Using OpenOCD as a Standalone FLASH Programmer

(a simple tutorial)

Motivation

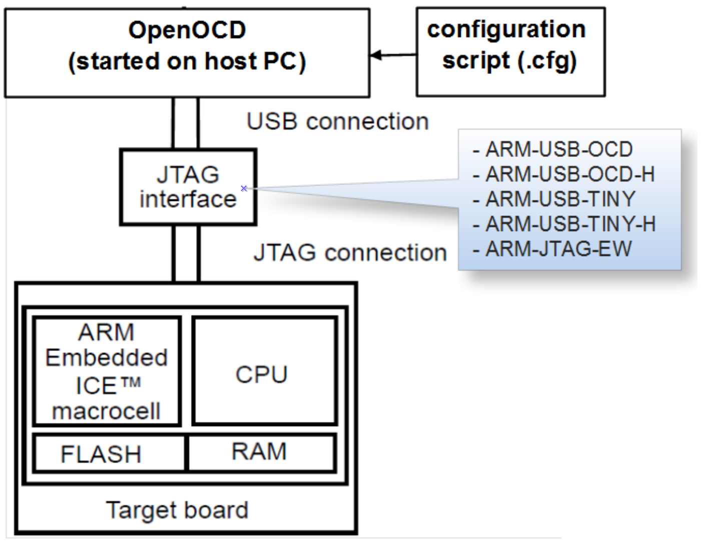
Using OpenOCD as just a programmer instead of a debug tool is very convenient in cases of mass production where you already have a prebuilt and already debugged image and you only need to download that image to the target device. In this mode of operation programming becomes as easy as starting the OpenOCD executable – all the rest is automatic.

Installing Device Drivers

Please follow the instructions in **README.txt** file in the drivers folder included in the package. There is nothing specific about the installation process if you have installed Windows drivers before.

Connection between OpenOCD and the Target Board

As can be seen in the picture below OpenOCD is an executable running on a host PC. OpenOCD communicates with a hardware JTAG adapter in the face of ARM-USB-OCD (and the rest of the series of products by Olimex). The JTAG adapter, in turn, communicates with the JTAG module in the target device. In order to know what to do, OpenOCD needs a configuration script, which contains information about the connected target device (type of processor, available memory, etc.) as well as commands to execute after initialization (in this case to program the device).



Connection between Target and OpenOCD

OpenOCD Configuration Files

As mentioned above, OpenOCD needs a configuration file upon starting. A typical solution is to override the search for the configuration file and provide one or more configuration files with different names using the **–f** switch. For your convenience configuration files are divided into 2 types, one to configure the JTAG interface being used and another to configure the target processor. Interface configuration files for the supported JTAG adapters are placed inside the directory of OpenOCD executable. Sample target configuration scripts are located in the 'OpenOCD\tcl\target' directory.

To implement the programming function of OpenOCD you need to add some lines to the configuration file of the target. These include commands to program the target and commands to start the programming itself.

After locating the correct configuration script for your target you need to create a procedure inside in the file to be called to program the device. This procedure will contain the operations for programming. If, for example, it is called "program_device" the syntax will go like this:

```
proc program_device () {
      <commands>
```

```
              …
        }
```

Note: Please follow the exact syntax, i.e. the needed space between the name of the procedure and the empty brackets.

Programming the flash memory of a device typically requires a sequence of:

1) Erasing the device memory
2) Optional: unlocking the flash pages of the device
3) Writing to the memory of the device

Luckily enough, all these operations are contained in the "**flash write_image**", so provided with the correct parameters it may be the only command inside the body of the procedure. Please consult OpenOCD User's Guide to see how this command works. However, additional commands may be needed or desired to accompany the "**flash write_image**". Here is a sample procedure for programming AT91SAM7S256.

```
proc program_device () {
      # halt the processor
      halt
      wait_halt

      # write file to flash memory
      arm7_9 dcc_downloads enable
      sleep 10
      poll
      flash probe 0
      flash write_image erase unlock "AT91SAM7_H256.BIN" 0x00100000
      sleep 10

      #start execution of the program just downladed
      reset run
      sleep 10

      #exit OpenOCD
      shutdown
}
```

In this case "**flash write_image**" is used to its full potential to erase and unlock flash memory before writing the image. Note that you should also provide the image file to program as well as the start address of the flash memory in the device. The example uses a binary file, however there are other formats supported. Besides, the input image file is located in the directory of OpenOCD, otherwise a relative of absolute path will be required. The target device is reset and run after programming. At the end of this procedure OpenOCD shuts down.

Note: The procedure you create should be put somewhere in the configuration file, there is no exact place for it.

The next thing to do is to put commands to actually start the programming process. Locate the end of the configuration file and add these lines (the place of the commands here is actually important!):

```
init
reset init
program_device ()
```

Please again note the space between the procedure name and the brackets.

Save the file and continue.

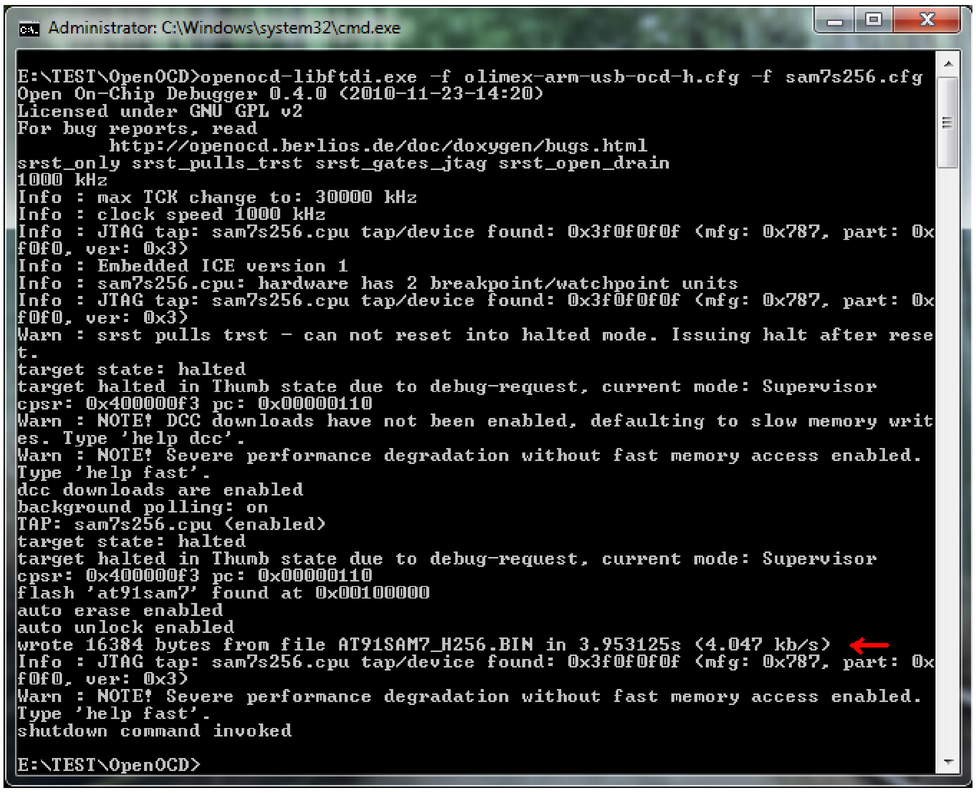
General note: It is actually not a bad idea to initially test basic connectivity with the device prior to editing the configuration file.

Starting OpenOCD

The OpenOCD executable file is called: **openocd-libftdi.exe**. It is ready to run and needs no installation!

If you do not provide any file with the –f switch the default file to search for is **'openocd.cnf'** so do not get upset if you just start '**openocd-libftdi.exe**' and end up with an error message. If you are using ARM-OCD-H and an AT91SAM7S256 processor, for example, type:

```
> openocd-libftdi.exe -f olimex-arm-usb-ocd-h.cfg -f sam7s256.cfg
```



```
Administrator: C:\Windows\system32\cmd.exe

E:\TEST\OpenOCD>openocd-libftdi.exe -f olimex-arm-usb-ocd-h.cfg -f sam7s256.cfg
Open On-Chip Debugger 0.4.0 (2010-11-23-14:20)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.berlios.de/doc/doxygen/bugs.html
srst_only srst_pulls_trst srst_gates_jtag srst_open_drain
1000 kHz
Info : max TCK change to: 30000 kHz
Info : clock speed 1000 kHz
Info : JTAG tap: sam7s256.cpu tap/device found: 0x3f0f0f0f (mfg: 0x787, part: 0x
f0f0, ver: 0x3)
Info : Embedded ICE version 1
Info : sam7s256.cpu: hardware has 2 breakpoint/watchpoint units
Info : JTAG tap: sam7s256.cpu tap/device found: 0x3f0f0f0f (mfg: 0x787, part: 0x
f0f0, ver: 0x3)
Warn : srst pulls trst - can not reset into halted mode. Issuing halt after rese
t.
target state: halted
target halted in Thumb state due to debug-request, current mode: Supervisor
cpsr: 0x400000f3 pc: 0x00000110
Warn : NOTE! DCC downloads have not been enabled, defaulting to slow memory writ
es. Type 'help dcc'.
Warn : NOTE! Severe performance degradation without fast memory access enabled.
Type 'help fast'.
dcc downloads are enabled
background polling: on
TAP: sam7s256.cpu (enabled)
target state: halted
target halted in Thumb state due to debug-request, current mode: Supervisor
cpsr: 0x400000f3 pc: 0x00000110
flash 'at91sam7' found at 0x00100000
auto erase enabled
auto unlock enabled
wrote 16384 bytes from file AT91SAM7_H256.BIN in 3.953125s (4.047 kb/s)
Info : JTAG tap: sam7s256.cpu tap/device found: 0x3f0f0f0f (mfg: 0x787, part: 0x
f0f0, ver: 0x3)
Warn : NOTE! Severe performance degradation without fast memory access enabled.
Type 'help fast'.
shutdown command invoked

E:\TEST\OpenOCD>
```

Screenshot with OpenOCD executed.

This command line assumes that both configuration files and the image file reside in the same folder as the OpenOCD executable. Upon starting OpenOCD connects to the JTAG module of the target device and reset is executed. Then the programming starts (successful execution is marked with an arrow) and after that the target is run.

Note: OpenOCD is quite verbose by nature. Although most of the warnings it provides are meaningful and worth clearing, the process will finish just fine in most cases in spite of the warnings.

Conclusion

Please note that OpenOCD is sometimes pretty tricky to work with so be aware of configuration and runtime issues. Well, if things get very ugly you should stop the server, reconnect the JTAG adapter and start over.

If in need of more information check out any of the following:

– http://openocd.berlios.de/web/ - the official web page of the OpenOCD project.
– http://forum.sparkfun.com/viewforum.php?f=18 – OpenOCD discussion board (very helpful indeed).
– http://www.olimex/dev/ - manufacturer of development boards and tools. That's us :).

Revision history

– 31 January 2011 – initial release.

THE END