Using OpenOCD JTAG for DICE Firmware Development

Doc Rev 0.9

12/11/2009

File:	DICE_OOCD_JTAG.doc
Printed:	11 December 2009
Updated:	11 December 2009
Version:	0.9

Document History

Version	**Date**	**Name**	**Comment**
0.4	February 20, 2008	B Karr	Document created
0.5	April 04, 2008	B Karr	Misc. edits.
0.6	June 20, 2008	B Karr	Section 2 “Using .gdbinit for gdb” Changed description of where to put the .gdbinit file Added instructions for enabling sw breakpoints
0.7	October 07, 2009	B Karr	Added discussion of the new (0.2.0) OpenOCD configuration file syntax
0.8	December 08, 2009	B Karr	doc rework
0.9	December 11, 2009	B Karr	Finished section 5.

CONTENTS

1. What's new

OpenOCD versions 0.3.1 and higher are supported. You may have good success with 0.2.0 and later, however the later versions have improvements that warrant updating.

This document has been reworked to describe use of OpenOCD for debugging with GDB and Insight, and for directly programming flash.

Due to licensing issues, OpenOCD binaries that support proprietary ftd2xx USB drivers cannot be distributed. For this reason, we only support OpenOCD binaries that support the libftdi (libusb-win32). Proprietary drivers may still be used, but the user must build OpenOCD from sources to use them. See below for details.

2. Introduction

This document describes how to use OpenOCD for DICE-based hardware bring-up and firmware development.

The Open On-Chip Debugger was developed by Dominic Rath. More information about OpenOCD can be found at http://openocd.berlios.de/web/

OpenOCD is an open source host software solution for accessing embedded ARM cores, among other uses, via a JTAG interface. It provides an interface between the gdb debugger and JTAG hardware interfaces, and has a number of built-in services which can be used to analyze a JTAG scan-chain and program various CPLD, FPGA and FLASH devices.

OpenOCD is compatible with a number JTAG interfaces which can be used with the DICE family of chips. This document no longer covers OpenOCD versions less than 0.2.0. The newer versions have a new configuration file syntax and a number of improvements that make older versions obsolete.

We acknowledge Uli Franke at Weiss Engineering Ltd. for contributing the initial setup procedures for using OpenOCD 0.1.x with the DICE chip family.

Thanks also to Joachim Stahlhacke for sharing info for using high-speed USB interfaces such as the JTAGkey2.

Last but not least, the Windows installer for OpenOCD that is included in the Firmware Development Environment is provided by Freddie Chopin. The latest installer can be found at www.freddiechopin.info

Any errors in this document are the author's, and not by the contributors mentioned above.

3. Typical Development Chain

Working from the target up, the OpenOCD JTAG development system consists of the target development hardware, the JTAG interface, a serial or parallel connection to the host workstation, a driver for the serial or parallel connection, an OpenOCD JTAG server and the gdb debugger, utility or script file.

Since the DICE firmware development is a Windows Environment, this document will focus only on the Windows OpenOCD environment.

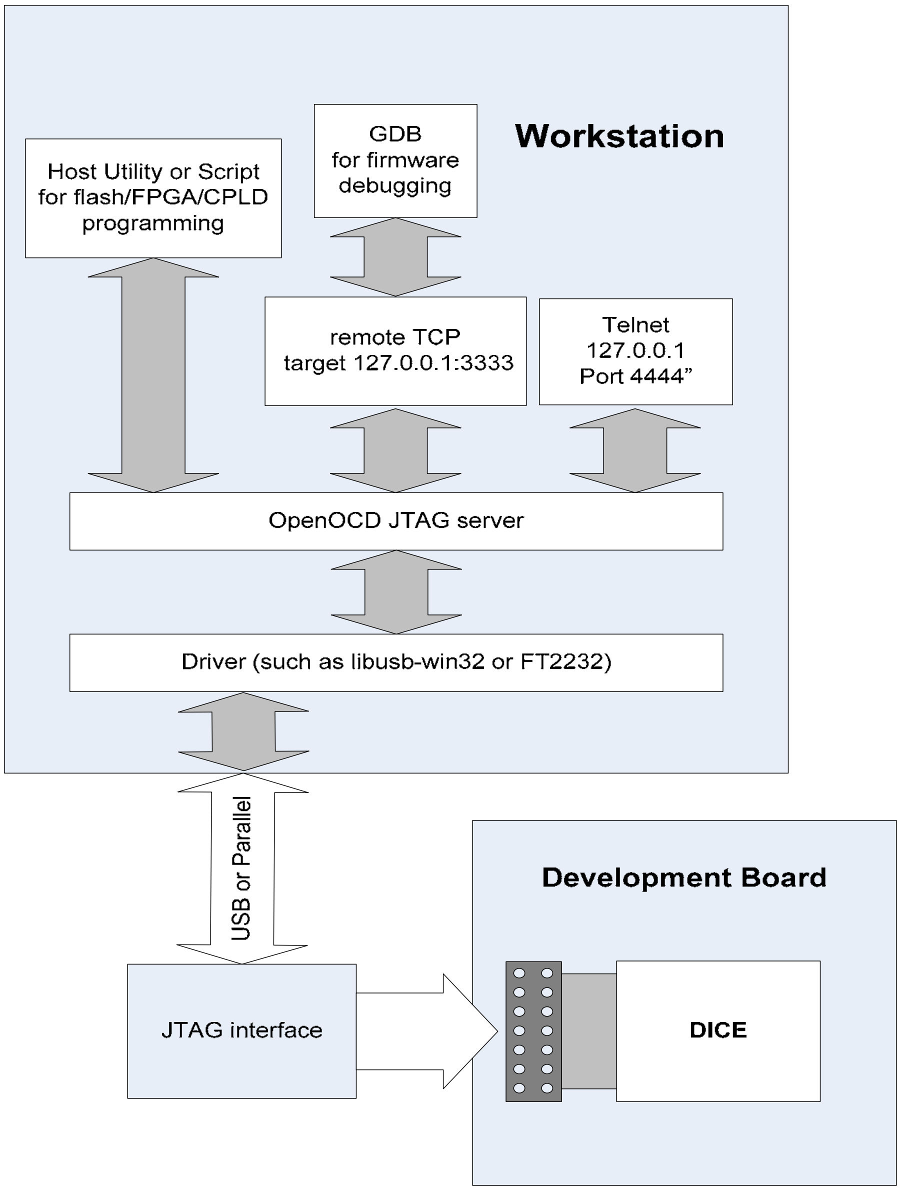

Workstation
Host Utility or Script for flash/FPGA/CPLD programming
GDB for firmware debugging
remote TCP target 127.0.0.1:3333
Telnet 127.0.0.1 Port 4444"
OpenOCD JTAG server
Driver (such as libusb-win32 or FT2232)
USB or Parallel
JTAG interface
Development Board
DICE

Figure 1: Typical development system

4. Quick Start

Obtain a JTAG dongle, and 20-pin to 14-pin adapter, that works with OpenOCD

See below for dongles we have used successfully. Most any OpenOCD-compatible JTAG interface should work well. The latest compatibility list is usually updated here: http://openocd.berlios.de/doc/pdf/openocd.pdf

Install the LibUSB version of OpenOCD, which will have the correct compatible driver

This can be found in the DICE Firmware Development Environment. Also, with each new release the open source community usually follows with an installer for it. Freddie Chopin maintains an installer at www.freddiechopin.info

Do not install any drivers or software that are bundled with the dongle, or are provided on the manufacturer's website, since they are usually the proprietary versions. The proper drivers will be installed by the LibUSB-based OpenOCD.

When the proprietary drivers are installed, the device will appear in the Device Manager in the 'Universal Serial Bus controllers' area. If you have installed the proprietary drivers, you must remove them using the FTClean utility from http://www.ftdichip.com/Resources/Utilities.htm

When properly installed with the non-proprietary drivers, the device will appear in the 'LibUSB-Win32 Devices' area.

Modify the batch files and configuration files for your situation

The configuration includes three main areas, the interface (JTAG dongle model), the target CPU, and the board-specific configurations. The batch files you will use invoke the OpenOCD server with an interface configuration file parameter and a DICE configuration file parameter. You can create the appropriate files by copying them out of this document, and they will also be included in your Subversion tag in the **/docs/public/firmware** directory.

Start a Debug Session or Program Flash

If you want to debug firmware, make sure all OpenOCD windows are closed, reset the board, run the **debug.bat** batch file, and then connect with gdb (or your favorite GUI for gdb such as Insight). You can also connect (at the same time as gdb if you wish) with telnet and interact with the server and target that way as well. See the OpenOCD documentation for available commands.

To program flash, make sure all OpenOCD windows are closed, then run **burn.bat**.

5. Batch Files and Configuration Files

Debugging with OpenOCD

Typically, you will only have to modify the batch file to specify the model of your JTAG interface.

You may have to modify the configuration files if your board is significantly different than the DICE EVM's. If your memories have different chip selects and sizes, then make the necessary changes in **dice_oocd_init.cfg**.

OpenOCD includes a configuration files for most available JTAG interfaces, so a typical batch file to run the OpenOCD server for debugging firmware looks like:

Top level batch file **debug.bat**:

```
@echo off
rem debug.bat – run openocd server specifying JTAG interface and DICE configuration

“C:\Program Files\OpenOCD\0.3.1\bin\openocd.exe” ^
-f “C:\Program Files\OpenOCD\0.3.1\interface\jtagkey.cfg” ^
-f dice_oocd_debug.cfg

pause
```

Where you may need to modify the version directory depending on which OpenOCD you are using. You will need version 0.3.1 or later. Of course, if the dice configuration file is not in the same directory as the batch file, then specify the full path to it. To use a different JTAG interface, just specify a different configuration file.

Configuring the DICE target and running the server

Below is an example configuration file used for debugging. When invoked in a batch file with a pause command, the OpenOCD server continues running so you can attach GDB or interact with the target using a telnet session.

Note that a **.gdbinit** file is not necessary when using these scripts. If you have a .gbinit file in the same directory as the binary, or in your cygwin home directory, make sure it does not have any commands in it that conflict with these files.

Target configuration file **dice_oocd_debug.cfg**:

```
# dice_oocd_debug.cfg
# invoke on the openocd.exe command-line after the interface configuration file

# configures the Dice CPU target for debugging


# specify the ports to use for gdb and telnet sessions
gdb_port 3333
telnet_port 4444

# set up variables
if { [info exists CHIPNAME] } {
   set  _CHIPNAME $CHIPNAME
} else {
   set  _CHIPNAME tcat-dice
}

if { [info exists ENDIAN] } {
   set  _ENDIAN $ENDIAN
} else {
   set  _ENDIAN little
}

if { [info exists CPUTAPID ] } {
   set _CPUTAPID $CPUTAPID
} else {
  # force an error till we get a good number
  set _CPUTAPID 0x1f0f0f0f

}

# set up maximum JTAG clock rate.
# if programming doesnt work (e.g. target doesnt stay in RESET) change JTAG speed
# optimal speed for high-speed interfaces, such as JTAGkey2, is 15000
#  other devices will default back to 6000
jtag_khz 15000

# use combined on interfaces or targets that can't set TRST/SRST separately
reset_config trst_and_srst srst_pulls_trst
jtag_nsrst_delay      250
jtag_ntrst_delay      250

# jtag scan chain
jtag newtap $_CHIPNAME cpu -irlen 4 -ircapture 0x1 -irmask 0xf -expected-id $_CPUTAPID

#target configuration
set _TARGETNAME [format "%s.cpu" $_CHIPNAME]
target create $_TARGETNAME arm7tdmi -endian $_ENDIAN -chain-position $_TARGETNAME -
variant arm7tdmi-s_r4

# set up the reset-init event handler
$_TARGETNAME configure -event reset-init { source dice_oocd_init.cfg }

# working area for fast transfers
# if your image uses over 2MB RAM, increase the physical address below
$_TARGETNAME configure -work-area-virt 0 -work-area-phys 0x00200000 -work-area-size
0x4000 -work-area-backup 0

# speed-ups, use if working area is defined
arm7_9 fast_memory_access enable
arm7_9 dcc_downloads enable

# enter normal command mode, allowing memory read/write comands
init

# set up DICE chip selects, memories, etc.
source dice_oocd_init.cfg

# program resids in RAM, enable software breakpoints
gdb_breakpoint_override soft

echo " "
echo "ready to attach gdb/telnet"
echo " "
```

The script above also loads **dice_oocd_init.cfg** which is listed below. This file sets up the DICE chip selects and memories.

```
# dice_oocd_init.cfg
# initialize DICE interrupts and memories
#   this replaces .gdbinit when debugging with gdb

reset halt

# disable watchdog
mww 0xbf000000 0x00001234
mww 0xbf000004 0x00005678

#########################################################
# setup int mux
mww 0xc7000024 0x0c83a443
mww 0xc7000028 0x14488025
mww 0xc700002c 0x20f6b98b
mww 0xc7000030 0x12
mww 0xc7000034 0x0
mww 0xc7000038 0x0

##########################################################
#set the "set0-set2" manager
#set0 - SRAM8
mww 0x81000094 0x10000041
#set1 - SRAM16
mww 0x81000098 0x10000041
#set2 - FLASH -
mww 0x8100009c 0x10011405
##########################################################
#set the static memory ctrl register
#SMCTRLR (set0: 8bit, set1: 16bit, set2: 16bit)
mww 0x810000a4 0x0000020F

##########################################################
#Static Memory Mask Register

#SMSKR0FLASH  -   2MB- set2
mww 0x81000054 0x00000246
#SMSKR1 SDRAM  -   8MB  -
mww 0x81000058 0x00000008
#SMSKR2SRAM8  - 512KB- set1
mww 0x8100005c 0x00000024
#SMSKR3SRAM16 - 512KB- set2
mww 0x81000060 0x00000124
#SMSKR4 DSP    - Disable
mww 0x81000064 0x00000020
#SMSKR5 FPGA   - Disable
mww 0x81000068 0x00000020
#SMSKR6 FPGA   - Disable
mww 0x8100006c 0x00000020
#SMSKR7 FPGA   - Disable
mww 0x81000070 0x00000020

##########################################################
#SDRAM Configuration & Initialization
mww 0x81000000 0x00140f68
mww 0x81000004 0x00dcd449
mww 0x81000008 0x00012000
#50MHz clock
mww 0x81000010 0x0000030c
# 2MHz clock
#mww 0x81000010 0x0000001F
mww 0x81000058 0x00000008
#mww 0x8100000c 0x00002071

##########################################################
#route CS in GPCSR Registers
#GPCSR_IO CS2,CS3,CS4,CS5,CS6,CS7
mww 0xc7000004 0x000502AA
#GPCSR_IO CS2,CS3,CS4
mww 0xc7000004 0x00050280

##########################################################
#set all the base addresses,
#FLASH
mww 0x81000014 0x04000000
#SDRAM
mww 0x81000018 0x00000000
#SRAM8
mww 0x8100001c 0x02080000
#SRAM16
mww 0x81000020 0x02000000
#DSP
mww 0x81000024 0x05000000
#FPGA
mww 0x81000028 0x06000000
#FPGA
mww 0x8100002c 0x07000000
#FPGA
#mww 0x81000030 0x08000000

##########################################################
#set the alias and remap registers to be the same as base
```

```
#CSALIAS0
mww 0x81000074 0x04000000
#CSALIAS1
mww 0x81000078 0x00000000
#CSREMAP0
mww 0x81000084 0x04000000
#CSREMAP1
mww 0x81000088 0x01000000
```

When successfully running, the output in the OpenOCD server window will be as follows:

```
Open On-Chip Debugger 0.3.1 (2009-11-13-16:13)
$URL$
For bug reports, read
        http://openocd.berlios.de/doc/doxygen/bugs.html
debug_level: 2
15000 kHz
trst_and_srst srst_pulls_trst srst_gates_jtag trst_push_pull srst_open_drain
jtag_nsrst_delay: 250
jtag_ntrst_delay: 250
fast memory access is enabled
dcc downloads are enabled
Info : clock speed 6000 kHz
Info : JTAG tap: tcat-dice.cpu tap/device found: 0x1f0f0f0f (mfg: 0x787, part: 0xf0f0,
ver: 0x1)
Info : Embedded ICE version 1
target state: halted
target halted in ARM state due to breakpoint, current mode: Supervisor
cpsr: 0x00000013 pc: 0x00038b34
Info : JTAG tap: tcat-dice.cpu tap/device found: 0x1f0f0f0f (mfg: 0x787, part: 0xf0f0,
ver: 0x1)
Warn : srst pulls trst - can not reset into halted mode. Issuing halt after reset.
target state: halted
target halted in ARM state due to breakpoint, current mode: Supervisor
cpsr: 0x00000013 pc: 0x00038b60
force soft breakpoints

ready to attach gdb/telnet
```

Connect with GDB

1) If you already have firmware installed on the board, make sure autorun is not active.

Boot the board ad enter ctrl+c in the CLI to get a RedBoot prompt. Then run fconfig.

```
Watchdog Int installed!!!

RedBoot(tm) bootstrap and debug environment [ROMRAM]
Non-certified release, version v2_0 - built 08:44:32, Dec 18 2007

Platform: TCAT DICE/VB (ARM7TDMI)
Copyright (C) 2000, 2001, 2002, Red Hat, Inc.

RAM: 0x00000000-0x00800000, 0x00016528-0x007ed000 available
== Executing boot script in 20.000 seconds - enter ^C to abort
^C
RedBoot>
RedBoot> fco
Run script at boot: false
Update RedBoot non-volatile configuration - continue (y/n)? y
... Erase from 0x041e0000-0x041e1000: .
... Program from 0x007ee000-0x007ef000 at 0x041e0000: .
RedBoot>
```

2) A **.gdbinit** file is not necessary. It is not recommended at all if you are using a gdb GUI such as Insight. If you have a .gdbinit file in the same directory where you run gdb, or if you have on e in your home directory (i.e. c:\cygwin\home\joe) then

make sure it doesn't have any commands which will interfere with the OpenOCD configuration files.

A typical .gdbinit would look like the one below. This automates the attaching and loading of the image.

```
target remote 127.0.0.1:3333
load
c
```

3) In a bash window, go to your project's 'bin' directory and run gdb as follows
$ arm-elf-gdb.exe <debugImageName>

4) Attach, load and continue. If you are not using .gdbinit as above, enter the commands manually

```
(gdb) target remote 127.0.0.1:3333
(gdb) load
(gdb) c
```

You will then see a splash screen in the CLI.

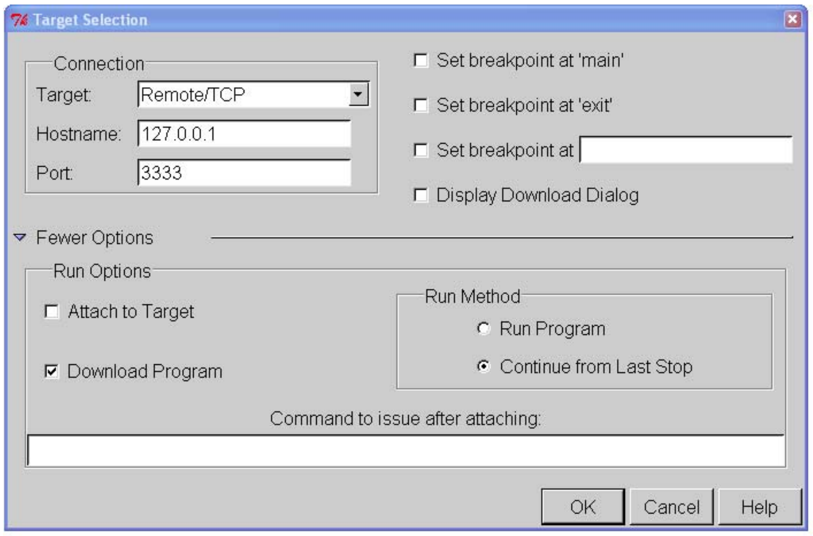
Connect with Insight

1) If you already have firmware installed on the board, make sure autorun is not active, as described above.

2) Make sure there is no .gdbinit file in the directory where you run Insight, and also no .gdbinit in your home directory, as described above. If you do have a .gdbinit, make sure it has no commands that will interfere with the OpenOCD configurations.

3) Open the debug image

4) Set up the Target Settings for the image (first time only)


Target Selection

Connection
Target: Remote/TCP
Hostname: 127.0.0.1
Port: 3333

☐ Set breakpoint at 'main'
☐ Set breakpoint at 'exit'
☐ Set breakpoint at
☐ Display Download Dialog

Run Options
☐ Attach to Target
☑ Download Program

Run Method
○ Run Program
● Continue from Last Stop

Command to issue after attaching:

4) Click on the Run button to begin.

Programming flash

Typically, you will only have to modify the batch file below to specify the model of your JTAG interface.

You may have to modify the configuration files if your board is significantly different than the DICE EVM's. If your flash part is not CFI, then modify the 'flash bank' command in **dice_oocd_flash.cfg**. If your memories have different chip selects and sizes, then make the necessary changes in **dice_oocd_init.cfg**.

In the case where you want to program an entire flash, you can use OpenOCD with the **dice_oocd_flash.cfg** script. This script also uses **dice_oocd_init.cfg**.

Make sure that OpenOCD is not already running before running this script.

This assumes that you have a full flash image called **completeFlashImage.bin** in the same directory as the batch file.

To get started, a pre-made image is included in your subversion tag in the **/docs/public/firmware/OpenOCD/flashImages** directory. You can extract a full image yourself using the TCAT Control Panel (with Firmware Utilities tab enabled) or with the 'dice' tool as follows

```
dice dfd completeFlashImage.bin
```

Files

Top level batch file **burn.bat**:

```
@echo off
rem burn.bat – program EVM flash specifying JTAG interface and DICE configuration

“C:\Program Files\OpenOCD\0.3.1\bin\openocd.exe” ^
–f “C:\Program Files\OpenOCD\0.3.1\interface\jtagkey.cfg” ^
–f dice_oocd_flash.cfg

pause
```

Target Configuration File **dice_oocd_flash.cfg**:

```
# dice_oocd_flash.cfg
# configures the Dice CPU target for writing flash
#  invoke on the openocd.exe command-line after the interface configuration file
#  assumes the flash file completeFlashImage.bin is in the same directory

# specify the ports to use for gdb and telnet sessions
gdb_port 3333
telnet_port 4444

# set up variables
if { [info exists CHIPNAME] } {
   set  _CHIPNAME $CHIPNAME
} else {
   set  _CHIPNAME tcat-dice
}

if { [info exists ENDIAN] } {
   set  _ENDIAN $ENDIAN
} else {
   set  _ENDIAN little
}
```

```
if { [info exists CPUTAPID ] } {
   set _CPUTAPID $CPUTAPID
} else {
  # force an error till we get a good number
  set _CPUTAPID 0x1f0f0f0f

}

# set up maximum JTAG clock rate.
# if programming doesnt work (e.g. target doesnt stay in RESET) change JTAG speed
# optimal speed for high-speed interfaces, such as JTAGkey2
jtag_khz 15000

# use combined on interfaces or targets that can't set TRST/SRST separately
reset_config trst_and_srst srst_pulls_trst
jtag_nsrst_delay      250
jtag_nsrst_delay      250

#jtag scan chain
jtag newtap $_CHIPNAME cpu -irlen 4 -ircapture 0x1 -irmask 0xf -expected-id $_CPUTAPID

#target configuration
set _TARGETNAME [format "%s.cpu" $_CHIPNAME]
target create $_TARGETNAME arm7tdmi -endian $_ENDIAN -chain-position $_TARGETNAME -
variant arm7tdmi-s_r4

# working area for fast transfers
# this uses internal SRAM, if remapped properly after 'init' below
$_TARGETNAME configure -work-area-virt 0 -work-area-phys 0 -work-area-size 0x4000 -work-
area-backup 0

# speed-ups
arm7_9 dcc_downloads enable
arm7_9 fast_memory_access enable

# flash memories on DICE EVM's are cfi type
# flash bank <driver> <base> <size> <chip_width> <bus_width>
flash bank cfi 0x04000000 0x200000 2 2 $_TARGETNAME

# enter normal command mode, allowing memory read/write comands
init

# set up DICE chip selects, memories, etc.
source dice_oocd_init.cfg

# Remap address zero to the on-chip RAM, which will be used for the flash programming
routine and its buffers
mww 0xc0000008 0x00000001

# probe for the flash and write the file
flash probe 0
echo " "
echo "** programming flash"
echo "** working area warnings are normal"
flash write_image erase completeFlashImage.bin 0x04000000 bin

echo " "
echo "** done. reset the board to run from flash."
echo " "

# stop OpenOCD
shutdown
```

When the flash is successfully programed, the output will be as listed below. The working area warnings are normal. We specify 0x4000 bytes of working area (the amount in the DICE internal SRAM), and OpenOCD tries to allocate 0x8000 bytes and reduces the allocation amount until it matches the amount specified in the script. When the process is finished, reset the board to run from flash.

```
Open On-Chip Debugger 0.3.1 (2009-11-13-16:13)
$URL$
For bug reports, read
        http://openocd.berlios.de/doc/doxygen/bugs.html
15000 kHz
```

```
trst_and_srst srst_pulls_trst srst_gates_jtag trst_push_pull srst_open_drain
jtag_nsrst_delay: 250
jtag_nsrst_delay: 250
dcc downloads are enabled
fast memory access is enabled
Info : clock speed 6000 kHz
Info : JTAG tap: tcat-dice.cpu tap/device found: 0x1f0f0f0f (mfg: 0x787, part: 0xf0f0,
ver: 0x1)
Info : Embedded ICE version 1
target state: halted
target halted in ARM state due to breakpoint, current mode: Supervisor
cpsr: 0x00000013 pc: 0x0003950c
Info : JTAG tap: tcat-dice.cpu tap/device found: 0x1f0f0f0f (mfg: 0x787, part: 0xf0f0,
ver: 0x1)
Warn : srst pulls trst - can not reset into halted mode. Issuing halt after reset.
target state: halted
target halted in ARM state due to breakpoint, current mode: Supervisor
cpsr: 0x00000013 pc: 0x00039538
Info : Flash Manufacturer/Device: 0x0020 0x2249
flash 'cfi' found at 0x04000000

** programming flash
** working area warnings are normal
auto erase enabled
Warn : not enough working area available(requested 32768, free 16288)
Warn : not enough working area available(requested 16384, free 16288)
wrote 2097152 byte from file completeFlashImage.bin in 59.984375s (34.142223 kb/s)

** done. reset the board to run from flash.

Press any key to continue . . .
```

The 15000 kHz indicates that a high-speed JTAG interface (JTAGkey2) is used. If a lower speed interface is used, then this will automatically throttle down to 6000 KHz, and the programming time will be somewhat longer.

Customizing the flash file system

If you want to add, remove or change individual files in the flash file system, you can use the Control Panel to do this if you have loaded the flash image which contains a dice application. If you have loaded the RedBoot-only flash image, then you can load files using the CLI serial interface. See the Firmware Development User Guide for details.

6. Details

Development Board
The development board will have a target DICE chip and some combination of connected memories. When the OpenOCD server is connected through the driver and JTAG interface, it can configure the DICE interrupts, memory interfaces, and run state.

These CPU target and development board initializations are contained in configuration files, which have commands written in a minimal form of the Tcl scripting language. Examples of these ‘target’ and ‘board’ configurations are provided in the OpenOCD installation directory. The target and board configurations for DICE EVM’s are listed earlier in this document, and will usually run with custom hardware if the memories are configured similarly to the EVM’s.

JTAG interface
The JTAG interface is a hardware dongle connected to the JTAG port on the development board, and connected to the PC via USB or parallel port. In this document, we describe the Amontec JTAGkey and Olimex USB interfaces.

OpenOCD is informed about which dongle to look for using an ‘interface’ configuration file. Interface configuration files for most available JTAG dongles are included in the OpenOCD installation in the ‘interface’ directory.

Most JTAG interfaces that are compatible with OpenOCD should work for DICE development. The Olimex ARM USB Tiny and Amontec JTAGkey are known to work and others are reported to work. The USB Tiny and JTAGkey are described here as an example.

Driver
OpenOCD interacts with the target DICE through a driver which handles communications to the dongle using its particular i/o transport, such as parallel or USB. The supported driver depends on how the OpenOCD server is built.

OpenOCD server binaries cannot be distributed if they are built to use proprietary drivers, therefore we support the LibUSB version of the OpenOCD server, which can be freely distributed in binary form.

FTDI vs. libusb-win32
Many JTAG dongles use a common USB communications chip (the FT2232 family) to communicate between the JTAG port and the PC. The manufacturer usually provides a proprietary Windows driver for this (FTDI), and the open source community also provides a driver (libusb-win32) that can substitute for the proprietary driver.

The LibUSB driver is somewhat slower than the proprietary ones, but the advantage is that it can be distributed freely in binary form. The LibUSB driver is also 32-bit only, although it can be built for 64-bit systems. Search online for more info about this.

FTClean

Since OpenOCD is GPL licensed, any OpenOCD server application built to use a roprietary driver may not be redistributed. For this reason the user must build the application themselves. Later in the doc is a description of how to build the OpenOCD server for either USB-layer driver. For DICEII developers, contact TCAT for an additional patch that manages the watchdog timer in the background.

Full pre-built OpenOCD/driver distributions are available for the libusb-win32 driver, although the performance of the driver is somewhat less than the proprietary driver. TCAT makes these distributions available for those who don't want to build and install OpenOCD and libusb-win32 themselves.

OpenOCD server

GDB Port

The gdb port is configurable in any of the startup configuration files. The JTAG server listens on the TCP port number defined by **gdb_port** for 'Remote TCP' connection requests from gdb and translates gdb commands for the driver interface. This is a TCP/IP connection through the local host address (127.0.0.1) and the commonly used number for the gdb_port is 3333.

The appropriate version of gdb is provided in the Dice Firmware Development Environment, which is found on the TCAT website.

Telnet Port

Once the OpenOCD server is running and properly connected, the user can interact with the target using a telnet session. The JTAG server also listens on the TCP port number defined by **telnet_port** for Telnet connections. The port number for this can be configured in any of the startup configurations files. This is a TCP/IP connection through the local host address (127.0.01) and the commonly used number for the telnet_port is 4444.

7. Setting up the JTAG interface

The interfaces typically have the ARM 20-pin JTAG connector, and the DICE EVM's and (usually) customer hardware have a 14-pin header, so you'll need a 20-to-14 pin adapter such as:

http://microcontrollershop.com/product_info.php?products_id=1133

http://www.segger.com/cms/jtag-14-pin-adapter.html

Olimex ARM USB Tiny

http://www.olimex.com/dev/arm-usb-tiny.html

Do not install the software from the included CD. Install the OpenOCD server, and use the batch and configuration files as described earlier.

For DICEII targets, see the section below *Using OpenOCD with DICEII*.

Amontec JTAGkey

http://www.amontec.com/jtagkey.shtml

Do not install the software from the Amontec website. Install the OpenOCD server, and use the batch and configuration files as described earlier.

For DICEII targets, see the section below *Using OpenOCD with DICEII*.

Connecting the JTAGkey probe

If you don't have a 20-to-14 pin adapter, you can use the breakout ribbon cable that Amontec includes to make signal connections. The table below shows the minimum connections required. Since the ribbon cable color coding may change, please check the cable colors before following them in the table below.

Amontec 20-pin ribbon cable	DICE EVM 14-pin JTAG header	Signal
pin 1 brown	pin 1	VREF
3 orange	3	TRST_N
5 green	5	TDI
7 violet	7	TMS
9 white	9	TCK
13 orange	11	TDO
20 black	14	GND
nc	12	Pulled up
all others nc	all others nc	

8. Using OpenOCD with DICEII

The watchdog timer in the DICEII chip cannot be disabled, so it must be periodically reset by some means. The application firmware has a thread which periodically resets the watchdog. The RedBoot ROM monitor also has a watchdog thread, so the target stays alive when using serial debugging. However there is no debug monitor running when using JTAG debugging, so it must be periodically reset when the processor is halted, such as during long downloads and when stepping through code.

The OpenOCD server can be modified to periodically reset the watchdog timer. To do this, find the file **src/target/arm7_9_common.c** and make the following changes:

1) Add this function:

```
int arm7_9_handle_target_wd(void *priv)
{
    target_t *target = priv;
    u8 wd_reset[4] = { 0x78, 0x56, 0, 0};

    if (target->state == TARGET_HALTED)
    {
            DEBUG("wd");
            target->type->write_memory(target, 0xbf000004, 4, 1, wd_reset);
    }

    return ERROR_OK;
}
```

2) In the function arm7_9_init_arch_info(), add this line:

```
    /* watchdog reset, 2 second interval */
    target_register_timer_callback(arm7_9_handle_target_wd, 2000, 1, target);
```

3) Rebuild the server as described below.

This is not an issue with later DICE variants, such as DICE-JR and DICE-Mini, and you can use the OpenOCD server provided in the generic installer.

9. Building OpenOCD from the Sources

There are a few cases where you may want to build the server program yourself, such as when patching it for use with DICEII or when you want to build it for the proprietary USB drivers for your JTAG interface to gain a small increase in performance. You may also want to build from the latest sources online in case there is need for a new feature or fix before someone has built an installer package for it.

To build form the sources, you will need a git client to check out the OpenOCD sources from the git repository, and you will need a few other prerequisites in your Cygwin installation.

Devel
- autoconf: Wrapper scripts for autoconf commands
- automake: Wrapper scripts for automake and aclocals
- git: distributed revison control client
- gcc, gcc-core: C compiler upgrade helper
- libtool: A shared library generation tool
- make: The GNU version of the 'make' utility

Libs
- ioperm: support for ioperm()/iopl() functions
- popt: library for parsing cmdline parameters

To update your Cygwin, go to the cygwin\cygwin_mirror directory and run setup.exe. If it asks you to update your setup.exe, then do so and run it again. Then choose Install from Internet, click through the defaults, choose a download site, and then select the binaries for the packages mentioned above.

Newer OpenOCD releases may require additional prerequisites.

LibUSB

A fairly good forum topic on this can be found here:
http://forum.sparkfun.com/viewtopic.php?t=11221&start=0&postdays=0&postorder=asc&highlight=

To build the server with compatibility with most JTAG interfaces, use the following configure when building the server (as described in Freddie Chopin's info.txt):

```
$ ./configure --enable-ft2232_libftdi --enable-gw16012 --enable-parport \
        --disable-parport-ppdev --enable-parport-giveio --enable-presto_libftdi \
        --enable-amtjtagaccel --enable-arm-jtag-ew --enable-jlink --enable-rlink \
        --enable-usbprog --enable-vsllink
```

libftd2xx

This walkthrough is helpful for building the server for use with the ftd2xx proprietary drivers:

http://piconomic.berlios.de/build_openocd.html

10. Troubleshooting

OpenOCD cannot find interface
Verify the drivers are located in the LibUSB-win32 Devices category in the Device Manager.

If they are there, verify that you haven't previously installed the proprietary drivers and run FTClean.

Make sure you are using the correct interface file for your JTAG device.

Memory access errors
Shutdown the OpenOCD server, unplug/plug the JTAG dongle USB connection, power cycle the target board and try again.

Try reducing the jtag_khz value in the configuration file.

Make sure the target is powered.

Connection refused
Make sure there isn't another instance of the server running.

Make sure the target is powered.

Syntax errors
Make sure you are not running an older version of the server.