OLIMEX

OpenOCD | Development Suite

Eclipse Helios
OpenOCD
Zylin

Introduction to OLIMEX ODS

Olimex ODS (OpenOCD Development Suite) is a great tool for programming and debugging ARM chips in the familiar Eclipse environment. It features several open-source tools as well as numerous preconfigured basic projects for OLIMEX boards.

❖ What is Olimex ODS?

To create the OpenOCD Development Suite we combined several open-source tools into one easy to use package:

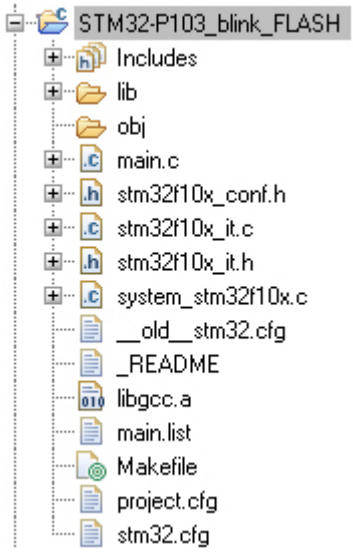
- ***The Eclipse IDE*** – one of the most widely-used open-source IDEs in the world. We used the CDT version of Eclipse Helios as the basis for OlimexODS. You can find all the documentation on Eclipse Helios here: http://help.eclipse.org/helios/index.jsp
- The ***YAGARTO (Yet Another GNU Arm Toolchain)*** toolchain as the compiler. You can find the YAGARTO project here: http://sourceforge.net/projects/yagarto/

  You can find links to extensive documentation in the /yagarto/ folder in ODS

  All the code debugging in ODS is done using **arm-none-eabi-gdb** – you can see all the secific commads with which Zylin calls gdb in each of the projects' own debug configuration.
- ***Zylin Embedded CDT for Eclipse*** http://opensource.zylin.com/embeddedcdt.html
- ***OpenOCD*** – a tool for flashing ARM chips using a number of different JTAG adapters. We do our best to include the latest stable release in each new release of ODS. The best source for information on OpenOCD common comands and practices is the OpenOCD manual

pdf file included in the /openocd/ folder of ODS, as well as the Readme fies of each project.

❖ The structure of an ODS Project



Each project in the default workspace is configured to work with a specific OLIMEX board. The projects usually do somehing basic such as blink LEDs, which makes them perfect to use as a basis for your more complicated projects.

Note: There is a tutorial on how to create a new project as a copy of an existing one (with all the proper configuration) – EclipseTutorials.pdf located in the main ODS directory.

We try to make the structure of the projects as similar to one another as possible. Let's take a look at the structure of the *STM32-P103_blink_FLASH* project:

- The first part of the project name – STM32-P103 shows which olimex board the project is intended for.
- The middle part of the project name hints at what the project actually does – in this case blink the board LEDs.
- The last part of the project name shows which memory region the compiled code is intended for. This could be internal FLASH(most common), internal RAM, external FLASH, external RAM etc. These regions are mapped to a specific address – in ODS this is usually done in the openOCD chip-specific .cfg files(discussed later in this article) or the startup code(.s). If you want to know more about how your chip operates, search for "memory mapping" in your reference manual.
- Every project has a **_README** plain text file with important notes – some are specific to the project, some are common and can be found in all **_README** files.
- **Source code**
  - The **main.c** file – the main entry point for your application after the startup code does its job

- **Stm32f10x_conf.h** – the configuration file for the STM32f1x peripheral libraries, contained in the /lib/inc/ and /lib/src/ directories
- **Stm32f10x_it** files – this is where the interrupt handlers are usually defined and implemented
- The **/lib/** subdirectory also contains startup files, linker scripts and some other core files, needed for a successful compilation.

- **project.cfg -** this is a standard OpenOCD cfg file, loaded when you launch your external tool (your hardware debugger) from the Eclipse interface. Each external tool configured by us calls by default a "project.cfg" file located in the main project dir. Here, the board-specific .cfg file is called. You can put your additional project-specific OpenOCD commands here – there is no need to edit the general board- and target-specific .cfg files bundled with OpenOCD.

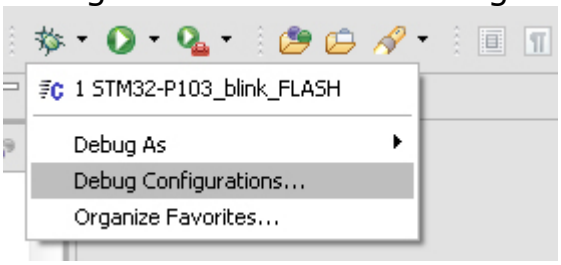
  We aim to provide a board-specific .cfg file for each board we include in the workspace. These files call the chip-specific .cfg files, as well as define some constants rekated to memory. For example, the **olimex-stm32-p103.cfg** file looks like this:

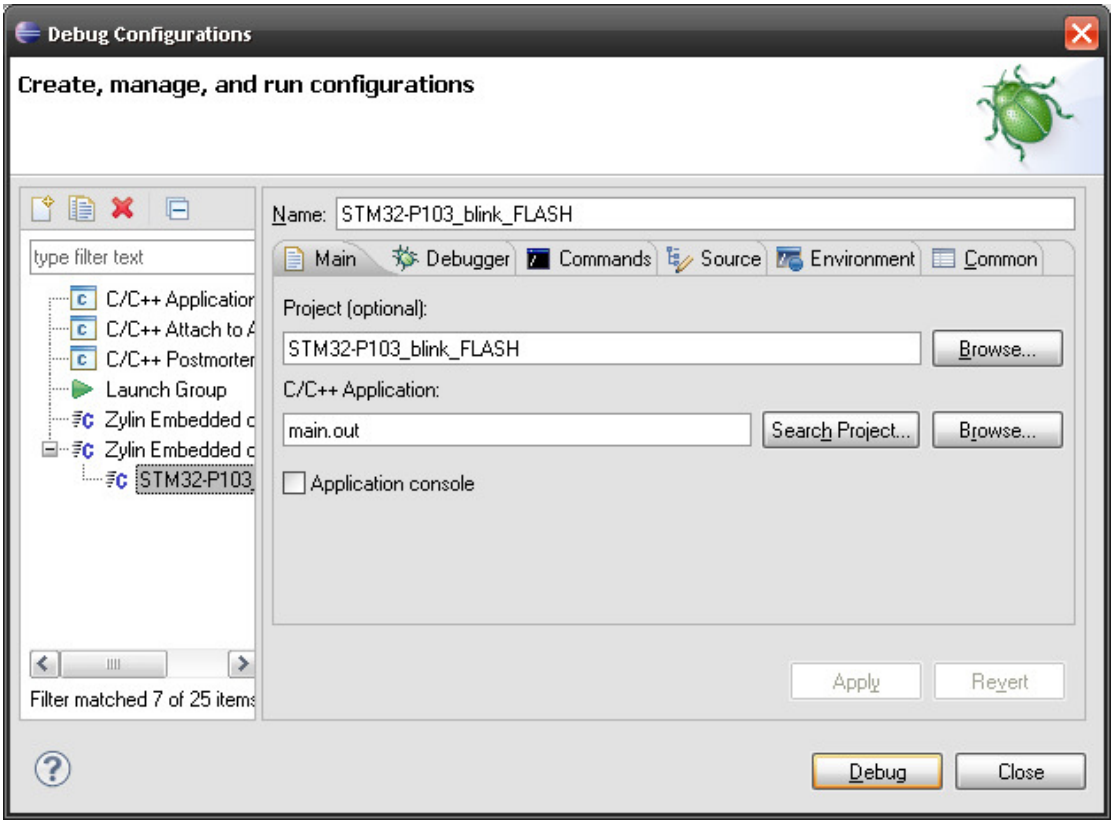
  ```
  # Work-area size (RAM size) = 20kB for STM32F103Z
  set WORKAREASIZE 0x5000
  source [find scripts/target/stm32f1x.cfg]
  ```

  A lot of detailed information on the .cfg files heirarchy and structure can be found in the OpenOCD manual.

- The **Makefile** – each project comes with a specific Makefile. You may need to edit the makefile if you wish to include additional files and folders. Refer to the GNU Toolchain documentation for Makefile-specific questions.
- The **Debug Configuration –** each project has a specific set of commands passed by Zylin to GDB(and through GDB to the already launched and successfully connected OpenOCD) to enable visual debugging of the project's source code using all of the powerful tools of the Eclipse environment.

You can access the options for the project debug configuration in the following way:





The different tabs contain all the settings for the GDB debugger. The one which is usually specific to a project is the **Commands** tab, where the commands for GDB and the OpenOCD flashing process are. In this case:

```
target remote localhost:3333
monitor reset halt
monitor wait_halt
monitor sleep 100
monitor poll
monitor flash probe 0
monitor flash write_image erase main.bin 0x08000000
monitor sleep 200
monitor soft_reset_halt
```

monitor wait_halt
monitor poll
thbreak main
continue

Here, GDB connects to OpenOCD via port 3333, sends the the commands for flashing and resetting to OpenOCD(always preceeded by "monitor"), inserts a temporary hardware breakpoint at the beginning of **main()** and starts the debugging process.

At this point the Eclipse GUI switches to the Debug view and begins highlighting your code. Now you can make use of all of Eclipse's debugging tools – step through your code, watch variables, registers and many more.

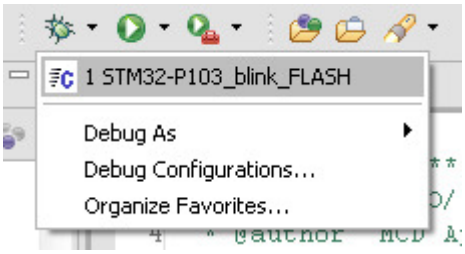
❖ Launching your project

The usual steps for launching a project are:

- **Build** your project
- **Connect** your hardware debugger to your PC and the board
- **Power on** your board - depending on the board this could be done through USB, some ext. power supply, the JTAG itself etc.
  Important: Always supply your board as described in the board's manual. Too much power and you will damage it; too little and the flashing process will fail.
  Always consult the _README for the specific project, there are cases in which a complete relaunch is needed for the project to start debugging successfully for the second time.
- **Launch** the appropriate External Tool from the Eclipse interface, like so:

  This usually needs to be done only once while you are working on a specific project if there are no errors with debugging. Always consult the _README and the Console output.

- **Launch the debug configuration** with your project's name:



- At this point the Eclipse IDE will switch to the **Debug view** and you can start debugging your code.

❖ Notes

If you have any questions or wish to report a bug, you can contatct Olimex support via e-mail:
support@olimex.com

or write in our forum:
https://www.olimex.com/forum/

20 March 2013