configuration).

shutdown Close the OpenOCD daemon, disconnecting all clients (GDB, Telnet).

debug_level [*n*] Display or adjust debug level to *n*<0-3>

log_output < *le*> Redirect logging to <file> (default: stderr)

script < *le*> Execute commands from <file>

5.1.1 Target state handling

poll [‘on’|‘off’] Poll the target for its current state. If the target is in debug mode, architecture specific information about the current state are printed. An optional parameter allows continuous polling to be enabled and disabled.

halt Send a halt request to the target. The debugger signals the debug request, and waits for the target to enter debug mode.

resume [*address*] Resume the target at its current code position, or at an optional address.

step [*address*] Single-step the target at its current code position, or at an optional address.

reset [‘run’|‘halt’|‘init’|‘run_and_halt’|‘run_and_init’] Do a hard-reset. The optional parameter specifies what should happen after the reset. This optional parameter overwrites the setting specified in the configuration file, making the new behaviour the default for the ‘**reset**’ command.

run Let the target run.

halt Immediately halt the target (works only with certain configurations).

init Immediately halt the target, and execute the reset script (works only with certain configurations)

run_and_halt Let the target run for a certain amount of time, then request a halt.

run_and_init Let the target run for a certain amount of time, then request a halt. Execute the reset script once the target entered debug mode.

5.1.2 Memory access commands

These commands allow accesses of a specific size to the memory system:

mdw <*addr*> [*count*] display memory words

mdh <*addr*> [*count*] display memory half-words

mdb *<addr>* [*count*] display memory bytes

mww *<addr>* *<value>* write memory word

mwh *<addr>* *<value>* write memory half-word

mwb *<addr>* *<value>* write memory byte

load_image *<le>* *<address>* [*'bin'*|*'ihex'*|*'elf'*] Load image *<le>* to target memory at *<address>*

dump_image *<le>* *<address>* *<size>* Dump *<size>* bytes of target memory starting at *<address>* to a (binary) *<le>*.

verify_image *<le>* *<address>* [*'bin'*|*'ihex'*|*'elf'*] Verify *<le>* to target memory starting at *<address>*. This will first attempt using a crc checksum, if this fails it will try a binary compare.

load_binary *<le>* *<address>* [DEPRECATED] Load binary *<le>* to target memory at *<address>*

dump_binary *<le>* *<address>* *<size>* [DEPRECATED] Dump *<size>* bytes of target memory starting at *<address>* to a (binary) *<le>*.

5.1.3 Flash commands

flash banks List configured flash banks

flash info *<num>* Print info about flash bank *<'num'>*

flash probe *<num>* Identify the flash, or validate the parameters of the configured flash. Operation depends on the flash type.

flash erase_check *<num>* Check erase state of sectors in flash bank *<num>*. This is the only operation that updates the erase state information displayed by *'flash info'*. That means you have to issue an *'erase_check'* command after erasing or programming the device to get updated information.

flash protect_check *<num>* Check protection state of sectors in flash bank *<num>*.

flash erase *<num>* *<rst>* *<last>* [DEPRECATED] Erase sectors at bank *<num>*, starting at sector *<rst>* up to and including *<last>*. Sector numbering starts at 0. Depending on the flash type, erasing might require the protection to be disabled first (e.g. Intel Advanced Bootblock flash using the CFI driver). This command was replaced by the new command *'flash erase_sector'* using the same syntax.

flash erase_sector *<num>* *<rst>* *<last>* Erase sectors at bank *<num>*, starting at sector *<rst>* up to and including *<last>*. Sector numbering starts at 0. Depending on the flash type, erasing might require the protection to be disabled first (e.g. Intel Advanced Bootblock flash using the CFI driver).

flash erase.address *<address>* *<length>* Erase sectors starting at *<address>* for *<length>* number of bytes

flash write *<num>* *<le>* *<o set>* [DEPRECATED] Write the binary *<le>* to flash bank *<num>*, starting at *<o set>* bytes from the beginning of the bank. This command was replaced by the new command *'flash write_binary'* using the same syntax.

flash write_binary *<num>* *<le>* *<o set>* Write the binary *<le>* to flash bank *<num>*, starting at *<'offset'>* bytes from the beginning of the bank.

flash write_image < *le* > [*o set*] [*type*] Write the image < *le* > to the current target's flash bank(s). A relocation [*o set*] can be specified and the file [*type*] can be specified explicitly as 'bin' (binary), 'ihex' (Intel hex), 'elf' (ELF file) or 's19' (Motorola s19).

flash protect < *num* > < *rst* > < *last* > < 'on' | 'off' > Enable (*on*) or disable (*o*) protection of flash sectors < *rst* > to < *last* > of 'flash bank' < *num* >.

flash auto_erase < *on* | *o* > Enable ('on') to erase flash banks prior to writing using the flash 'write_image' command only. Default is ('off'), flash banks have to be erased using 'flash erase' command.

5.2 Target Specific Commands

5.2.1 AT91SAM7 specific commands

The flash configuration is deduced from the chip identification register. The flash controller handles erases automatically on a page (128/265 byte) basis so erase is not necessary for flash programming. AT91SAM7 processors with less than 512K flash only have a single flash bank embedded on chip. AT91SAM7xx512 have two flash planes that can be erased separately. Only an EraseAll command is supported by the controller for each flash plane and this is called with

flash erase <num> *rst_plane last_plane* bulk erase flash planes first_plane to last_plane.
at91sam7 gpnvm <num> <bit> <'set'|'clear'> set or clear a gpnvm bit for the processor

5.2.2 STR9 specific commands

These are flash specific commands when using the str9xpec driver.

str9xpec enable_turbo <num> enable turbo mode, simply this will remove the str9 from the chain and talk directly to the embedded flash controller.

str9xpec disable_turbo <num> restore the str9 into jtag chain.

str9xpec lock <num> lock str9 device. The str9 will only respond to an unlock command that will erase the device.

str9xpec unlock <num> unlock str9 device.

str9xpec options_read <num> read str9 option bytes.

str9xpec options_write <num> write str9 option bytes.

5.2.3 STR9 configuration

str9x flash_config <bank> <BBSR> <NBBSR> <BBADR> <NBBADR> Configure str9 flash controller.

eg. str9x flash_config 0 4 2 0 0x80000
This will setup
BBSR - Boot Bank Size register
NBBSR - Non Boot Bank Size register
BBADR - Boot Bank Start Address register
NBBADR - Boot Bank Start Address register

5.2.4 STR9 option byte configuration

str9xpec options_cmap <num> <'bank0'|'bank1'> configure str9 boot bank.

str9xpec options_lvdthd <num> <'2.4v'|'2.7v'> configure str9 lvd threshold.

str9xpec options_lvdsel <num> <'vdd'|'vdd_vddq'> configure str9 lvd source.

str9xpec options_lvdwarn <bank> <'vdd'|'vdd_vddq'> configure str9 lvd reset warning source.

5.2.5 STM32x specific commands

These are flash specific commands when using the stm32x driver.

stm32x lock <num> lock stm32 device.

stm32x unlock <num> unlock stm32 device.

stm32x options_read *<num>* read stm32 option bytes.

stm32x options_write *<num>* *<'SWWDG'|'HWWDG'>* *<'RSTSTNDBY'|'NORSTSTNDBY'>*
<'RSTSTOP'|'NORSTSTOP'> write stm32 option bytes.

stm32x mass_erase *<num>* mass erase flash memory.

5.3 Architecture Specific Commands

5.3.1 ARMV4/5 specific commands

These commands are specific to ARM architecture v4 and v5, like all ARM7/9 systems or Intel XScale (XScale isn't supported yet).

armv4.5 reg Display a list of all banked core registers, fetching the current value from every core mode if necessary. OpenOCD versions before rev. 60 didn't fetch the current register value.

armv4.5 core_mode [*'arm'*|*'thumb'*] Displays the core_mode, optionally changing it to either ARM or Thumb mode. The target is resumed in the currently set *'core_mode'*.

5.3.2 ARM7/9 specific commands

These commands are specific to ARM7 and ARM9 targets, like ARM7TDMI, ARM720t, ARM920t or ARM926EJ-S.

arm7.9 sw_bkpts <*'enable'*|*'disable'*> Enable/disable use of software breakpoints. On ARMv4 systems, this reserves one of the watchpoint registers to implement software breakpoints. Disabling SW Bkpts frees that register again.

arm7.9 force_hw_bkpts <*'enable'*|*'disable'*> When *'force_hw_bkpts'* is enabled, the *'sw_bkpts'* support is disabled, and all breakpoints are turned into hardware breakpoints.

arm7.9 dbgrrq <*'enable'*|*'disable'*> Enable use of the DBGRRQ bit to force entry into debug mode. This should be safe for all but ARM7TDMI-S cores (like Philips LPC).

arm7.9 fast_writes <*'enable'*|*'disable'*> See *'arm7_9 fast_memory_access'* instead.

arm7.9 fast_memory_access <*'enable'*|*'disable'*> Allow the OpenOCD to read and write memory without checking completion of the operation. This provides a huge speed increase, especially with USB JTAG cables (FT2232), but might be unsafe if used with targets running at a very low speed, like the 32kHz startup clock of an AT91RM9200.

arm7.9 dcc_downloads <*'enable'*|*'disable'*> Enable the use of the debug communications channel (DCC) to write larger (>128 byte) amounts of memory. DCC downloads offer a huge speed increase, but might be potentially unsafe, especially with targets running at a very low speed. This command was introduced with OpenOCD rev. 60.

5.3.3 ARM920T specific commands

arm920t cache_info Print information about the caches found. This allows you to see if your target is a ARM920T (2x16kByte cache) or ARM922T (2x8kByte cache).

arm920t md<*bhw*>_phys <*addr*> [*count*] Display memory at physical address *addr*.

arm920t mw<*bhw*>_phys <*addr*> <*value*> Write memory at physical address *addr*.

arm920t read_cache < *lename*> Dump the content of ICache and DCache to a file.

arm920t read_mmu < *lename*> Dump the content of the ITLB and DTLB to a file.

arm920t virt2phys <*VA*> Translate a virtual address to a physical address.

5.4 Debug commands

The following commands give direct access to the core, and are most likely only useful while debugging the OpenOCD.

arm7_9 write_xpsr *<32-bit value>* *<'0=cpsr', '1=spsr'>* Immediately write either the current program status register (CPSR) or the saved program status register (SPSR), without changing the register cache (as displayed by the **'reg'** and **'armv4_5 reg'** commands).

arm7_9 write_xpsr_im8 *<8-bit value>* *<rotate 4-bit>* *<0=cpsr,1=spsr>* Write the 8-bit value rotated right by 2*rotate bits, using an immediate write operation (similar to **'write_xpsr'**).

arm7_9 write_core_reg *<num>* *<mode>* *<value>* Write a core register, without changing the register cache (as displayed by the **'reg'** and **'armv4_5 reg'** commands). The *<mode>* argument takes the encoding of the [M4:M0] bits of the PSR.

5.5 JTAG commands

scan_chain Print current scan chain configuration.

jtag_reset Toggle reset lines *<trst>* *<srst>*.

endstate *<tap_state>* Finish JTAG operations in *<tap_state>*.

runtest *<num_cycles>* Move to Run-Test/Idle, and execute *<num_cycles>*

statemove [*tap_state*] Move to current endstate or [*tap_state*]

irscan Execute IR scan *<device>* *<instr>* [*dev2*] [*instr2*] ...

drscan Execute DR scan *<device>* [*dev2*] [*var2*] ...

verify_ircapture Verify value captured during Capture-IR *<'enable'|'disable'>*

var Allocate, display or delete variable *<name>* [*num_ elds|del*] [*size1*] ...

field Display/modify variable field *<var>* *< eld>* [*value| ip*]

6 Sample Scripts

This page will collect some script examples for different CPUs.

The configuration script can be divided in the following section:

- daemon configuration
- interface
- jtag scan chain
- target configuration
- flash configuration

Detailed information about each section can be found at OpenOCD configuration

6.1 OMAP5912 Flash Debug

The following two scripts were used with a wiggler PP and and a TI OMAP5912 dual core processor - (<http://www.ti.com>), on a OMAP5912 OSK board - (<http://www.spectrumdigital.com>).

6.1.1 Openocd config

```
#daemon configuration
telnet_port 4444
gdb_port 3333

#interface
interface parport
parport_port 0x378
parport_cable wiggler
jtag_speed 0

#use combined on interfaces or targets that can't set TRST/SRST separately
reset_config trst_and_srst

#jtag scan chain
#format I IRC IRCM IDCODE (Length, IR Capture, IR Capture Mask, IDCODE)
jtag_device 38 0x0 0x0 0x0
jtag_device 4 0x1 0x0 0xe
jtag_device 8 0x0 0x0 0x0

#target configuration
daemon_startup reset

#target <type> <endianmess> <reset mode> <chainpos> <variant>
target arm926ejs little run_and_init 1 arm926ejs
target_script 0 reset omap5912_osk.init
run_and_halt_time 0 30

# omap5912 lcd frame buffer as working area
working_area 0 0x20000000 0x3e800 nobackup

#flash bank <driver> <base> <size> <chip_width> <bus_width>
flash bank cfi 0x00000000 0x1000000 2 2 0
```

6.1.2 Openocd init

```
#
# halt target
#
poll
sleep 1
halt
wait_halt
#
# disable wdt
#
mw 0xffffec808 0x000000f5
mw 0xffffec808 0x000000a0

mw 0xffffeb048 0x0000aaaa
sleep 500
mw 0xffffeb048 0x00005555
sleep 500
#
# detect flash
#
flash probe 0
```

6.2 STR71x Script

The following script was used with an Amontec JTAGkey and a STR710 / STR711 cpu:

```
#daemon configuration
telnet_port 4444
gdb_port 3333

#interface
interface ft2232
ft2232_device_desc "Amontec JTAGkey A"
ft2232_layout jtagkey
ft2232_vid_pid 0x0403 0xcff8
jtag_speed 0

#use combined on interfaces or targets that can't set TRST/SRST separately
reset_config trst_and_srst srst_pulls_trst

#jtag scan chain
#format L IRC IRCMIDCODE (Length, IR Capture, IR Capture Mask, IDCODE)
jtag_device 4 0x1 0xf 0xe

#target configuration
daemon_startup reset

#target <type> <startup mode>
#target arm7tdmi <endianness> <reset mode> <chainpos> <variant>
target arm7tdmi little run_and_halt 0 arm7tdmi
run_and_halt_time 0 30

working_area 0 0x2000C000 0x4000 nobackup

#flash bank <driver> <base> <size> <chip_width> <bus_width>
flash bank str7x 0x40000000 0x00040000 0 0 0 STR71x
```

6.3 STR750 Script

The following script was used with an Amontec JTAGkey and a STR750 cpu:

```
#daemon configuration
telnet_port 4444
gdb_port 3333

#interface
interface ft2232
ft2232_device_desc "Amontec JTAGkey A"
ft2232_layout jtagkey
ft2232_vid_pid 0x0403 0xcff8
jtag_speed 19

#use combined on interfaces or targets that can't set TRST/SRST separately
#reset_config trst_and_srst srst_pulls_trst
reset_config trst_and_srst srst_pulls_trst

#jtag scan chain
#format L IRC IRCMIDCODE (Length, IR Capture, IR Capture Msk, IDCODE)
jtag_device 4 0x1 0xf 0xe

#jtag nTRST and nSRST delay
jtag_nsrst_delay 500
jtag_ntrst_delay 500

#target configuration
daemon_startup reset

#target <type> <startup node>
#target arn7tdmi <reset node> <chainpos> <endianness> <variant>
target arn7tdmi little run_and_halt 0 arn7tdmi
run_and_halt_time 0 30

working_area 0 0x40000000 0x4000 nobackup

#flash bank <driver> <base> <size> <chip_width> <bus_width>
flash bank str7x 0x20000000 0x00004000 0 0 0 STR75x
```

6.4 STR912 Script

The following script was used with an Amontec JTAGkey and a STR912 cpu:

```
#daemon configuration
telnet_port 4444
gdb_port 3333

#interface
interface ft2232
ft2232_device_desc "Amontec JTAGkey A"
ft2232_layout jtagkey
jtag_speed 1

#use combined on interfaces or targets that can't set TRST/SRST separately
reset_config trst_and_srst

#jtag scan chain
#format L IRC IRCMIDCODE (Length, IR Capture, IR Capture Msk, IDCODE)
jtag_device 8 0x1 0x1 0xfe
```

```

jtag_device 4 0x1 0xf 0xe
jtag_device 5 0x1 0x1 0x1e

#target configuration
daemon_startup reset

#target <type> <startup node>
#target arn966e <endianmess> <reset node> <chainpos> <variant>
target arn966e little reset_halt 1 arn966e
run_and_halt_time 0 30

working_area 0 0x50000000 16384 nobackup

#flash bank <driver> <base> <size> <chip_width> <bus_width>
flash bank str9x 0x00000000 0x00080000 0 0 0

```

6.5 STR912 comstick

The following script was used with a Hitex STR9 Comstick:

```

#daemon configuration
telnet_port 4444
gdb_port 3333

#interface
interface ft2232
ft2232_device_desc "STR9-comstick A"
ft2232_layout comstick
jtag_speed 1

jtag_nsrst_delay 100
jtag_nrst_delay 100

#use combined on interfaces or targets that can't set TRST/SRST separately
reset_config trst_and_srst

#jtag scan chain
#format L IRC IRCM IDCODE (Length, IR Capture, IR Capture Mask, IDCODE)
jtag_device 8 0x1 0x1 0xfe
jtag_device 4 0x1 0xf 0xe
jtag_device 5 0x1 0x1 0x1e

#target configuration
daemon_startup reset

#target <type> <startup node>
#target arn966e <endianmess> <reset node> <chainpos> <variant>
target arn966e little reset_halt 1 arn966e
run_and_halt_time 0 30

working_area 0 0x50000000 16384 nobackup

#flash bank <driver> <base> <size> <chip_width> <bus_width>
flash bank str9x 0x00000000 0x00080000 0 0 0

```

6.6 STM32x Script

The following script was used with an Amontec JTAGkey and a STM32x cpu:

```

#daemon configuration

```

```

telnet_port 4444
gdb_port 3333

#interface
interface ft2232
ft2232_device_desc "Anontec JTAGkey A"
ft2232_layout jtagkey
jtag_speed 10

jtag_nsrst_delay 100
jtag_ntrst_delay 100

#use combined on interfaces or targets that can't set TRST/SRST separately
reset_config trst_and_srst

#jtag scan chain
#format L IRC IRCM IDCODE (Length, IR Capture, IR Capture Msk, IDCODE)
jtag_device 4 0x1 0xf 0xe
jtag_device 5 0x1 0x1 0x1e

#target configuration
daemon_startup reset

#target <type> <startup node>
#target cortex_n8 <endianness> <reset node> <chainpos> <variant>
target cortex_n8 little run_and_halt 0
run_and_halt_time 0 30

working_area 0 0x20000000 16384 nobackup

#flash bank <driver> <base> <size> <chip_width> <bus_width>
flash bank stn82x 0x08000000 0x00020000 0 0 0

```

6.7 STM32x Performance Stick

The following script was used with the Hitex STM32 Performance Stick

```

#daemon configuration
telnet_port 4444
gdb_port 3333

#interface
interface ft2232
ft2232_device_desc "STM32-PerformanceStick A"
ft2232_layout stn82stick
jtag_speed 10

jtag_nsrst_delay 100
jtag_ntrst_delay 100

#use combined on interfaces or targets that can't set TRST/SRST separately
reset_config trst_and_srst

#jtag scan chain
#format L IRC IRCM IDCODE (Length, IR Capture, IR Capture Msk, IDCODE)
jtag_device 4 0x1 0xf 0xe
jtag_device 5 0x1 0x1 0x1e
jtag_device 4 0x1 0xf 0xe

```

```

#target configuration
daemon_startup reset

#target <type> <startup node>
#target cortex_n8 <endianness> <reset node> <chainpos> <variant>
target cortex_n8 little run_and_halt 0
run_and_halt_time 0 30

working_area 0 0x20000000 16384 nobackup

#flash bank <driver> <base> <size> <chip_width> <bus_width>
flash bank stn82x 0x08000000 0x00020000 0 0 0

```

6.8 LPC2129 Script

The following script was used with an wiggler PP and a LPC-2129 cpu:

```

#daemon configuration
telnet_port 4444
gdb_port 3333

#interface
interface parport
parport_port 0x378
parport_cable wiggler
jtag_speed 0

#use combined on interfaces or targets that can't set TRST/SRST separately
reset_config trst_and_srst srst_pulls_trst

#jtag scan chain
#format I IRC IRCMIDCODE (Length, IR Capture, IR Capture Msk, IDCODE)
jtag_device 4 0x1 0xf 0xe

#target configuration
daemon_startup reset

#target <type> <startup node>
#target arm7tdmi <endianness> <reset node> <chainpos> <variant>
target arm7tdmi little run_and_halt 0 arm7tdmi-s_r4
run_and_halt_time 0 30

working_area 0 0x40000000 0x4000 nobackup

#flash bank <driver> <base> <size> <chip_width> <bus_width>
flash bank lpc2000 0x0 0x40000 0 0 0 lpc2000_v1 14765 calc_checksum

```

6.9 LPC2148 Script

The following script was used with an Amontec JTAGkey and a LPC2148 cpu:

```

#daemon configuration
telnet_port 4444
gdb_port 3333

#interface
interface ft2232
ft2232_device_desc "Amontec JTAGkey A"
ft2232_layout jtagkey
ft2232_vid_pid 0x0403 0xcff8

```

```

jtag_speed 3

#use combined on interfaces or targets that can't set TRST/SRST separately
reset_config trst_and_srst srst_pulls_trst

#jtag scan chain
#format L IRC IRCM IDCODE (Length, IR Capture, IR Capture Mask, IDCODE)
jtag_device 4 0x1 0xf 0xe

#target configuration
daemon_startup reset

#target <type> <startup node>
#target arm7tdmi <endianness> <reset node> <chainpos> <variant>
target arm7tdmi little run_and_halt 0 arm7tdmi-s_r4
run_and_halt_time 0 30

working_area 0 0x40000000 0x8000 nobackup

#flash configuration
flash bank lpc2000 0x0 0x7d000 0 0 0 lpc2000_v1 14765 calc_checksum

```

6.10 LPC2294 Script

The following script was used with an Amontec JTAGkey and a LPC2294 cpu:

```

#daemon configuration
telnet_port 4444
gdb_port 3333

#interface
interface ft2232
ft2232_device_desc "Amontec JTAGkey A"
ft2232_layout jtagkey
ft2232_vid_pid 0x0403 0xcff8
jtag_speed 3

#use combined on interfaces or targets that can't set TRST/SRST separately
reset_config trst_and_srst srst_pulls_trst

#jtag scan chain
#format L IRC IRCM IDCODE (Length, IR Capture, IR Capture Mask, IDCODE)
jtag_device 4 0x1 0xf 0xe

#target configuration
daemon_startup reset

#target <type> <startup node>
#target arm7tdmi <endianness> <reset node> <chainpos> <variant>
target arm7tdmi little run_and_halt 0 arm7tdmi-s_r4
run_and_halt_time 0 30

working_area 0 0x40000000 0x4000 nobackup

#flash configuration
flash bank lpc2000 0x0 0x40000 0 0 0 lpc2000_v1 14765 calc_checksum

```


6.11 AT91R40008 Script

The following script was used with an Amontec JTAGkey and a AT91R40008 cpu:

```
#daemon configuration
telnet_port 4444
gdb_port 3333

#interface
interface ft2232
ft2232_device_desc "Amontec JTAGkey A"
ft2232_layout jtagkey
ft2232_vid_pid 0x0403 0xcff8
jtag_speed 0
jtag_nsrst_delay 200
jtag_nrst_delay 200

#use combined on interfaces or targets that can't set TRST/SRST separately
reset_config srst_only srst_pulls_trst

#jtag scan chain
#format L IRC IRCM IDCODE (Length, IR Capture, IR Capture Msk, IDCODE)
jtag_device 4 0x1 0xf 0xe

#target configuration
daemon_startup reset

#target <type> <startup mode>
#target arm7tdmi <endianness> <reset mode> <chainpos> <variant>
target arm7tdmi little run_and_halt 0 arm7tdmi
run_and_halt_time 0 30
```

6.12 AT91SAM7s Script

The following script was used with an Olimex ARM-JTAG-OCD and a AT91SAM7S64 cpu:

```
#daemon configuration
telnet_port 4444
gdb_port 3333

#interface
interface ft2232
ft2232_device_desc "Olimex OpenOCD JTAG A"
ft2232_layout olimex-jtag
ft2232_vid_pid 0x15BA 0x0003
jtag_speed 0
jtag_nsrst_delay 200
jtag_nrst_delay 200

#use combined on interfaces or targets that can't set TRST/SRST separately
reset_config srst_only srst_pulls_trst

#jtag scan chain
#format L IRC IRCM IDCODE (Length, IR Capture, IR Capture Msk, IDCODE)
jtag_device 4 0x1 0xf 0xe

#target configuration
daemon_startup reset

#target <type> <startup mode>
```

```

#target arm7tdmi <endianness> <reset node> <chainpos> <variant>
target arm7tdmi little run_and_halt 0 arm7tdmi
run_and_halt_time 0 30

# flash-options AT91
working_area 0 0x00200000 0x4000 nobackup
flash bank at91sam7 0 0 0 0 0

# Information:
# erase command (telnet-interface) for complete flash:
# flash erase <num> 0 numlockbits-1 (can be seen from output of flash info 0)
# SAM564 with 16 lockbits and bank 0: flash erase 0 0 15
# set/clear NVM Bits:
# at91sam7 gpvm <num> <bit> <set|clear>
# disable locking from SAMBA:
# flash protect 0 0 1 off

```

6.13 XSCALE IXP42x Script

The following script was used with an Amontec JTAGkey-Tiny and a xscale ixp42x cpu:

```

#daemon configuration
telnet_port 4444
gdb_port 3333

#interface
interface ft2232
ft2232_device_desc "Amontec JTAGkey A"
ft2232_layout jtagkey
ft2232_vid_pid 0x0403 0xcff8
jtag_speed 0
jtag_nsrst_delay 200
jtag_ntrst_delay 200

#use combined on interfaces or targets that can't set TRST/SRST separately
reset_config srst_only srst_pulls_trst

#jtag scan chain
#format L IRC IRCM IDCODE (Length, IR Capture, IR Capture Mask, IDCODE)
jtag_device 7 0x1 0x7f 0x7e

#target configuration
daemon_startup reset

#target <type> <startup node>
#target arm7tdmi <reset node> <chainpos> <endianness> <variant>
target xscale big run_and_halt 0 IXP42x
run_and_halt_time 0 30

```

6.14 Cirrus Logic EP9301 Script

The following script was used with FT2232 based JTAG interfaces and a Cirrus Logic EP9301 processor on an Olimex CS-E9301 board.

```

#daemon configuration
telnet_port 4444
gdb_port 3333

#interface
interface ft2232

```

```

#Olinex ARM USB- OCD
#ft2232_device_desc "Olinex OpenOCD JTAG"
#ft2232_layout olinex-jtag
#ft2232_vid_pid 0x15ba 0x0003

#Anontec JTAGkey (and JTAGkey-Tiny)
#Serial is only necessary if more than one JTAGkey is connected
ft2232_device_desc "Anontec JTAGkey A"
#ft2232_serial AMJKV31
#ft2232_serial T1P3S2V8
ft2232_layout jtagkey
ft2232_vid_pid 0x0403 0xcff8

#wiggler/parallel port interface
#interface parport
#parport_port 0x378
#parport_cable wiggler
#jtag_speed 0
jtag_speed 1
reset_config trst_and_srst

#jtag scan chain
#format L IRC IRCMIDCODE (Length, IR Capture, IR Capture Msk, IDCODE)
jtag_device 4 0x1 0xf 0xe

jtag_nsrst_delay 100
jtag_nrst_delay 100

#target configuration
daemon_startup attach

#target <type> <endianess> <reset mode>
target arm920t little reset_halt 0
working_area 0 0x80014000 0x1000 backup

#flash configuration
#flash bank <driver> <base> <size> <chip_width> <bus_width> [driver_options ...]
flash bank cfi 0x60000000 0x1000000 2 2 0

```

6.15 Hilscher netX 100 / 500 Script

The following script was used with an Amontec JTAGkey and a Hilscher netX 500 cpu:

```

#daemon configuration
telnet_port 4444
gdb_port 3333

#interface
interface ft2232
ft2232_device_desc "Anontec JTAGkey A"
ft2232_layout jtagkey
ft2232_vid_pid 0x0403 0xcff8
jtag_speed 5

#use combined on interfaces or targets that can't set TRST/SRST separately
reset_config trst_and_srst

#jtag scan chain

```

```

#format L IRC IRCMIDCODE (Length, IR Capture, IR Capture Msk, IDCODE)
jtag_device 4 0x1 0xf 0xe

jtag_nsrst_delay 100
jtag_nrst_delay 100

#target configuration
daemon_startup reset

#target <type> <endianmess> <startup node> <chainpos> <variant>
target arm926ejs little run_and_halt 0 arm926ejs
run_and_halt_time 0 500

```

6.16 Marvell/Intel PXA270 Script

```

# config for Intel PXA270
# not, as of 2007-06-22, openocd only works with the
# libftd2xx library from ftdi. libftdi does not work.

telnet_port 3333
gdb_port 4444

interface ft2232
ft2232_layout olinex-jtag
ft2232_vid_pid 0x15BA 0x0003
ft2232_device_desc "Olinex OpenOCD JTAG"
jtag_speed 0
# set jtag_nsrst_delay to the delay introduced by your reset circuit
# the rest of the needed delays are built into the openocd program
jtag_nsrst_delay 260
# set the jtag_nrst_delay to the delay introduced by a reset circuit
# the rest of the needed delays are built into the openocd program
jtag_nrst_delay 0

#use combined on interfaces or targets that can't set TRST/SRST separately
reset_config trst_and_srst separate

#jtag scan chain
#format L IRC IRCMIDCODE (Length, IR Capture, IR Capture Msk, IDCODE)
jtag_device 7 0x1 0x7f 0x7e

#target configuration
daemon_startup reset

target xscale little reset_halt 0 pxa27x

# maps to PXA internal RAM If you are using a PXA255
# you must initialize SDRAM or leave this option off
working_area 0 0x5c000000 0x10000 nobackup

run_and_halt_time 0 30

#flash bank <driver> <base> <size> <chip_width> <bus_width>
# works for P30 flash
flash bank cfi 0x00000000 0x1000000 2 4 0

```

7 GDB and Openocd

Openocd complies with the remote gdbserver protocol, and as such can be used to debug remote targets.

7.1 Connecting to gdb

A connection is typically started as follows:

```
target remote localhost:3333
```

This would cause gdb to connect to the gdbserver on the local pc using port 3333.

To see a list of available openocd commands type ‘`monitor help`’ on the gdb commandline.

Openocd supports the gdb ‘`qSupported`’ packet, this enables information to be sent by the gdb server (openocd) to gdb. Typical information includes packet size and device memory map.

Previous versions of openocd required the following gdb options to increase the packet size and speed up gdb communication.

```
set remote memory-write-packet-size 1024  
set remote memory-write-packet-size fixed  
set remote memory-read-packet-size 1024  
set remote memory-read-packet-size fixed
```

This is now handled in the ‘`qSupported`’ PacketSize.

7.2 Programming using gdb

By default the target memory map is not sent to gdb, this can be enabled by the following openocd config option:

```
gdb_memory_map enable
```

For this to function correctly a valid flash config must also be configured in openocd. For speed also configure a valid working area.

Informing gdb of the memory map of the target will enable gdb to protect any flash area of the target and use hardware breakpoints by default. This means that the openocd option ‘`arm7_9 force_hw_bkpts`’ is not required when using a memory map.

To view the configured memory map in gdb, use the gdb command ‘`info mem`’ All other unassigned addresses within gdb are treated as ram.

If ‘`gdb_flash_program enable`’ is also used, gdb will be able to program any flash memory using the vFlash interface.

gdb will look at the target memory map when a load command is given, if any areas to be programmed lie within the target flash area the vFlash packets will be used.

Incase the target needs configuring before gdb programming, a script can be executed.

```
target_script 0 gdb_programconfig config.script
```

To verify any flash programming the gdb command ‘`compare-sections`’ can be used.

8 FAQ

1. OpenOCD complains about a missing cygwin1.dll

Make sure you have Cygwin installed, or at least a version of OpenOCD that claims to come with all the necessary dlls. When using Cygwin, try launching the OpenOCD from the Cygwin shell.

2. I'm trying to set a breakpoint using GDB (or a frontend like Insight or Eclipse), but OpenOCD complains that "Info: arm7_9-common.c:213 arm7_9_add_breakpoint(): sw breakpoint requested, but software breakpoints not enabled".

GDB issues software breakpoints when a normal breakpoint is requested, or to implement source-line single-stepping. On ARMv4T systems, like ARM7TDMI, ARM720t or ARM920t, software breakpoints consume one of the two available hardware breakpoints, and are therefor disabled by default. If your code is running from RAM, you can enable software breakpoints with the `'arm7_9 sw_bkpts enable'` command. If your code resides in Flash, you can't use software breakpoints, but you can force OpenOCD to use hardware breakpoints instead: `'arm7_9 force_hw_bkpts enable'`.

3. When erasing or writing LPC2000 on-chip flash, the operation fails sometimes and works sometimes fine.

Make sure the core frequency specified in the `'flash lpc2000'` line matches the clock at the time you're programming the flash. If you've specified the crystal's frequency, make sure the PLL is disabled, if you've specified the full core speed (e.g. 60MHz), make sure the PLL is enabled.

4. When debugging using an Amontec Chameleon in its JTAG Accelerator configuration, I keep getting "Error: amt_jtagaccel.c:184 amt_wait_scan_busy(): amt_jtagaccel timed out while waiting for end of scan, rtck was disabled".

Make sure your PC's parallel port operates in EPP mode. You might have to try several settings in your PC Bios (ECP, EPP, and different versions of those).

5. When debugging with the OpenOCD and GDB (plain GDB, Insight, or Eclipse), I get lots of "Error: arm7_9-common.c:1771 arm7_9_read_memory(): memory read caused data abort".

The errors are non-fatal, and are the result of GDB trying to trace stack frames beyond the last valid frame. It might be possible to prevent this by setting up a proper "initial" stack frame, if you happen to know what exactly has to be done, feel free to add this here.

6. I get the following message in the OpenOCD console (or log file): "Warning: arm7_9-common.c:679 arm7_9_assert_reset(): srst resets test logic, too".

This warning doesn't indicate any serious problem, as long as you don't want to debug your core right out of reset. Your .cfg file specified `'jtag_reset trst_and_srst srst_pulls_trst'` to tell the OpenOCD that either your board, your debugger or your target uC (e.g. LPC2000) can't assert the two reset signals independently. With this setup, it's not possible to halt the core right out of reset, everything else should work fine.

7. When using OpenOCD in conjunction with Amontec JTAGkey and the Yagarto Toolchain (Eclipse, arm-elf-gcc, arm-elf-gdb), the debugging seems to be unstable.

When single-stepping over large blocks of code, GDB and OpenOCD quit with an error message. Is there a stability issue with OpenOCD?

No, this is not a stability issue concerning OpenOCD. Most users have solved this issue by simply using a self-powered USB Hub, which they connect their Amontec JTAGkey to. Apparently, some computers do not provide a USB power supply stable enough for the Amontec JTAGkey to be operated.

8. When using the Amontec JTAGkey, sometimes OpenOCD crashes with the following error messages: "Error: ft2232.c:201 ft2232_read(): FT_Read returned: 4" and "Error: ft2232.c:365 ft2232_send_and_recv(): couldn't read from FT2232". What does that mean and what might be the reason for this?

First of all, the reason might be the USB power supply. Try using a self-powered hub instead of a direct connection to your computer. Secondly, the error code 4 corresponds to an FT_IO_ERROR, which means that the driver for the FTDI USB Chip ran into some sort of error - this points us to a USB problem.

9. When using the Amontec JTAGkey, sometimes OpenOCD crashes with the following error message: "Error: gdb_server.c:101 gdb_get_char(): read: 10054". What does that mean and what might be the reason for this?

Error code 10054 corresponds to WSAECONNRESET, which means that the debugger (GDB) has closed the connection to OpenOCD. This might be a GDB issue.

10. In the configuration file in the section where flash device configurations are described, there is a parameter for specifying the clock frequency for LPC2000 internal flash devices (e.g. `'flash bank lpc2000 0x0 0x40000 0 0 lpc2000_v1 0 14746 calc_checksum'`), which must be specified in kilohertz. However, I do have a quartz crystal of a frequency that contains fractions of kilohertz (e.g. 14,745,600 Hz, i.e. 14,745.600 kHz). Is it possible to specify real numbers for the clock frequency?

No. The clock frequency specified here must be given as an integral number. However, this clock frequency is used by the In-Application-Programming (IAP) routines of the LPC2000 family only, which seems to be very tolerant concerning the given clock frequency, so a slight difference between the specified clock frequency and the actual clock frequency will not cause any trouble.

11. Do I have to keep a specific order for the commands in the configuration file?

Well, yes and no. Commands can be given in arbitrary order, yet the devices listed for the JTAG scan chain must be given in the right order (jtag-device), with the device closest to the TDO-Pin being listed first. In general, whenever objects of the same type exist which require an index number, then these objects must be given in the right order (jtag_devices, targets and flash banks - a target references a jtag-device and a flash bank references a target).

12. Sometimes my debugging session terminates with an error. When I look into the log file, I can see these error messages: Error: arm7_9-common.c:561 arm7_9_execute_sys_speed(): timeout waiting for SYSCOMP

Appendix A GNU Free Documentation License

Version 1.1, March 2000

Copyright (C) 2000, 2003 Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The “Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “you.”

A “Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not “Transparent” is called “Opaque.”

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long

as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the

Title Page. If there is no section entitled “History” in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the “History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. In any section entitled “Acknowledgements” or “Dedications”, preserve the section’s title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section entitled “Endorsements.” Such a section may not be included in the Modified Version.

N. Do not retitle any existing section as “Endorsements” or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents,

unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled “History” in the various original documents, forming one section entitled “History”; likewise combine any sections entitled “Acknowledgements”, and any sections entitled “Dedications.” You must delete all sections entitled “Endorsements.”

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an “aggregate”, and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document’s Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement

between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (C) year your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.1
or any later version published by the Free Software Foundation;
with the Invariant Sections being *list their titles*, with the
Front-Cover Texts being *list*, and with the Back-Cover Texts being *list*.
A copy of the license is included in the section entitled "GNU
Free Documentation License."

If you have no Invariant Sections, write “with no Invariant Sections” instead of saying which ones are invariant. If you have no Front-Cover Texts, write “no Front-Cover Texts” instead of “Front-Cover Texts being *list*”; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Index

A

about	2
Architecture Specific Commands	19
ARM7/9 specific commands	19
arm7_9 dbgrq	19
arm7_9 dcc_downloads	19
arm7_9 fast_memory_access	19
arm7_9 fast_writes [DEPRECATED]	19
arm7_9 force_hw_bkpts	19
arm7_9 sw_bkpts	19
arm7_9 write_core_reg	20
arm7_9 write_xpsr	20
arm7_9 write_xpsr_im8	20
arm720t options	12
arm7tdmi options	12
arm920t cache_info	19
arm920t md<bhw>_phys	19
arm920t mw<bhw>_phys	19
arm920t options	12
arm920t read_cache	19
arm920t read_mmu	19
ARM920T specific commands	19
arm920t virt2phys	19
arm966e options	12
arm9tdmi options	12
ARMV4/5 specific commands	19
armv4_5 core_mode	19
armv4_5 reg	19
AT91R40008 Script	29
at91sam7 gpnvm	17
at91sam7 options	13
AT91SAM7 specific commands	17
AT91SAM7s Script	29

B

building openocd	4
------------------------	---

C

cfi options	12
chameleon	9
Cirrus Logic EP9301 Script	30
commands	14
configuration	7
Connecting to gdb	33

D

daemon_startup	7
Debug commands	20
debug_level	14
developers	3
dlc5	9

drscan	21
dump_binary	15
dump_image	15

E

endstate	21
ep93xx options	10

F

faq	34
field	21
flash auto_erase	16
flash autoerase	12
flash bank	12
flash banks	15
Flash commands	15
Flash configuration	12
flash erase	15
flash erase_address	15
flash erase_check	15
flash erase_sector	15
flash info	15
flash probe	15
flash protect	16
flash protect_check	15
flash write	15
flash write_binary	15
flash write_image	16
flashlink	9
ft2232_device_desc	9
ft2232_layout	9

G

GDB and Openocd	33
gdb_detach	7
gdb_flash_program	7
gdb_memory_map	7
gdb_port	7

H

halt	14
Hilscher netX 100 / 500 Script	31

I

interface	7
irscan	21

J

JTAG commands	21
jtag_device	8
jtag_nsrst_delay	8
jtag_ntrst_delay	8
jtag_reset	21
jtag_speed	7

L

load_binary	15
load_image	15
log_output	14
lpc2000 options	12
LPC2129 Script	27
LPC2148 Script	27
LPC2294 Script	28

M

Marvell/Intel PXA270 Script	32
mdb	15
mdh	14
mdw	14
mwb	15
mwh	15
mww	15

O

old_amt_wiggler	9
OMAP5912 Flash Debug	22

P

parport_cable	8
parport_port	8, 9
parport_write_on_exit	9
poll	14
Programming using gdb	33

R

reset	14
reset halt	14
reset init	14
reset run	14
reset run_and_halt	14
reset run_and_init	14
reset_config	8
reset_halt	11
reset_init	11
reset_run	11
resume	14
run_and_halt	11
run_and_halt_time	11
run_and_init	11

running openocd	6
runtest	21

S

scan_chain	21
script	14
scripts	22
shutdown	14
sleep	14
statemove	21
stellaris (LM3Sxxx) options	13
step	14
stm32x lock	17
stm32x mass_erase	18
stm32x options	13
stm32x options_read	18
stm32x options_write	18
STM32x Performance Stick Script	26
STM32x Script	25
STM32x specific commands	17
stm32x unlock	17
str7 options	13
STR71x Script	23
STR750 Script	24
STR9 configuration	17
STR9 option byte configuration	17
str9 options	13
STR9 specific commands	17
STR912 comstick Script	25
STR912 Script	24
str9x flash_config	17
str9xpec disable_turbo	17
str9xpec enable_turbo	17
str9xpec lock	17
str9xpec options_cmap	17
str9xpec options_lvdsel	17
str9xpec options_lvdthd	17
str9xpec options_lvdwarn	17
str9xpec options_read	17
str9xpec options_write	17
str9xpec unlock	17

T

target	11
Target Specific Commands	17
target_script	11
telnet_port	7
triton	9

V

var	21
verify_image	15
verify_ircapture	21

W

wiggler 8

working_area 11

X

XSCALE IXP42x Script..... 30

xscale options 12