Freedom Studio User Manual

2021-04

Table of Contents

Introduction

Freedom Studio is an integrated development environment which can be used to write and debug software targeting SiFive based processors. Freedom Studio is based on the industry standard Eclipse platform and is bundled with a pre-built RISC-V GCC Toolchain, OpenOCD, and the freedom-e-sdk. The freedom-e-sdk is a complete software development kit targeting SiFive bare metal processors.

See the What's New section for a summary of new features in this release.

Product Overview

This section will describe the individual components used in a release.

The major versions of the Eclipse feature plugins are as follows:

- Eclipse 2020.12
- Java 15 JDK (https://adoptopenjdk.net/)
- Eclipse C/C++ Development Tools
- Git Integration for Eclipse (eGit)
- Terminal View Core
- SiFive RISC-V Cross Compiler
- SiFive OpenOCD Debugging
- SiFive J-LINK Debugging
- SiFive QEMU Debugging
- SiFive freedom-e-sdk
- SiFive Trace Decoder

Setting Up Freedom Studio

Download and Install

Freedom Studio can be downloaded from the SiFive website at the following link:

https://www.sifive.com/boards/#software

Downloads are provided for Windows, MacOS, and Linux.

Windows Installation

Freedom Studio Package Path Lengths and Windows

The Freedom Studio distribution archives contain long deep paths. All paths are less than the Windows MAX_PATH limit of 268 characters (the longest is around 199 characters). This means that the native Windows extraction tool can successfully extract the Freedom Studio archive if the sum of the path length to the installation location and the deepest path in the archive is less than MAX_PATH.

If you want to install to a location that may exceed MAX_PATH then you must use a third-party extraction tool (like 7-Zip) to extract the archive and ensure that Windows long paths are enabled.

Let us cover two important points:

Point #1: It is important that you choose an installation path that does not contain spaces. Freedom Studio will check the installation path when started and will warn you if it detects a path that contains any space characters.

Point #2: You should consider enabling Windows Long Path support. You should do this before extracting the product archive. The Freedom Studio installation folder may contain paths that are deep enough to exceed the "legacy" MAX_PATH (=260) character limit imposed by Windows. This limit is still enabled by default, but Windows 10 (starting with version 1607) allows for disabling this limit by installing a specific register key/value using the Windows regedit tool:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\FileSystem
LongPathsEnabled REG_DWORD = 0x1
```

To simplify this process you can download the following registry file and double-click it to install this key automatically:

```
https://static.dev.sifive.com/dev-tools/FreedomStudio/misc/EnableLongPaths.reg
```

If you still have problems extracting the archive after enabling Long Path Support contact support@sifive.com

More information on this topic can be found here:

```
https://docs.microsoft.com/en-us/windows/desktop/fileio/naming-a-file#paths
```

You can install multiple versions of Freedom Studio on your system and use all of them.

We recommend that you keep the installation path as short as possible. We suggest creating a folder at the root of your installation drive called "FreedomStudio" (no spaces). Then inside that folder you can install multiple versions of Freedom Studio into subfolders. Like:

```
c:\FreedomStudio
  |
  +- FreedomStudio-2019.03
  +- FreedomStudio-2021.03
```

The product zip archive extracts to a long folder name (for instance `FreedomStudio-4.7.2.2019-03-4-win32.win32.x86_64`). We recommend that you shorten the folder name using a naming scheme similar (or identical) to the one shown above.

We recommend using a tool like 7-Zip to handle large zip archives on Windows. Unzip the downloaded zip archive to a directory on your PC by right-clicking on the zip file and selecting "Extract All". After unzipping the bundle, you can open Freedom Studio by double-clicking on FreedomStudio.exe in the installation directory.

For more information about setting up SiFive development platforms, please consult the platform's User Guide and Windows Board Setup.

MacOS Installation

Important: Freedom Studio must be installed into the Applications folder. You can do this by dragging the FreedomStudio folder to the Applications folder in the Finder.

Extract the Freedom Studio tarball by double clicking the bundle. Freedom Studio is not a signed macOS application and therefore may present an error when running. In order to run Freedom Studio on macOS it may be necessary to open Freedom Studio for the first time as described in this URL: https://support.apple.com/kb/PH25088?locale=en_US

It is also possible to execute this command line to remove the extended attribute marking the .app file for quarantine:

```
$ xattr -d com.apple.quarantine FreedomStudio.app
```

Start Freedom Studio by clicking on FreedomStudio.app found in the FreedomStudio folder which was just extracted.

For setting up SiFive development platforms, please consult the platform's User Guide and macOS Board Setup.

Linux Installation

Important: It is important that you choose an installation path that does not contain spaces. Freedom Studio will check the installation path when started and will warn you if it detects a path that contains any space characters.

Important: Starting with FreedomStudio 2019.08, Freedom Studio will no longer run on CentOS6 because the upgraded Eclipse platform (2019.06) only supports GTK3, and GTK3 is not available on CentOS6.

Important: Starting with FreedomStudio 2021.03, Freedom Studio will no longer run on Ubuntu 16.04

Extract FreedomStudio.tar.gz to the desired folder using the following command:

```
tar xzf /path/to/FreedomStudio.tar.gz
```

For setting up SiFive development platforms, please consult the platform's User Guide and Linux OS Board Setup.

Tools Setup

Freedom Studio will automatically detect its installation path on the first run and configure itself to use the bundled tools. If, for any reason, Freedom Studio was not able to detect the bundled tools, it will prompt the user to enter the tool paths directly with a dialog box. If prompted, be sure to select the "bin" directory which contains the tool binaries. These paths will set the global defaults used by Freedom Studio.

The tool paths can be changed at any time by clicking the following:

Windows and Linux: Window → Preferences → Freedom Studio

MacOS: Freedom Studio → Preferences → Freedom Studio

The tool path preferences can be set at 3 different scopes: Global, Workspace, and Project. Global scope sets the default for the installation and is the lowest priority. Workspace scope allows you to set the toolchain preferences specific to a given Workspace and will override the Global setting. Project scope, which can be set by right clicking a project in your workspace and selecting *Properties → Freedom Studio*, allows you to set preferences on a per-project basis. Project scope always takes priority over Global and Workspace.

This flexibility allows the user to easily work with several different tools installed on the same system while still maintaining project portability.

Getting Help

Knowledge Base Buttons

Freedom Studio has Knowledge Base buttons and links in various places. Pressing these buttons or links will open the SiFive Customer Knowledge Base in your browser with

related topics automatically listed. These buttons can be disabled and hidden on the *Freedom Studio → Assistive Feature* preference page.

Video Buttons

Freedom Studio has Watch Video buttons in various places. Pressing these buttons will open related how-to videos in your default browser. These buttons can be disabled and hidden on the *Freedom Studio → Assistive Feature* preference page.

Freedom Studio Bug Reports

See the Freedom Studio Bug Report Generator chapter for a handy way to gather many of the resources that will help us quickly resolve bugs you may have found. These bug report packages can be sent directly to SiFive through the customer support portal.

SiFive Forums

From the Help menu, select Open SiFive Community Forums. This will open your default browser to the SiFive Forums homepage.

SiFive Customer Support Portal

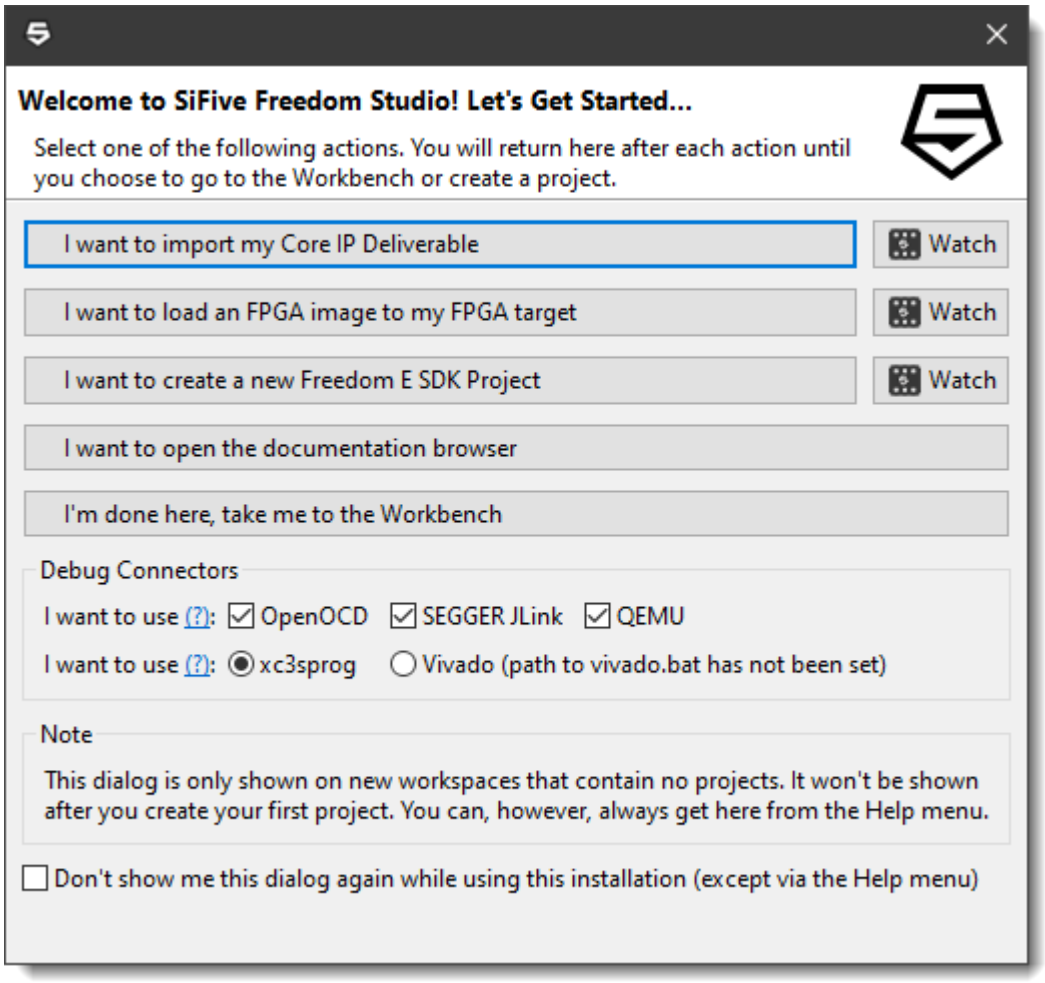
If you have a customer support portal account, you can launch the support portal from the Help menu by selecting "Open SiFive Customer Support Portal". This will open the portal in your default browser.

Other Resources

The Help menu has several other entries that can be helpful.

The Let's Get Started Dialog

When you start Freedom Studio with a new clean workspace you will be presented with the **Let's Get Started** dialog. This dialog is simply an easy way to get started with common first-time tasks.



Quick Actions

Some quick action buttons have a "Watch Video" button next to them. Pressing this button will open a how-to video link in your browser.

- **I want to import my Core IP Deliverable**
  Choose this option if you have a core IP deliverable that you would like to start working with. With this option you will select an IP Deliverable package (a tar.gz file) and Freedom Studio will create a new IP project from the package. The wizard will optionally offer to program an included MCS file or BIT file to a connected target; create a new software project from the freedom-e-sdk embedded in the package; and create a debug launch configuration for the example project. (See: IP Projects)

- **I want to load an FPGA image to my Arty board**
  Choose this option if you want to get started by programming an MCS or BIT file to your Arty board. You will also have the choice to jump right into creating a project at the end of the programming process. (See: FPGA Programming)

- **I want to create a new Freedom E SDK project**
  If you have a HiFive series select this option. (See: Create a Freedom E SDK Software Project)

- **Open the documentation browser**
  Choose this option to open the documentation browser. From here you can dig into all the documentation bundled with Freedom Studio.

- **Just take me to the workbench**
  If you don't want to start with any of the options listed above, choose this option and you'll be taken to your new clean workspace.

Debug Connectors

The Debug Connectors section of the Getting Started dialog lets you specify which debug connectors you want to use and how you prefer to program FPGA images. Check those you want to use and uncheck those that you will not be using. Uncheck items will no longer show up in the IDE and will help reduce the amount of UI clutter.

You can change these settings at any time using this dialog (from the Help menu) or from the Debug Connector Preference Page.

The Freedom Studio Environment

Workspaces

Eclipse uses workspaces to group together a set of related projects. Eclipse workspaces allow for a lot of flexibility in how to organize your projects. For example, it is possible to have a workspace which contains only a single project. It is also possible to have a workspace which contains multiple related projects such as a library project and an application which depends on that library.

Switching workspaces is accomplished by selecting *File → Switch Workspace*.

When starting Freedom Studio, Eclipse will prompt you to select a workspace. Freedom Studio will remember the locations of previously selected workspaces.

Important: When choosing a workspace location do not choose a location that contains spaces in the path.

Eclipse Perspectives

Eclipse uses perspectives to group windows together which are collectively useful for a given task.

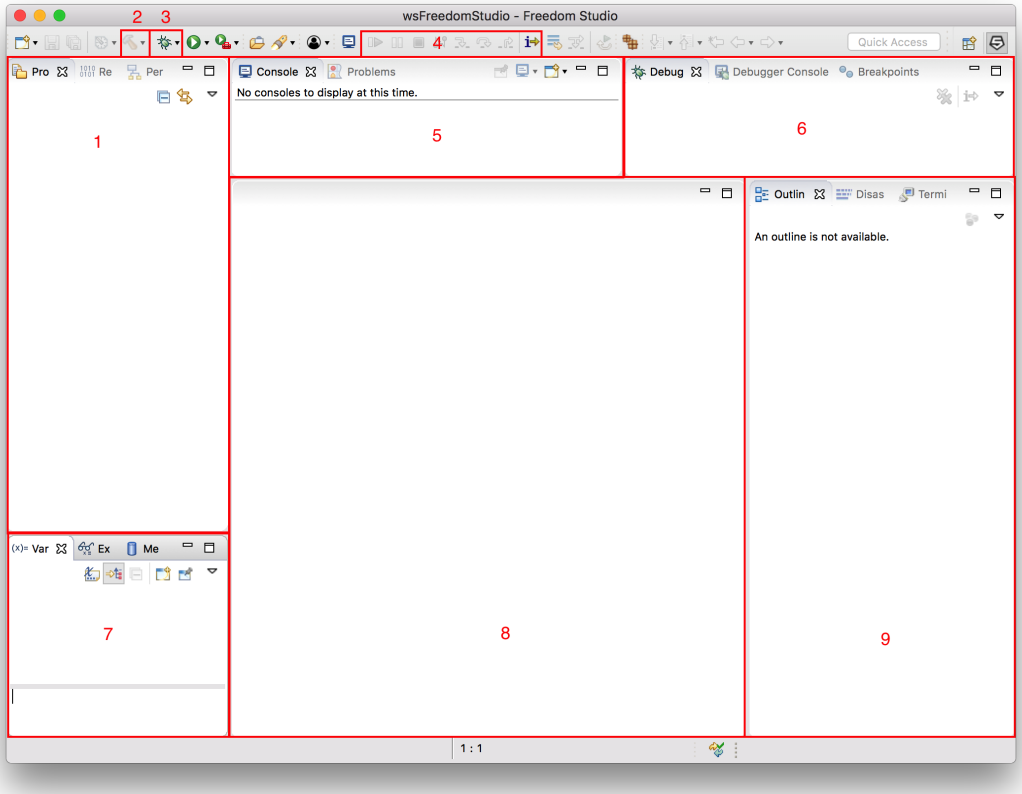
Freedom Studio ships with its own SiFive perspective which can be useful for both programming and debugging. Please see Section The SiFive Perspective for a detailed description of the SiFive Eclipse perspective.

Freedom Studio also ships with the standard Eclipse perspectives: C/C++ , Debug, and Git. From Eclipse, you can change perspectives by clicking *Window → Perspectives → Open Perspective*.

Perspectives are user customizable and persistent to a workspace.

The SiFive Perspective

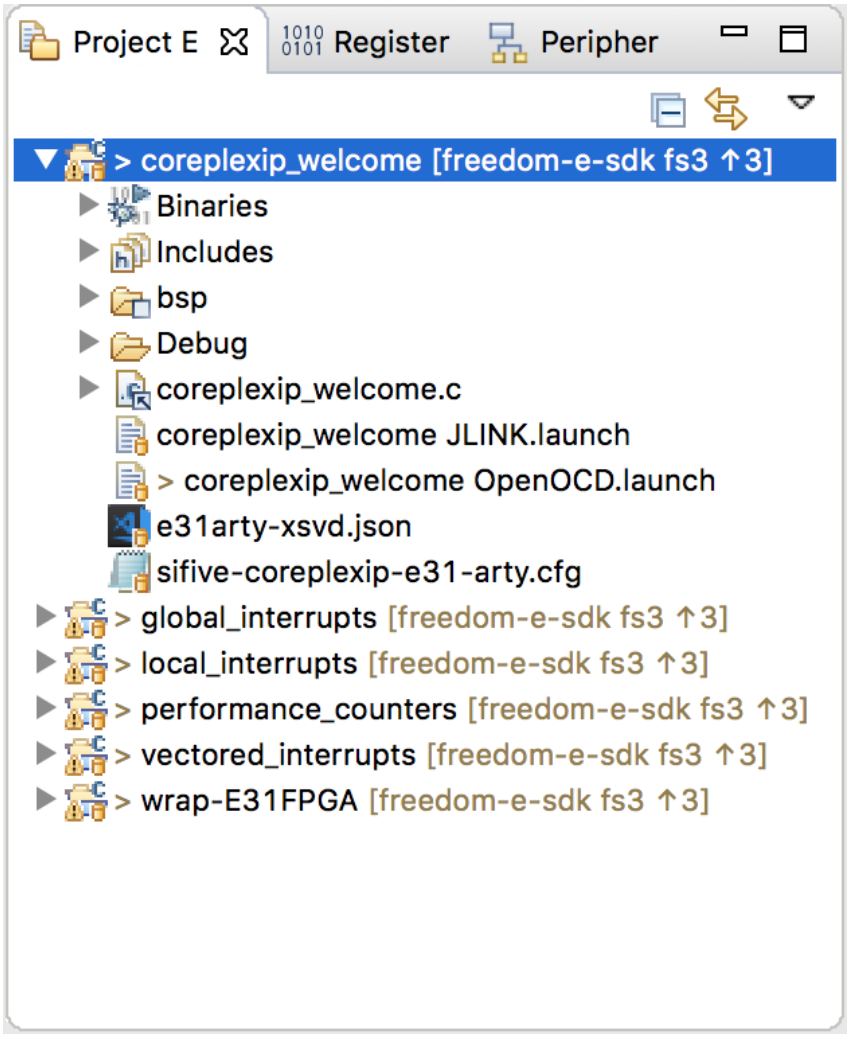
The SiFive Perspective.



1. Project Explorer, Register, and Peripheral Views. These views are described below.
2. Build Toolbar Button. Pressing this button will build (compile) the active project.

3. Debug Toolbar Button. The down arrow next to the bug lets you pick a specific configuration.
4. Debug Control Toolbar Buttons. These buttons are used for debug run, halt, and stepping control.
5. Console. These views display useful information when building applications.
6. Breakpoint and Debug Views display useful information when debugging applications.
7. Variable, Expression, and Memory Views. These views are described below.
8. Editor View is used to edit source code.
9. Outline, Disassembly, and Terminal Views are described below.

Project Explorer

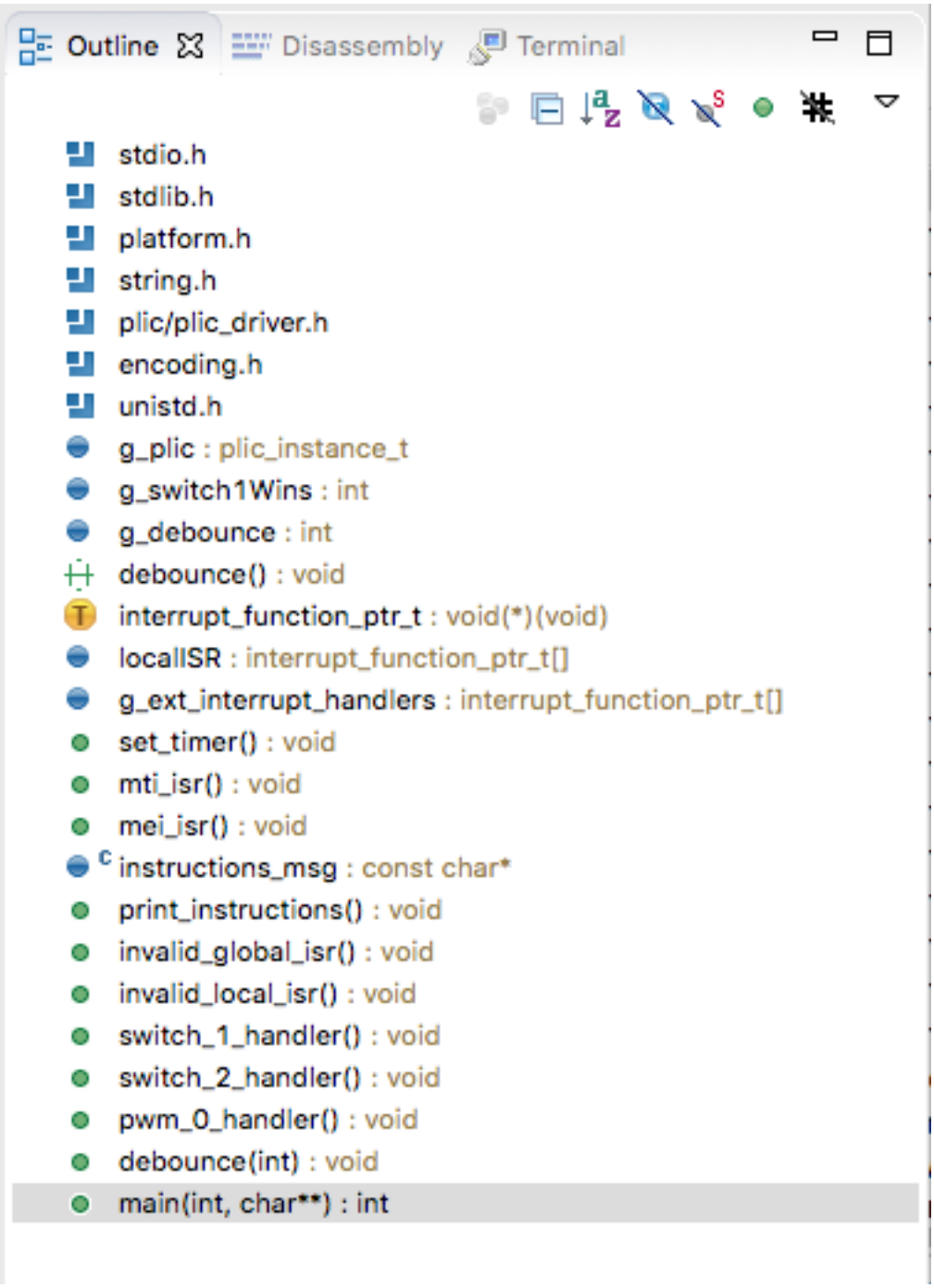


The Project Explorer view displays projects in the workspace. Use this view for opening, editing, and creating new project source files. If a project contains files under revision

control, Project Explorer will also display information regarding the repositories and branches.

Editor, Outline, Disassembly

The Editor and Outline views are used to write and navigate code. The Editor also provides useful contextual information for your code. Hovering the mouse over statements will reveal pop-ups which expand macros, evaluate variables and structures, provide function definitions, etc. Double-clicking a line number in the editor will set a breakpoint at that line.

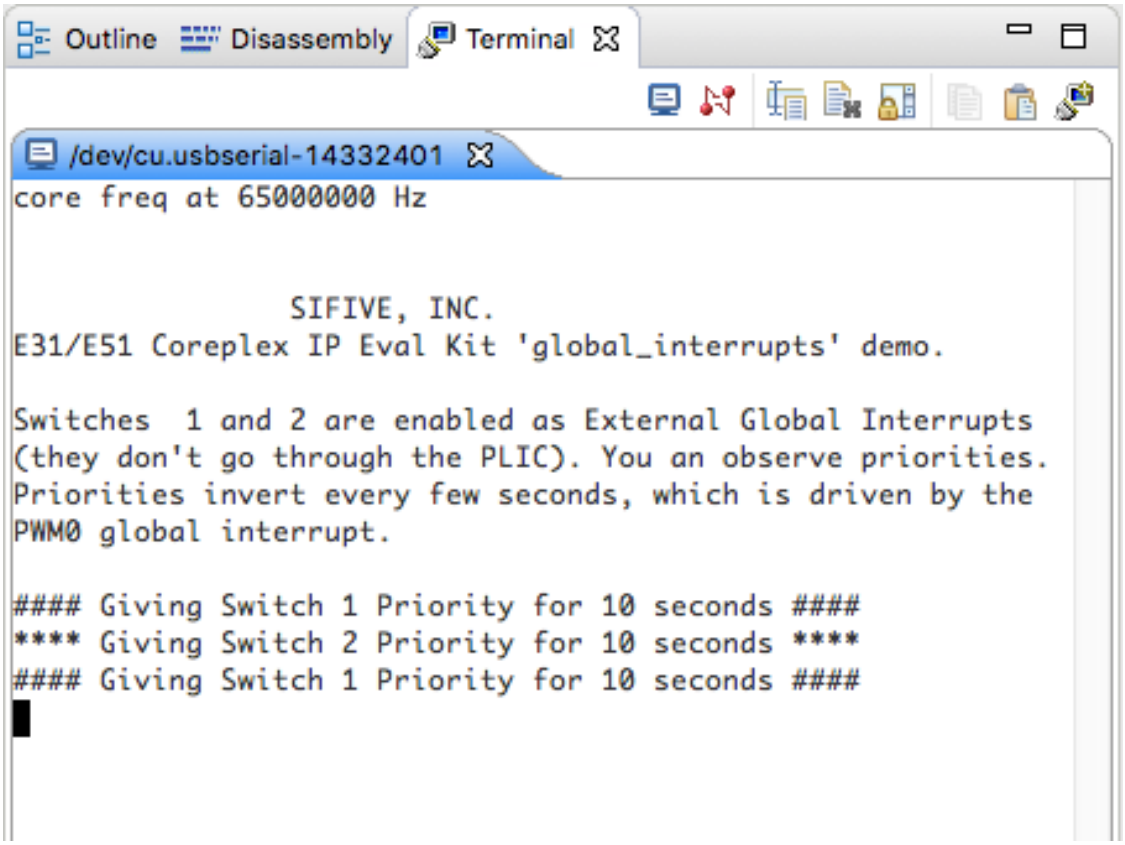


The Outline view, shown in above, gives a "top-level" view of the active file in the editor including functions, types, constants, etc… Clicking on an item in the Outline view will take you to that item's location in the source code.

Terminal

The Terminal view, shown below, can be used to display a local terminal, a serial terminal, or ssh into a remote machine. The serial terminal allows the user to view serial output,

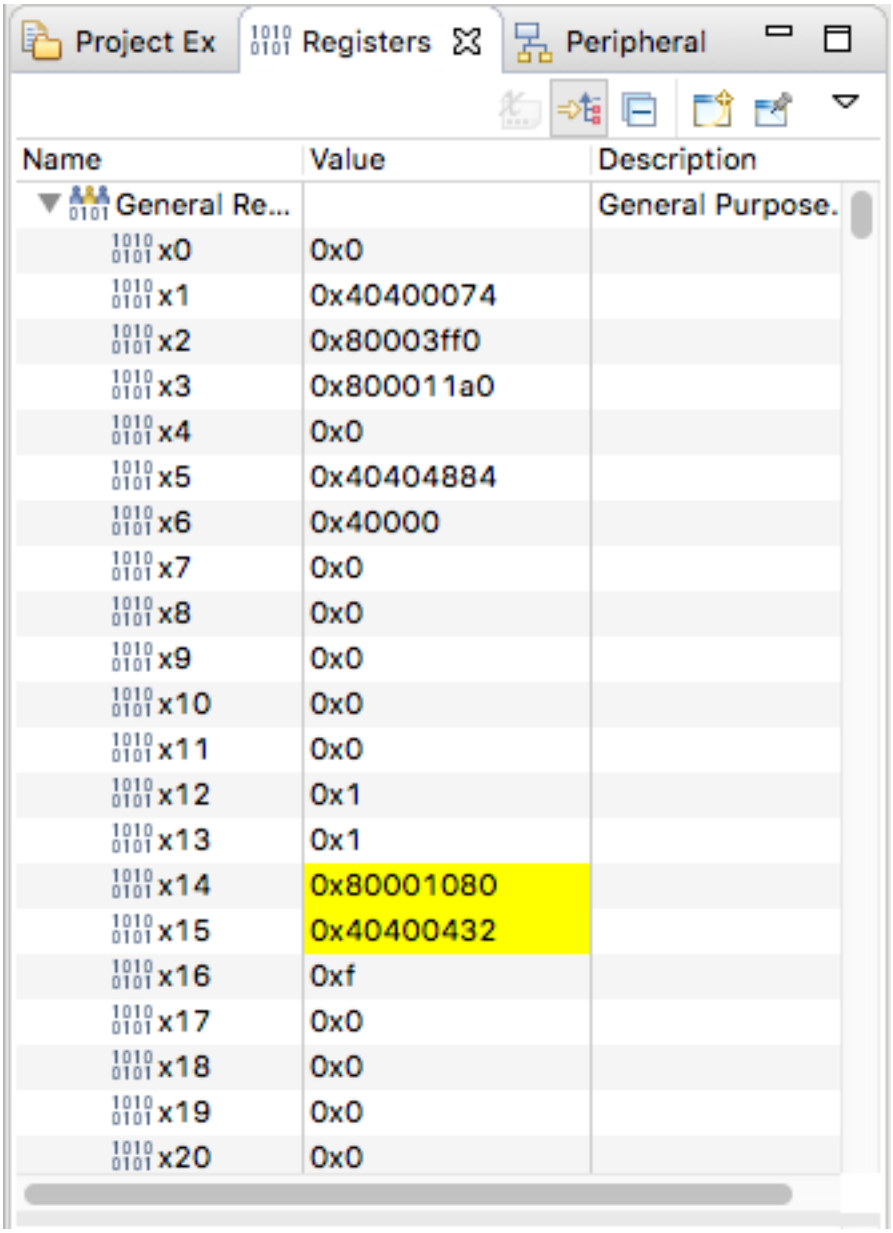
such as that from a SiFive development board, without leaving the development environment. On MacOS and Linux platforms, it is possible to open serial port directly, or open a local terminal and run GNU Screen.



Breakpoints

The Breakpoints view allows for creating, enabling, and disabling of breakpoints. You can set a breakpoint's properties by right-clicking on a breakpoint and selecting "Properties". From the properties menu, you can set properties such as breakpoint type (hard, soft), and ignore count.

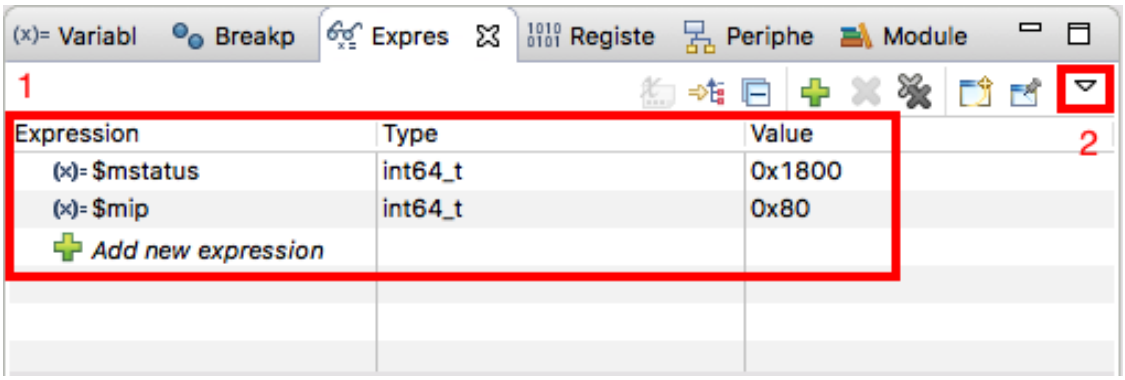
Registers



The Registers view displays the integer and floating point register files. It is possible to write to registers by double-clicking their value field. While stepping through code, the Registers view will highlight registers as they change.

Expressions

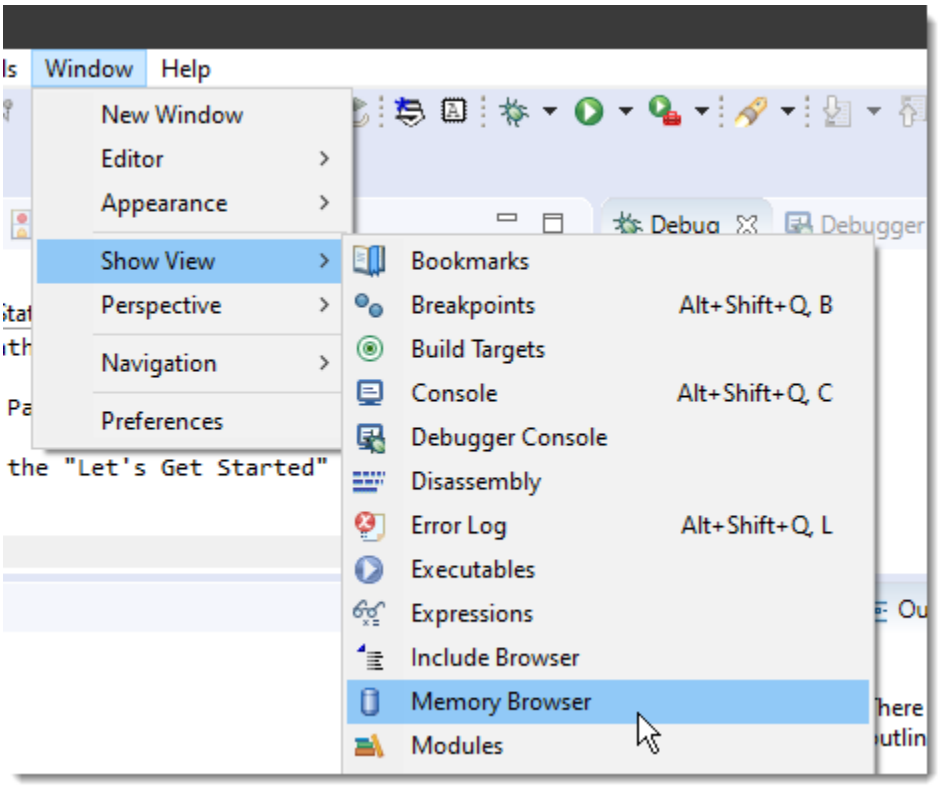
The Expression view allows you to view any variable within scope. In addition to variables, it is possible to use this view to see the current value of CSRs on your device. The Expression view, along with other eclipse views which display variables and memory, allows for changing the value format (for example to hexadecimal). The format can be changed by clicking the down arrow marked with "2" in screenshot:


(x)= Variabl
Breakp
Expres
1010 0101 Registe
Periphe
Module
1
2
Expression
Type
Value
(x)= $mstatus
int64_t
0x1800
(x)= $mip
int64_t
0x80
Add new expression

Memory Browser (instead of Memory View)

The SiFive Perspective now uses the Memory Browser by default for examining target memory. There are known problems with the Memory View that can cause Eclipse (and thus Freedom Studio) to hang. We do not recommend using the Memory View any longer.

If the Memory Browser is not open, you can open it via the Main Menu | Window | Show View menu:



IP Projects

Alongside Software Projects, Freedom Studio uses a project type called “IP Projects”. IP Projects are created by importing an IP Deliverable package. Once imported, you can use Freedom Studio to perform actions on the IP package assets.

Creating a new IP Project

There are two ways to create an IP Project:

1. Import an IP Deliverable package. You will use the “IP Project from IP Deliverable” wizard to import your package.
2. A clone of the open sourced freedom-e-sdk can be easily converted to an IP project to enable all the Freedom Studio integrations.

IP Project from IP Deliverable Wizard

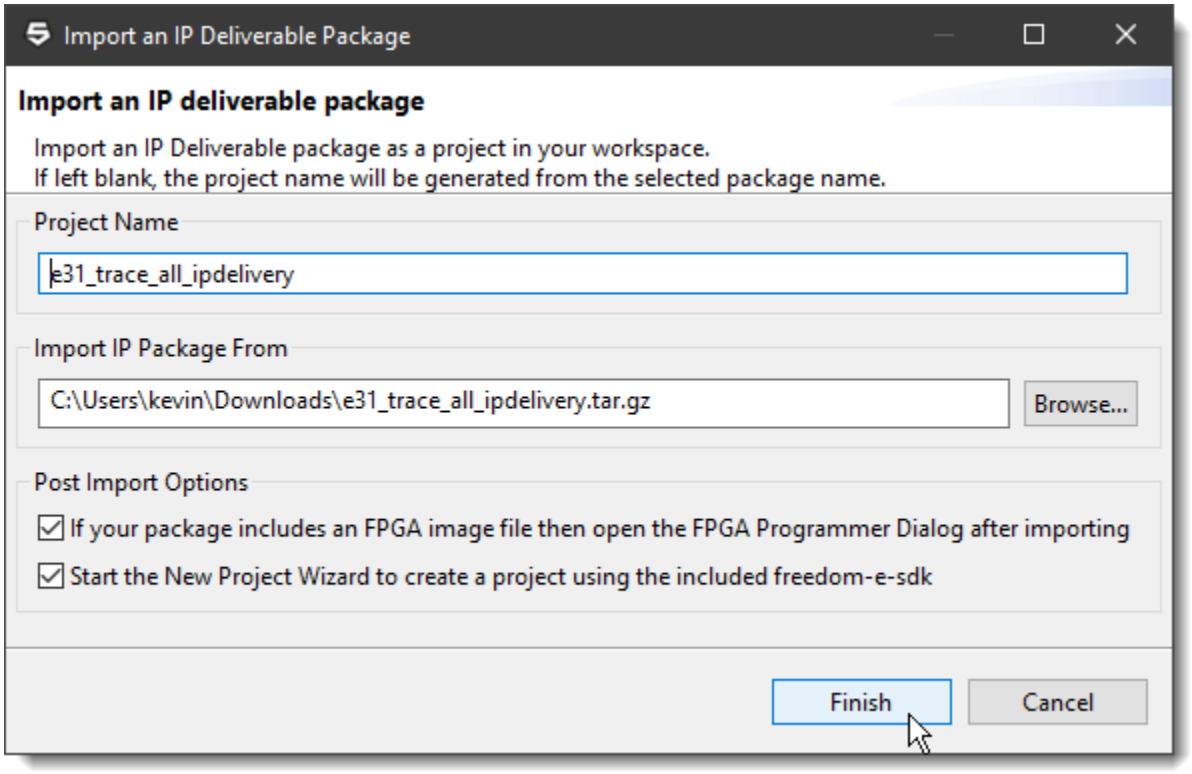
This wizard is accessible from all the usual spots in Freedom Studio:

- The “Let’s Get Started” Dialog (via the Help menu)
- The SiFiveTools menu
- Main Menu -> File -> New -> IP Project from IP Deliverable
- Project Explorer Context Menu
- New Workspace Project Explorer Menu

Before creating a new IP Project you should have an IP Deliverable tarball. If you do not, go to the SiFive Core Designer website and create an awesome SiFive RISC-V core design. When you receive your IP deliverable pack, return here to continue.

Open the Wizard

Open the “IP Project from IP Deliverable” wizard using any of the commands listed above. This wizard has only a single page:



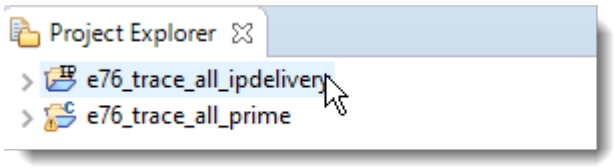
A project name will be generated automatically from the name of the IP tarball. You can accept this name or enter your own name.

At the bottom of the page are two options:

1. Option 1 tells Freedom Studio to open the Programmer Dialog to program the MCS or BIT file included in the IP package. The dialog will default to the MCS file (if one is found), but you can choose the BIT file (if one exists) from the dropdown selector.
2. Option 2 tells Freedom Studio to start the Freedom E SDK Software Project wizard when the import is complete.

Use the “Browse...” button to locate and select the IP project tar.gz file. Give the project a name (or accept the generated name) and click the [Finish] button.

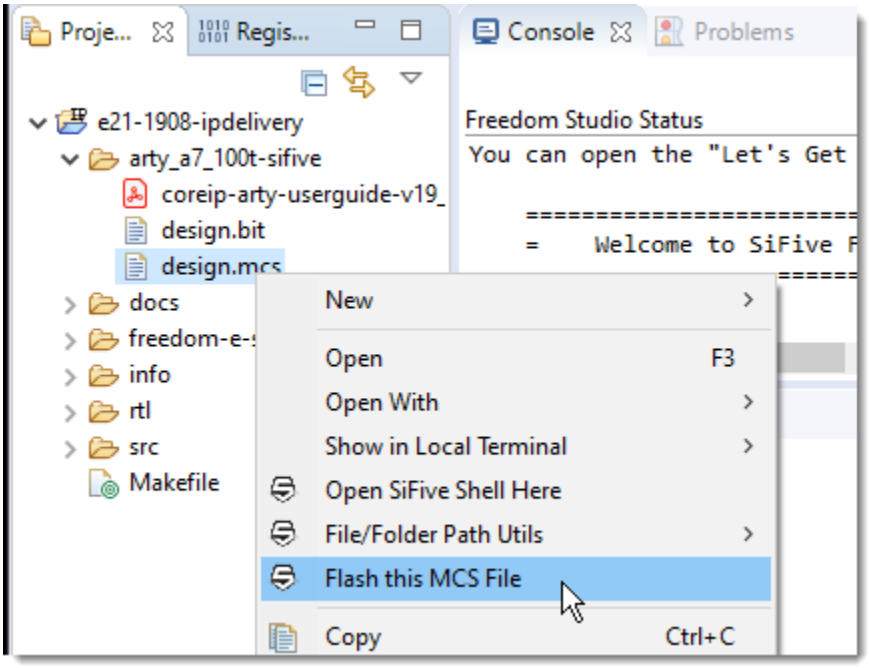
IP Projects are denoted in the Project Explorer with a small “IP” icon in the upper-right corner of the project icon.



Working with the IP Project

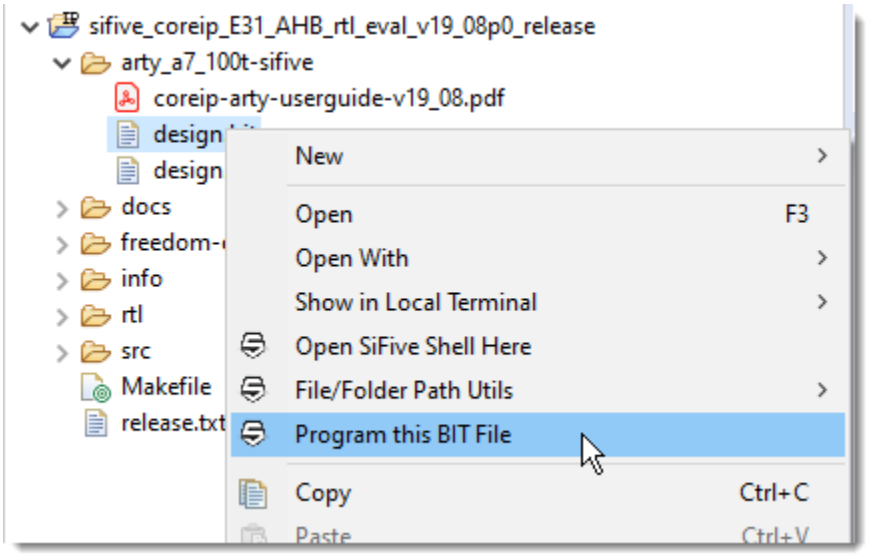
Now that you have a new IP Project, let us do stuff with it. You can:

- **Flash the included MCS file:** Double-click the MCS file, or right-click on the MCS file in the project and selecting "Flash this MCS File"



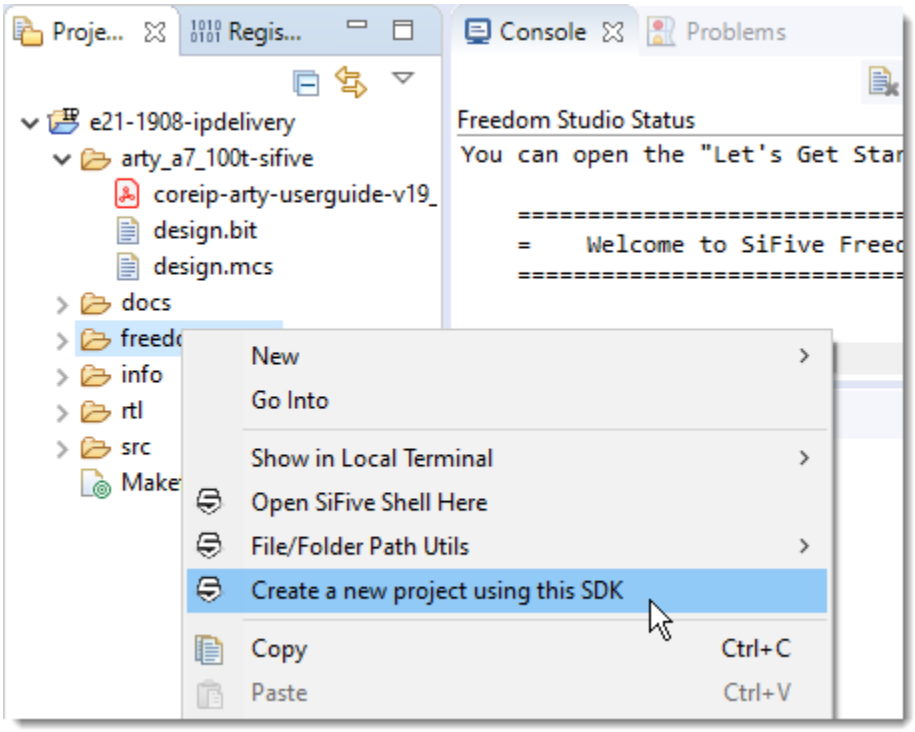
This will open the FPGA Programmer Dialog with the selected MCS file ready to go.

- **Program the included BIT file:** Double-click the BIT file, or right-click on the BIT file in the project and selecting "Program this BIT File"

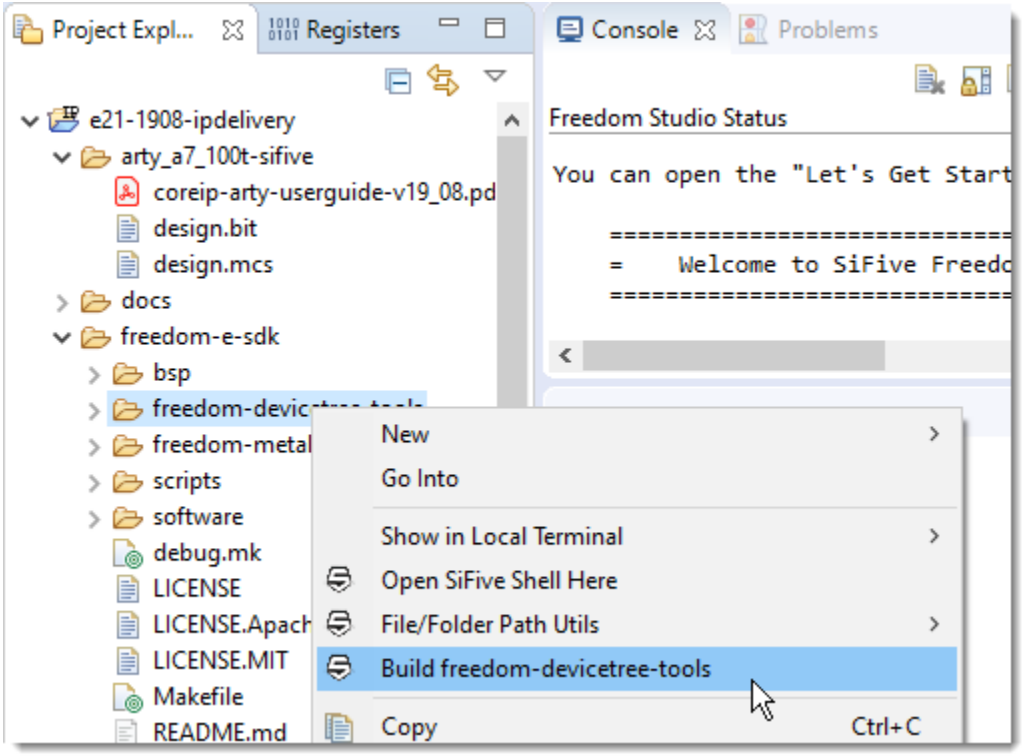


This will open the FPGA Programmer Dialog with the selected BIT file ready to go

- **Create a new Freedom E SDK Software Project:** Right-click on the project folder or the freedom-e-sdk folder and select “Create a new project using this SDK”

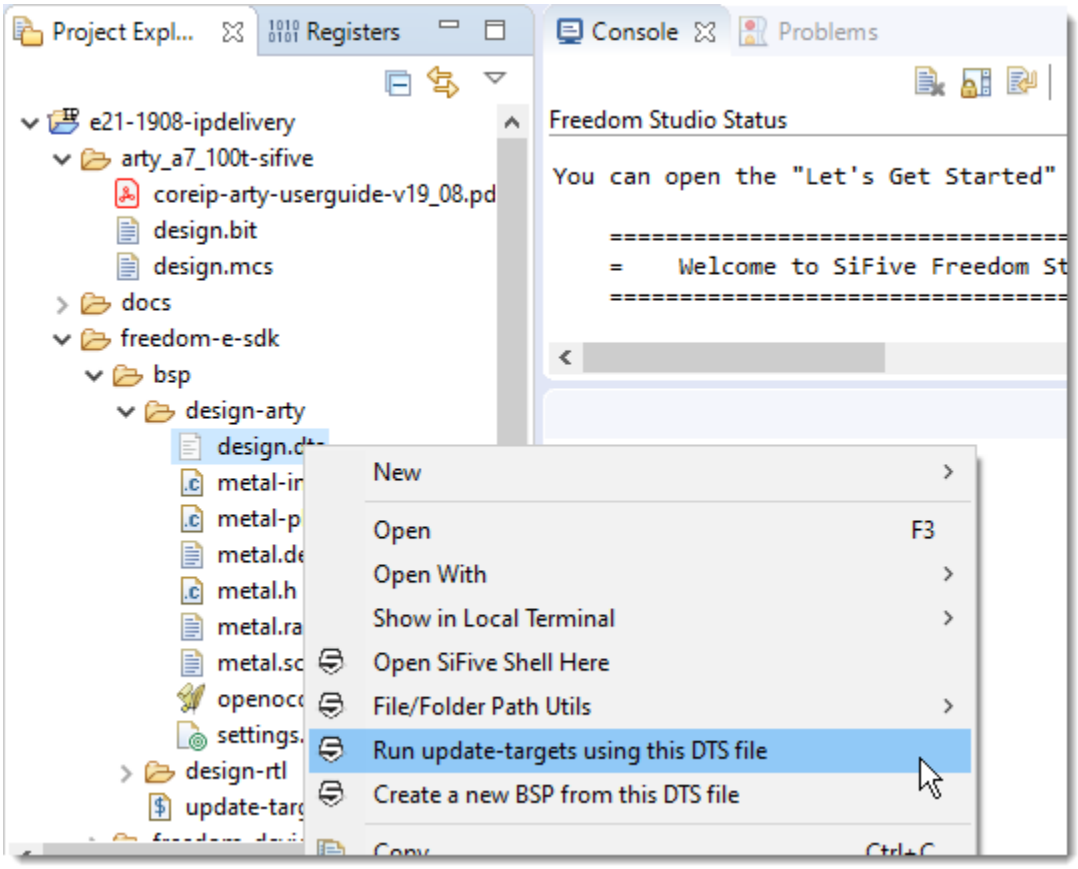


- **Build the freedom-devicetree-tools**: right-click on the “freedom-devicetree-tools” folder (found under the freedom-e-sdk folder) and select “Build freedom-devicetree-tools”



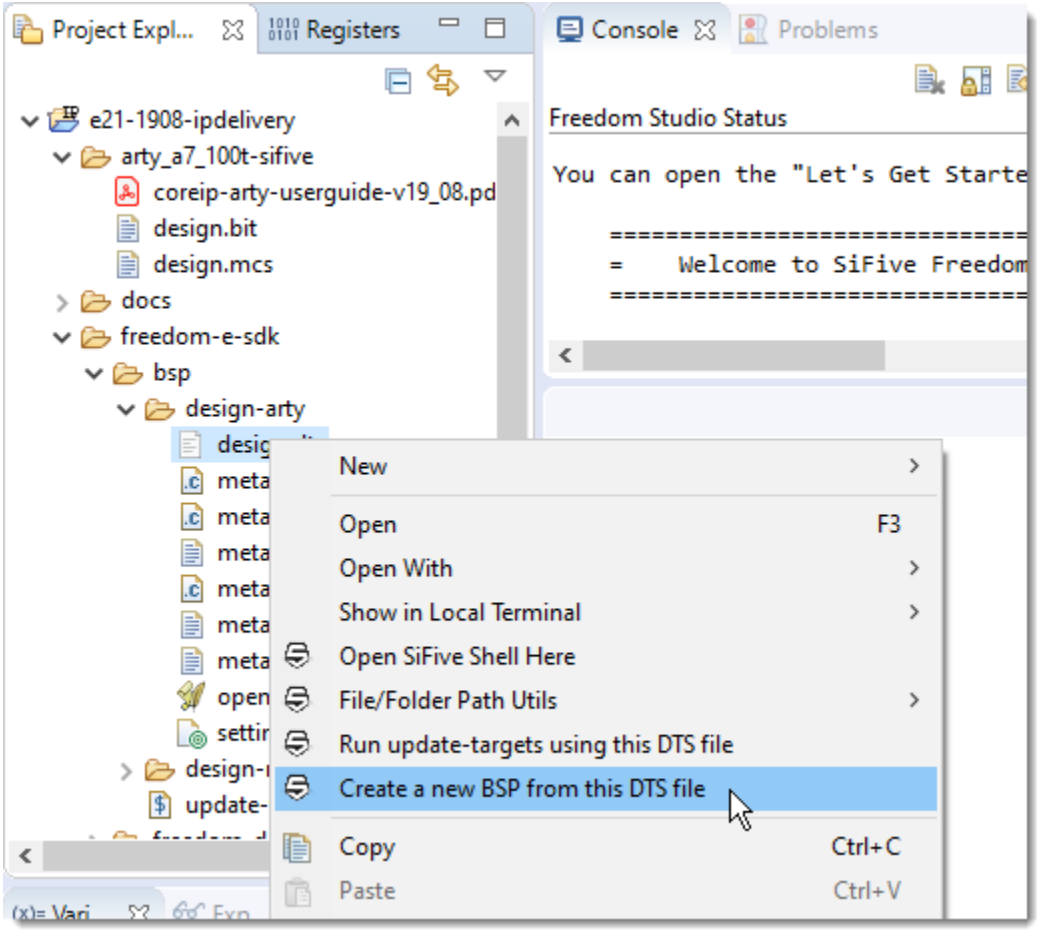
Building these tools requires that several native packages be installed on the host system. On Windows and MacOS Freedom Studio will offer to install these packages if they are not detected (they are not included with the Freedom Studio installation). On Linux, manual installation of these packages is required. See the [freedom-devicetree-tools github project] for details on which packages are required.

- **Rebuild your BSP**: [This only applies to IP packages <= 2019.08] If you have edited your BSP DTS file, right-click on the DTS file and select “Run update-targets using this DTS file”

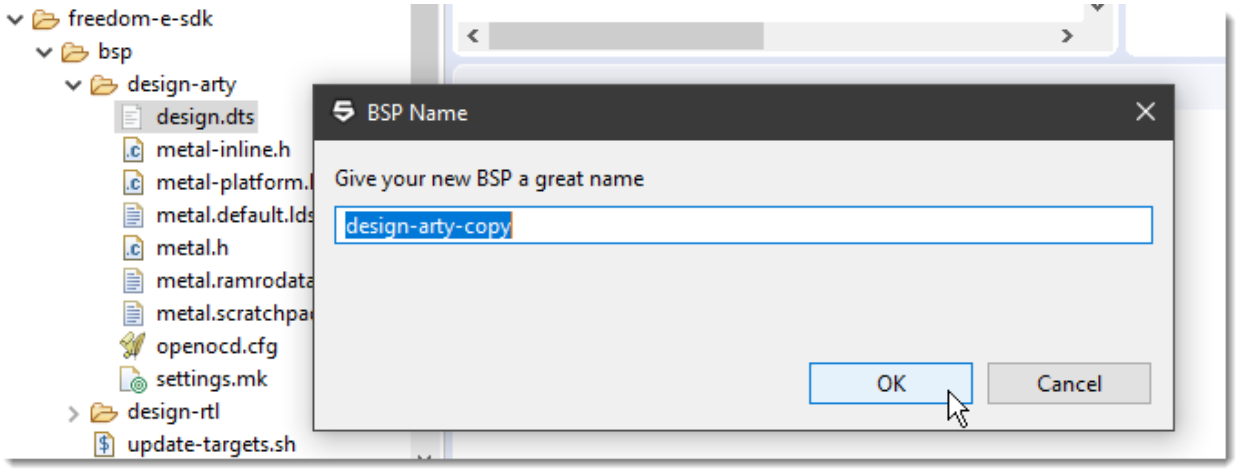


This command requires that the freedom-devicetree-tools are compiled. If they are not, Freedom Studio will ask if you would like to compile them first, then continue updating the BSP.

- **Create a new BSP from an existing BSP**: Right-click on a DTS file in a BSP folder and select "Create a new BSP from this DTS file"



You will be prompted to give your new BSP a name.



The BSP type (Arty or RTL) will be determined by the existing settings.mk file. If for

some reason the settings.mk is not present or does not specify the type, Freedom Studio will prompt you for the type of BSP to create.

When you click OK Freedom Studio will create a new BSP folder (a sibling to the existing folder) and automatically run update-targets on the new BSP to generate the BSP support files.

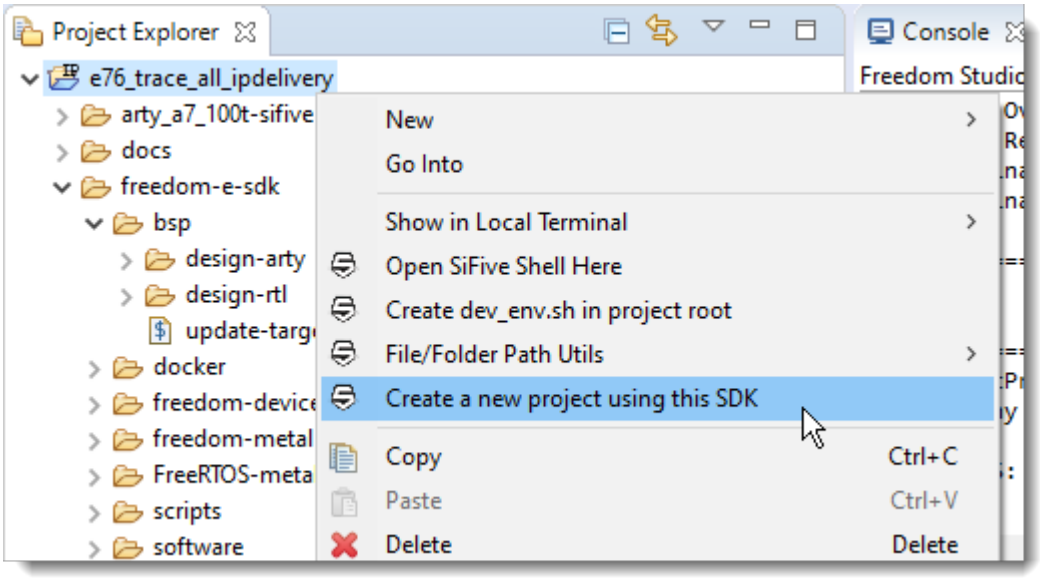
Freedom E SDK Example Software Projects

Creating an Example Software Project

Creating a new Freedom E SDK Project is very simple. There are multiple ways to start:

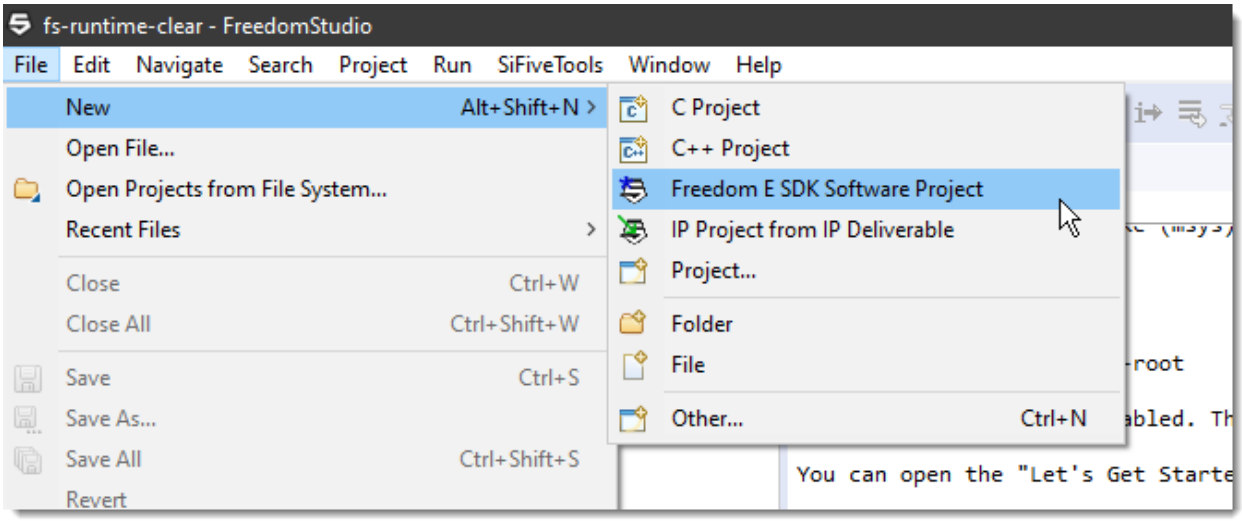
From an IP Project

If you have created an IP Project you can simply right-click on the project folder and select "Create a new project using this SDK"



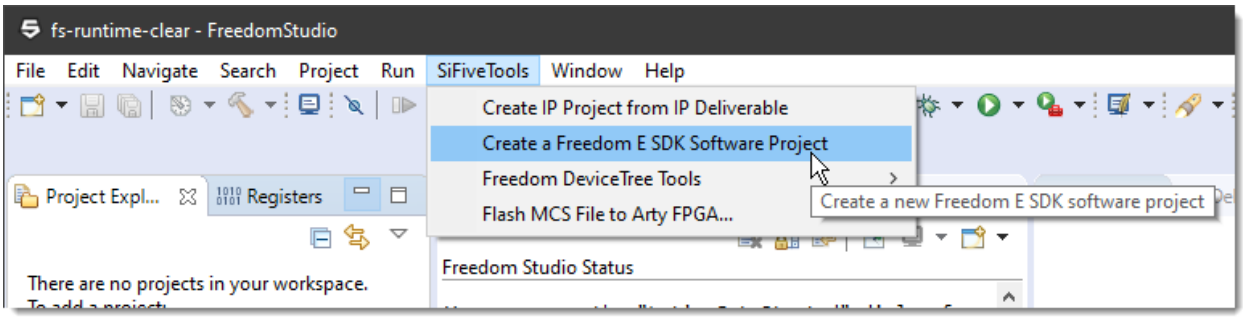
From the main menu

Select File → New → Freedom E SDK Software Project, as shown below:



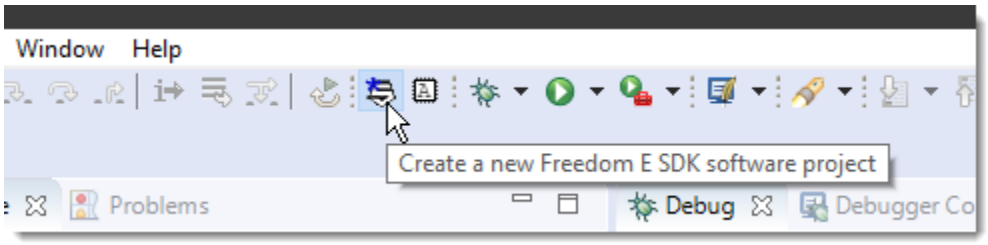
From the SiFiveTools menu

Select **Create a Freedom E SDK Software Project**:



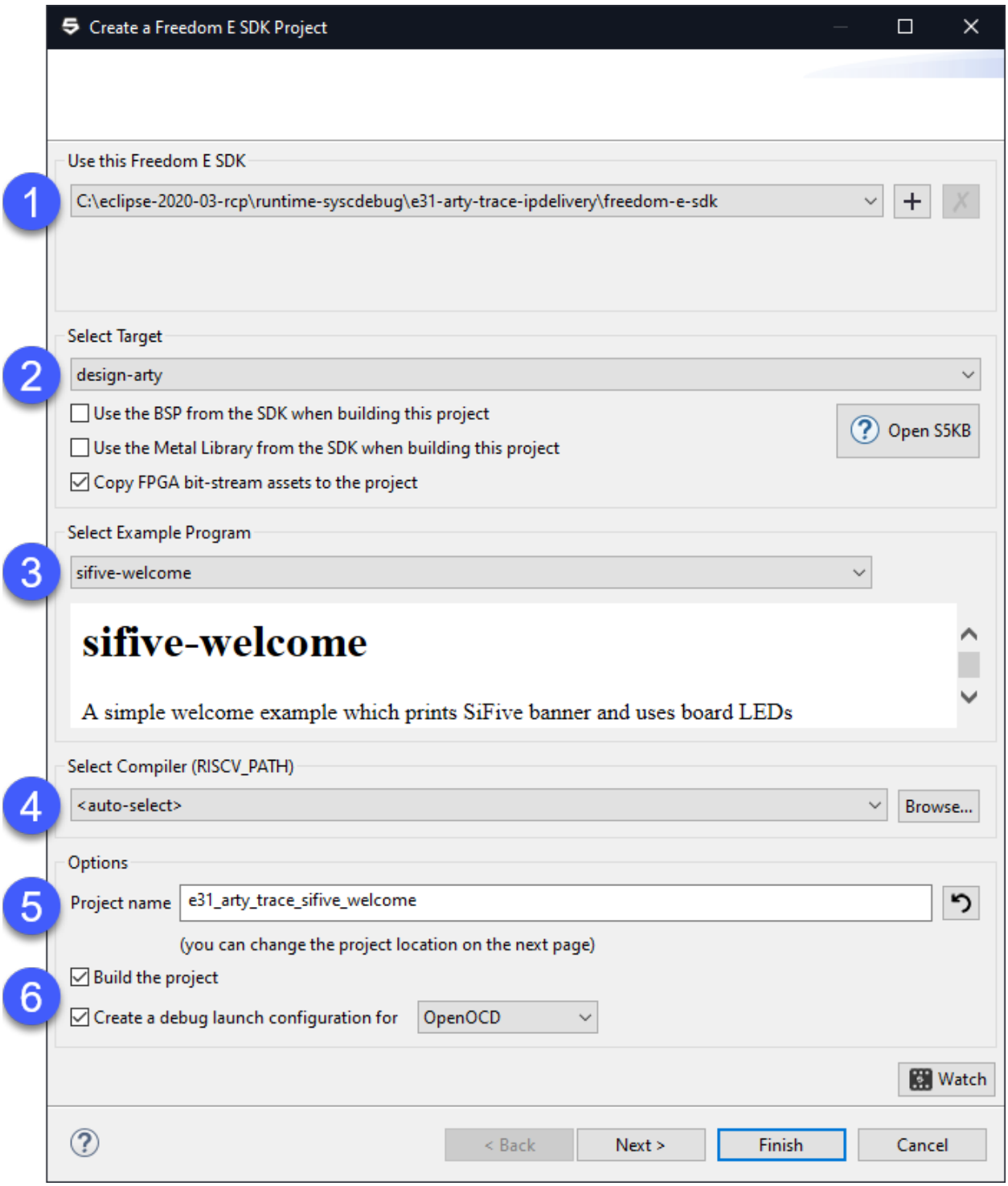
From the main toolbar

Click the “Create a New Freedom E SDK Software Project” icon, as shown:



The New Project Wizard

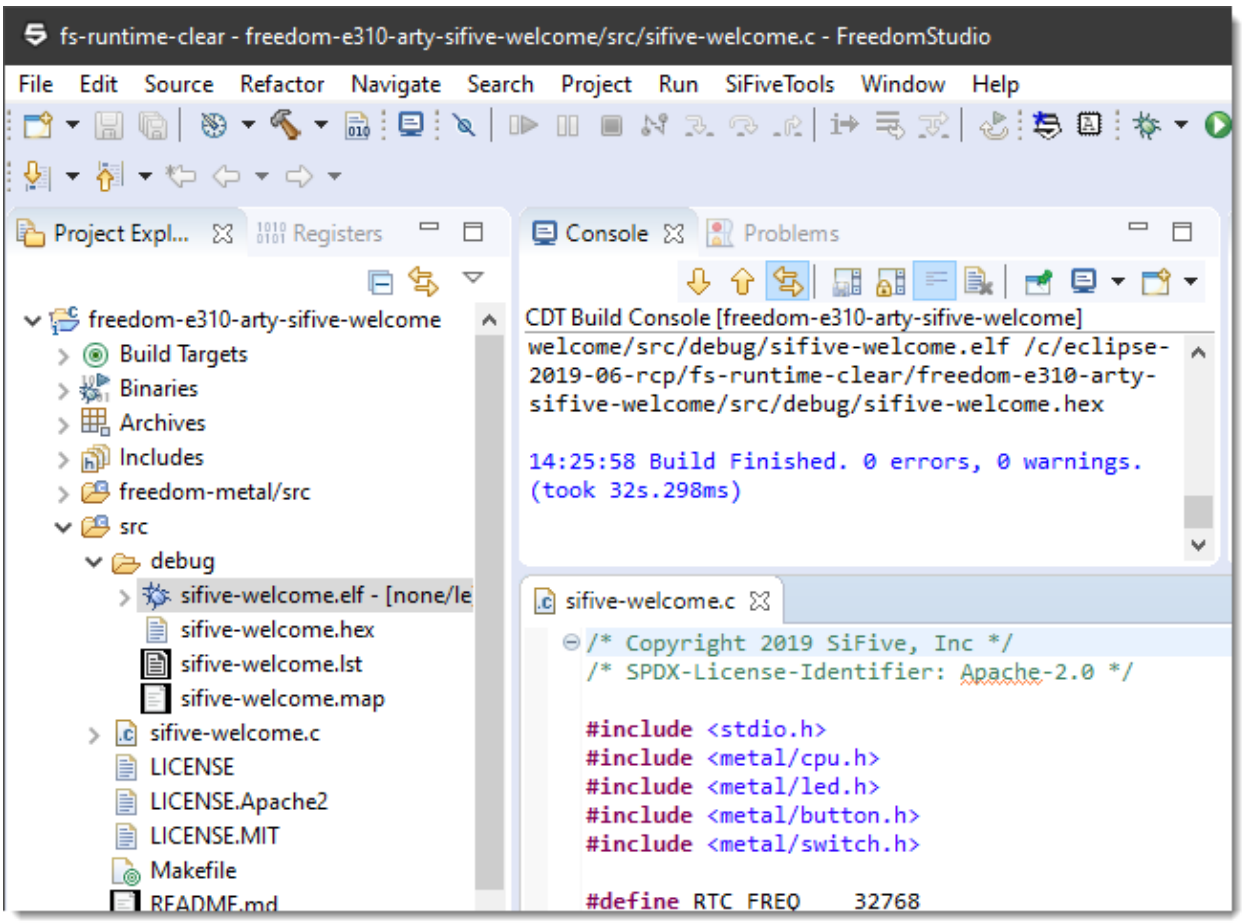
Selecting any of these will open the Freedom E SDK New Project Wizard. The first page of this wizard is shown below:



1. Starting in Freedom Studio 2019.08 you can easily work with multiple SDK instances. You can select from any SDK instance on your host computer and create software projects from the selected SDK. The drop-down box is automatically populated with any SDK instances found in your workspace projects. Using the '+' button you can also select an SDK instance that is not contained in your workspace.
2. When you first open the wizard the target selection box might be empty. You need to select a target from the options in the drop-down. You must select the target that matches your core and target platform of choice. You also have the option to copy the FPGA assets into the new project. This is strictly optional and just makes it easier to share projects.
3. Select an example program. Several examples are provided and each one demonstrates different features sets of the core.
4. Select the compiler you want to use for this project. Choosing "auto-select" will maintain legacy behavior and automatically choose the newest compiler from all "discovered" compilers. Or use the dropdown selector to force a specific version. Or use the Browse… button to choose a compiler that Freedom Studio has not discovered.
5. The project name is automatically generated based on your target and example selections. If you do not like the generated name you can change it.
6. Finally, you can choose to automatically build and create a debug launch configuration for your new project. Select the type of launch as determined by your debugger probe. Choose "OpenOCD" if you are using an Olimex probe, and "JLink" if you are using a JLink probe or a target with a built-in JLink OB device, and "QEMU" if you are using one of the QEMU targets. Selecting certain targets will automatically select the best option for that target. You can only create a debug launch configuration if you build the project first.

That is really all there is to creating a new Freedom E SDK project. If you are satisfied with your choices click the **Finish** button. If you would like to change the project location, click the **Next** button, and give your project a new location on the next page. *We recommend using the default location, which is a folder in your workspace folder*.

When you click the **Finish** button, Freedom Studio will create your new project and build it. When the build is complete Freedom Studio will reveal the built ELF file in the project explorer and open the main source file, as shown:



If you checked the "Create a debug launch configuration" checkbox when creating your project, the Debug Launch Configuration Dialog will automatically open after the ELF file is built and revealed. *This may take a few seconds to happen.*

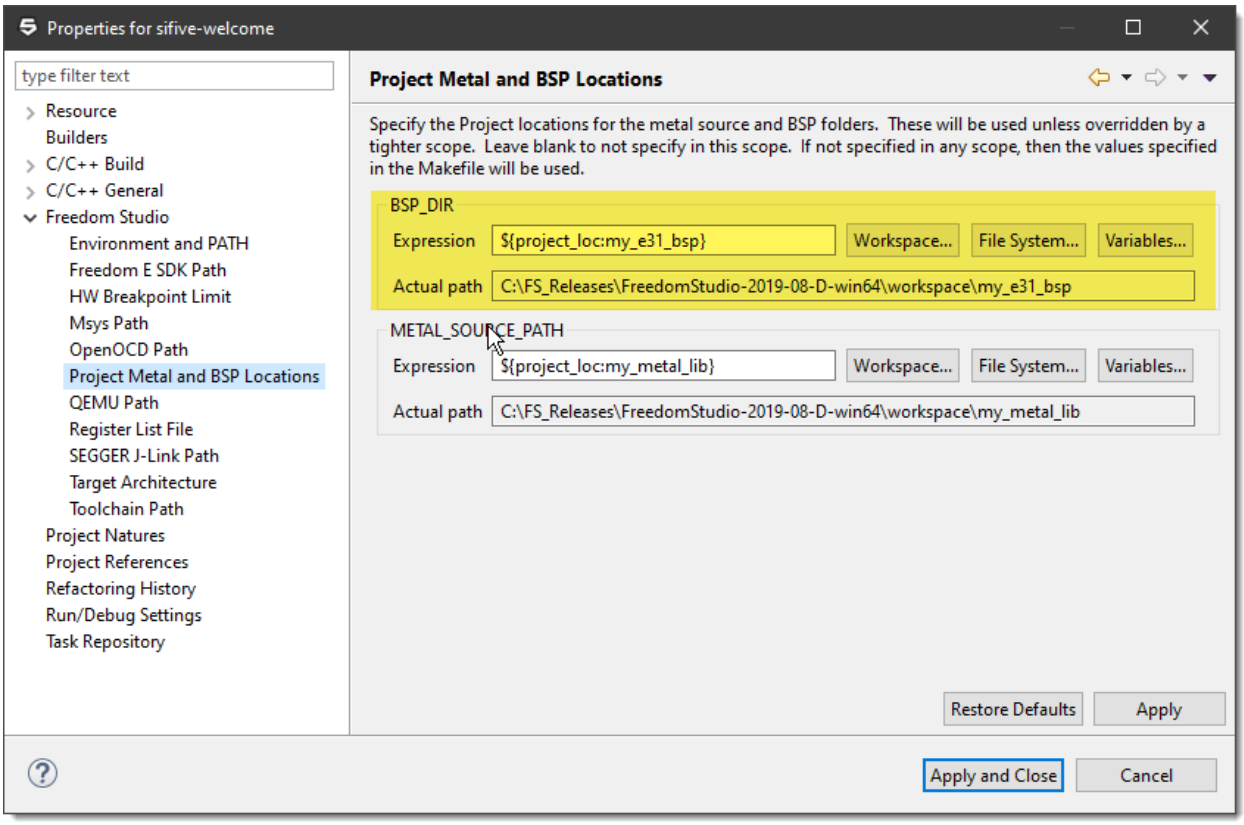
Share BSP with Multiple Projects

This is an advanced use scenario and is entirely optional. We recommend not doing this until you are more familiar with the SDK.

Prior to Freedom Studio 2019.08 each freedom-e-sdk based project had to have its own copy of the BSP. Changes in one copy had to be manually propagated to other copies.

You can now share a BSP with multiple projects. The BSP can be located in your workspace (as a separate project, or as part of a software project), or anywhere on the host file system. You can specify a BSP location via the Global Preferences, Workspace Preferences, or Project Properties.

For example, the Project Properties dialog shown here specifies that the BSP for this project should be pulled from the "my_e31_bsp" project in the Workspace.


Properties for sifive-welcome
type filter text
Resource
Builders
C/C++ Build
C/C++ General
Freedom Studio
Environment and PATH
Freedom E SDK Path
HW Breakpoint Limit
Msys Path
OpenOCD Path
Project Metal and BSP Locations
QEMU Path
Register List File
SEGGER J-Link Path
Target Architecture
Toolchain Path
Project Natures
Project References
Refactoring History
Run/Debug Settings
Task Repository
Project Metal and BSP Locations
Specify the Project locations for the metal source and BSP folders. These will be used unless overridden by a tighter scope. Leave blank to not specify in this scope. If not specified in any scope, then the values specified in the Makefile will be used.
BSP_DIR
Expression
${project_loc:my_e31_bsp}
Workspace...
File System...
Variables...
Actual path
C:\FS_Releases\FreedomStudio-2019-08-D-win64\workspace\my_e31_bsp
METAL_SOURCE_PATH
Expression
${project_loc:my_metal_lib}
Workspace...
File System...
Variables...
Actual path
C:\FS_Releases\FreedomStudio-2019-08-D-win64\workspace\my_metal_lib
Restore Defaults
Apply
Apply and Close
Cancel

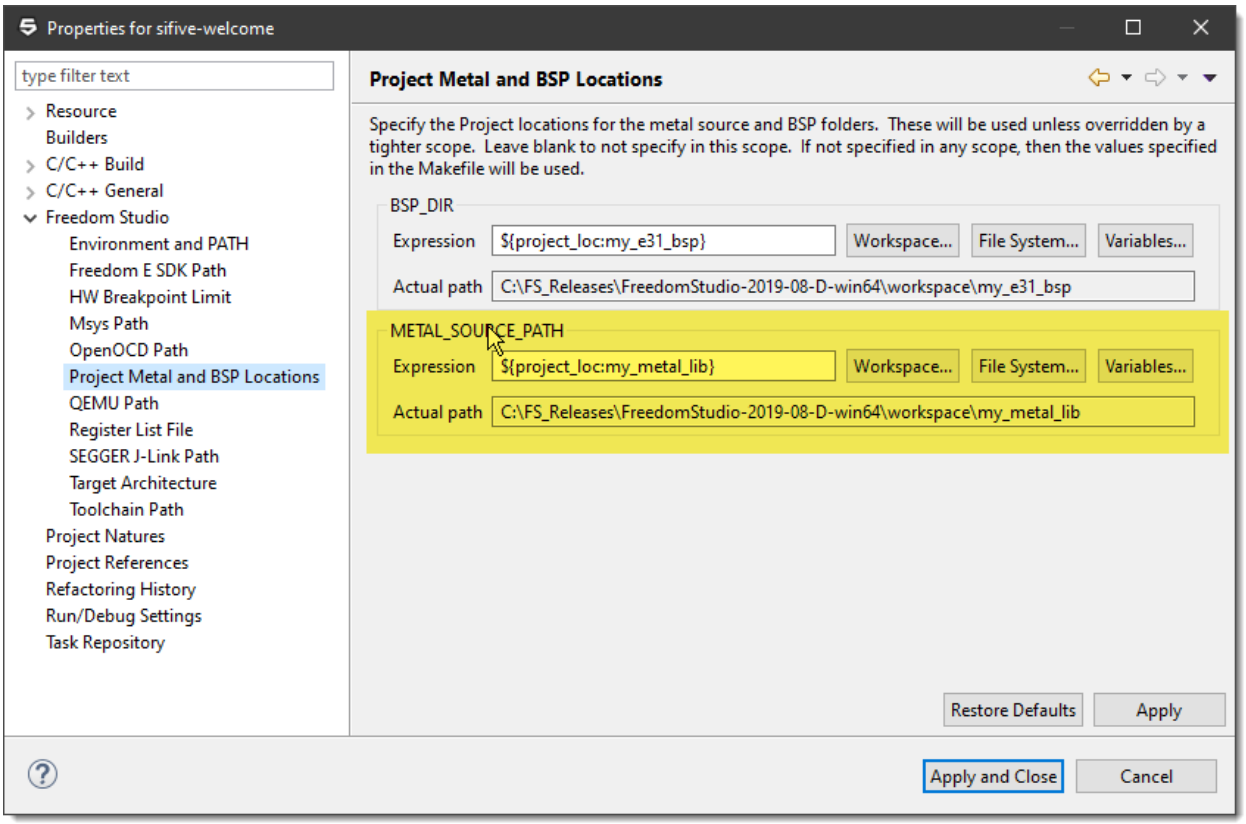
Share Metal Library with Multiple Projects

This is an advanced use scenario and is entirely optional. We recommend not doing this until you are more familiar with the SDK.

Prior to Freedom Studio 2019.08 each freedom-e-sdk based project had to have its own copy of the metal library. Changes in one copy had to be manually propagated to other copies.

You can now share a metal library with multiple projects. The metal library can be located in your workspace (as a separate project, or as part of a software project), or anywhere on the host file system. You can specify a metal library location via the Global Preferences, Workspace Preferences, or Project Properties.

For example, the Project Properties dialog shown here specifies that the metal library for this project should be pulled from the "my_metal_lib" project in the Workspace.



Benchmark Examples Default to Release Configuration

When creating a new freedom-e-sdk project with 'coremark' or 'dhrystone' the project will default to the "release" configuration. All other example programs will default to the 'debug' configuration.

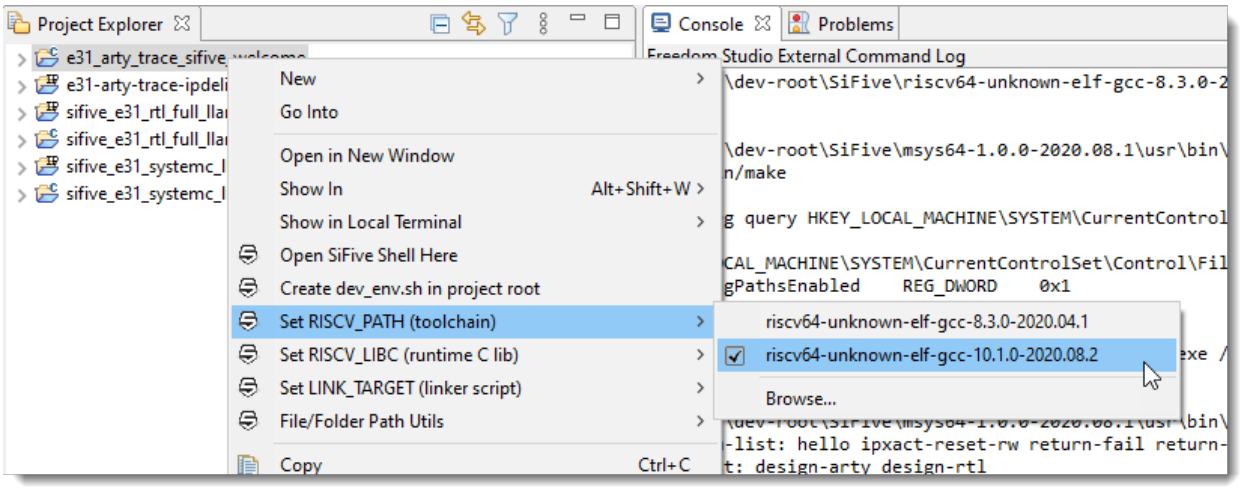
Defaulting to the 'release' configuration for benchmarks helps to ensure that: (1) accurate benchmark results are reported by default (the user does not have to remember to switch

to the ‘release’ configuration); (2) benchmarks will build successfully and fit into the available memory.

Configuring RISCV_PATH

Freedom Studio allow you to easily select from available toolchains.

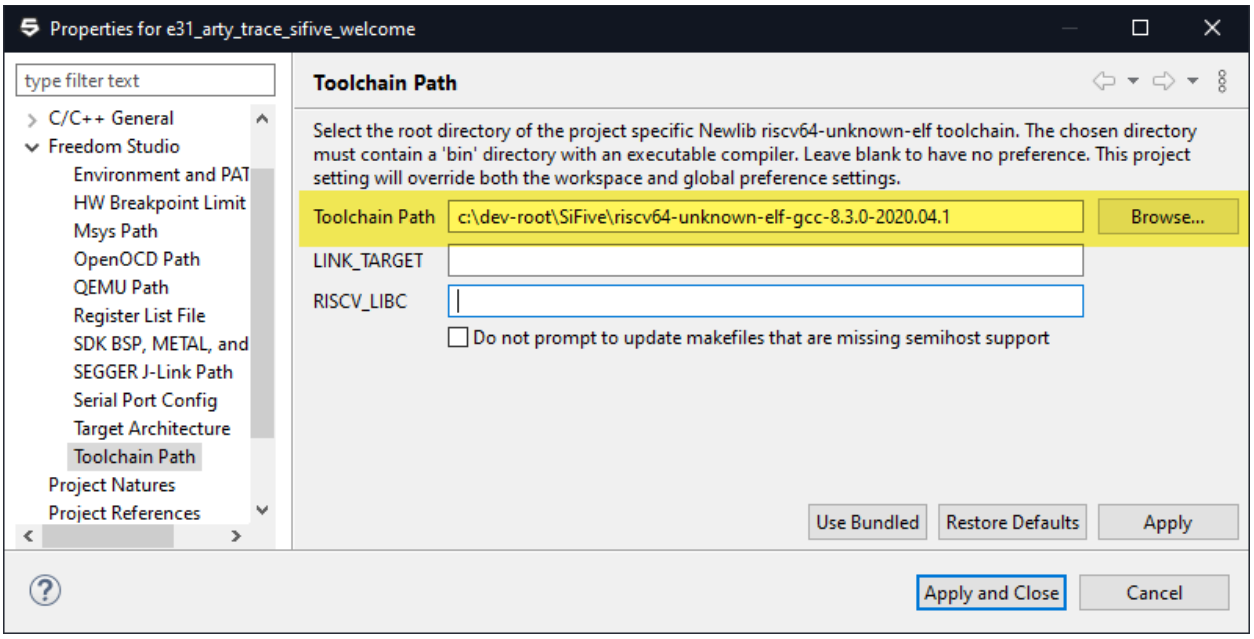
Open the project context menu, then open the Set RISCV_PATH submenu and select from the list of toolchains. The current selection will be checked.



The **[Use Makefile]** menu item simply instructs Freedom Studio not to override the RISCV_PATH environment variable when building the project. This causes the build to use whatever default value the Makefile specifies.

Any other selection will simply export the value to the RISCV_PATH environment variable for the build.

The RISCV_PATH setting is stored in the project properties and may be edited there as well.

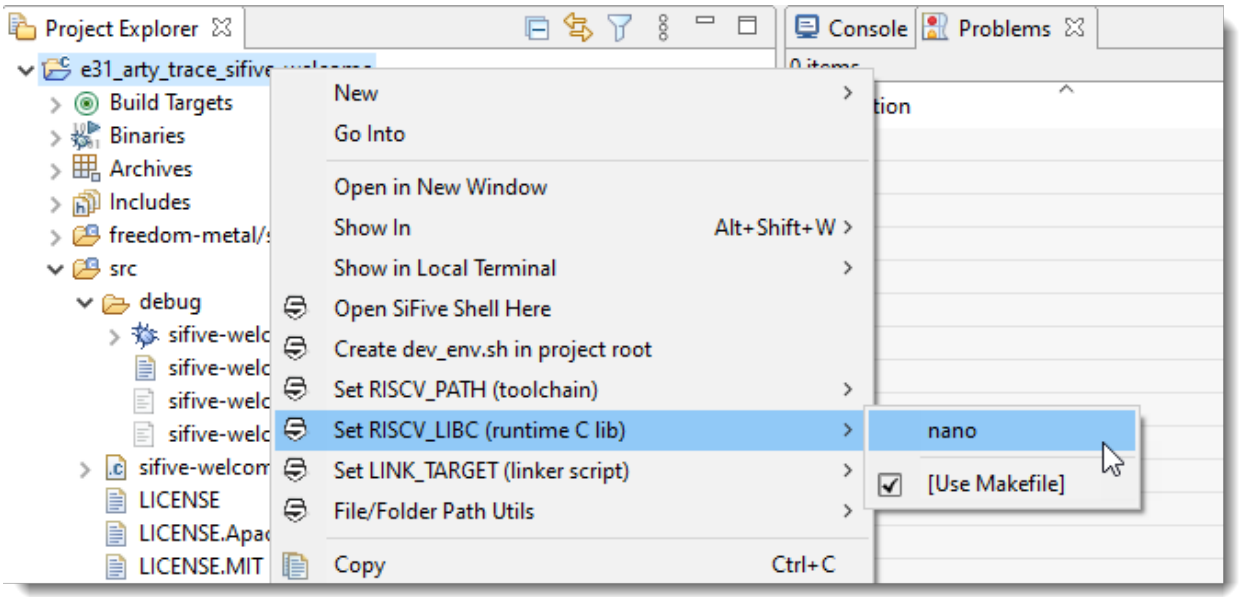


Whenever you make a selection from the RISCV_PATH menu Freedom Studio will offer to rebuild the project to use the new toolchain selection.

Configuring RISCV_LIBC

Freedom Studio allow you to easily select from available runtime libraries included in the toolchains for the project. The current selection will be checked.

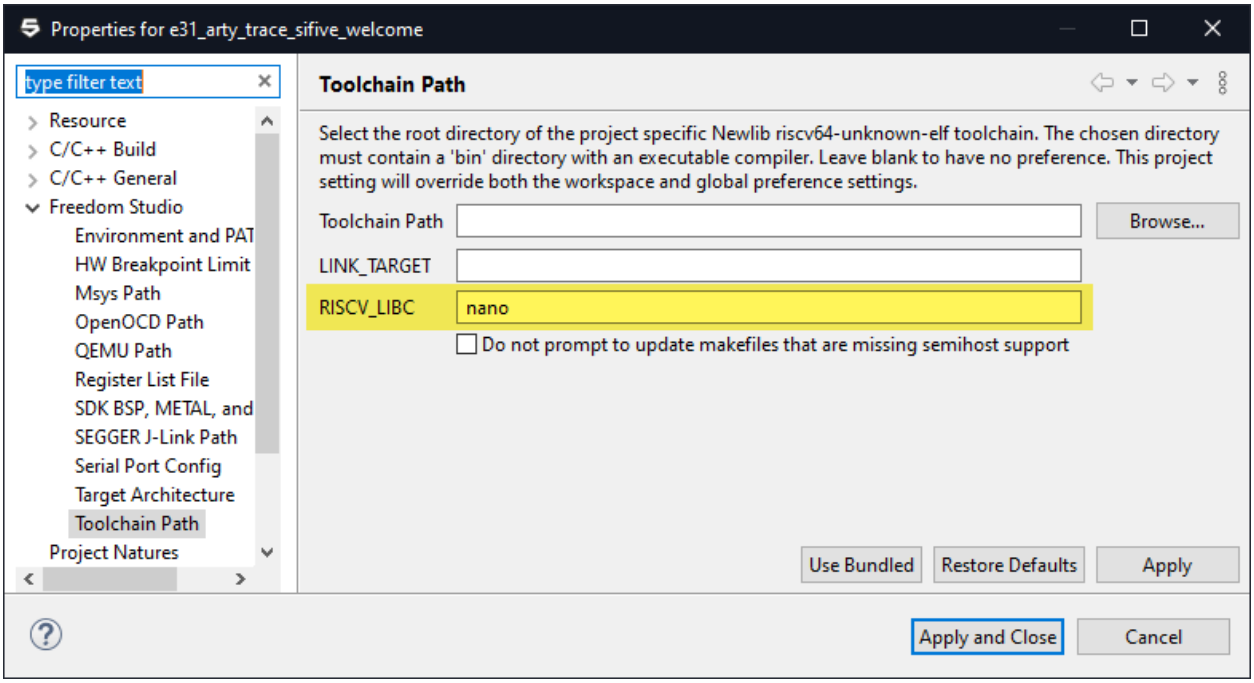
Open the project context menu, then open the Set RISCV_LIBC submenu and select from the listed runtime libraries.



The **[Use Makefile]** menu item simply instructs Freedom Studio not to override the RISCV_LIBC environment variable when building the project. This causes the build to use whatever default value the Makefile specifies.

Any other selection will simply export the value to the RISCV_LIBC environment variable for the build.

The RISCV_LIBC setting is stored in the project properties and may be edited there as well.

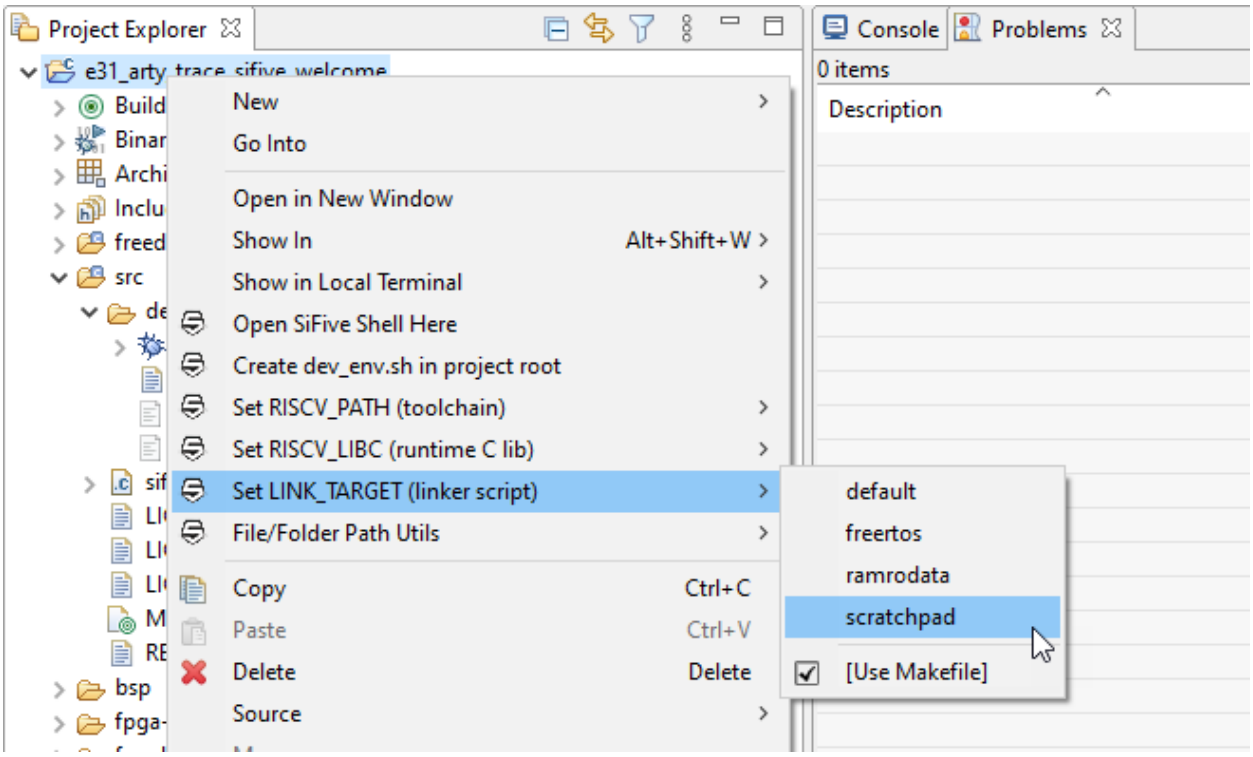


Whenever you make a selection from the RISCV_LIBC menu Freedom Studio will offer to rebuild the project to use the new runtime library.

Configuring LINK_TARGET

Freedom Studio allow you to easily select from available linker scripts (found in the BSP folder) for the project.

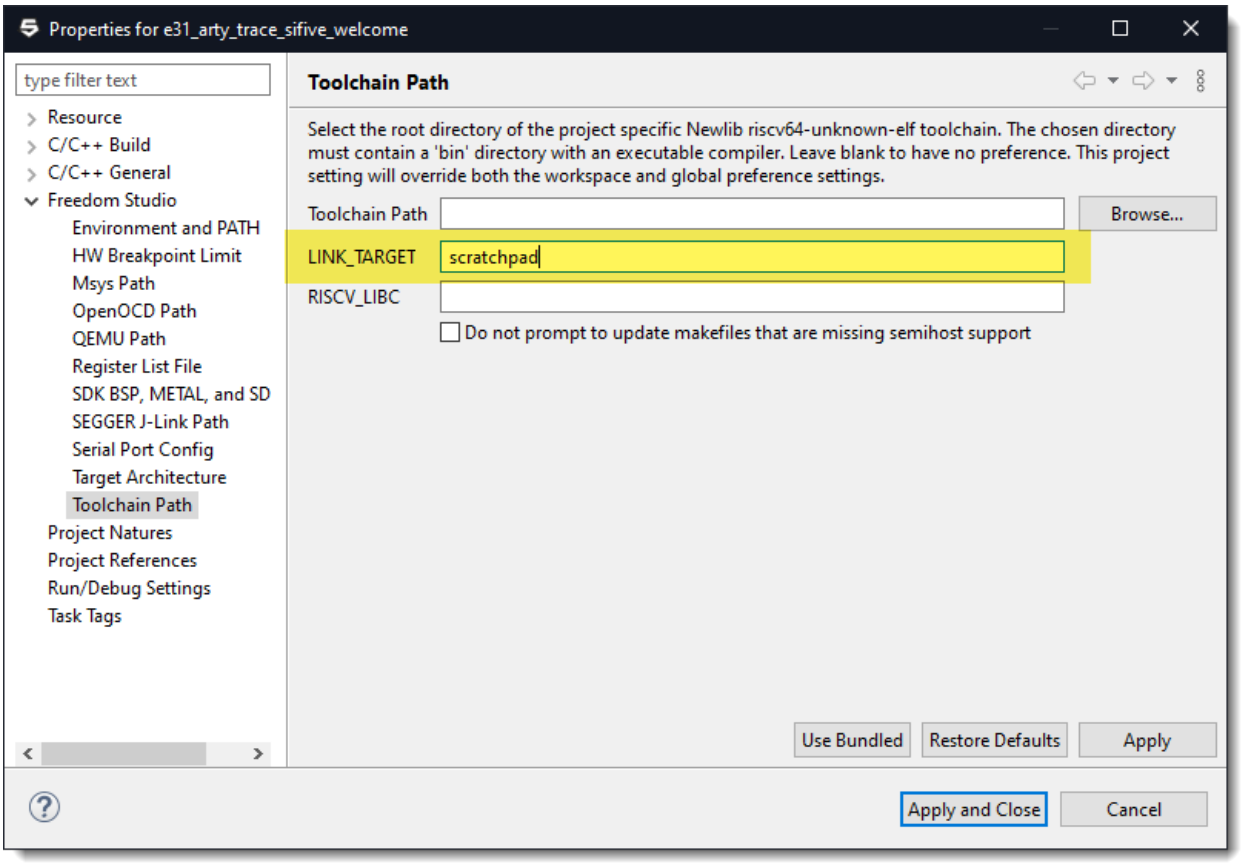
Open the project context menu, then open the Set LINK_TARGET submenu and select from the listed linker script names. The current selection will be checked.



The **[Use Makefile]** menu item simply instructs Freedom Studio not to override the LINK_TARGET environment variable when building the project. This causes the build to use whatever default value the Makefile specifies.

Any other selection will simply export the value to the LINK_TARGET environment variable for the build.

The LINK_TARGET setting is stored in the project properties and may be edited there as well.



Whenever you make a selection from the LINK_TARGET menu Freedom Studio will offer to rebuild the project to link with the new linker script.

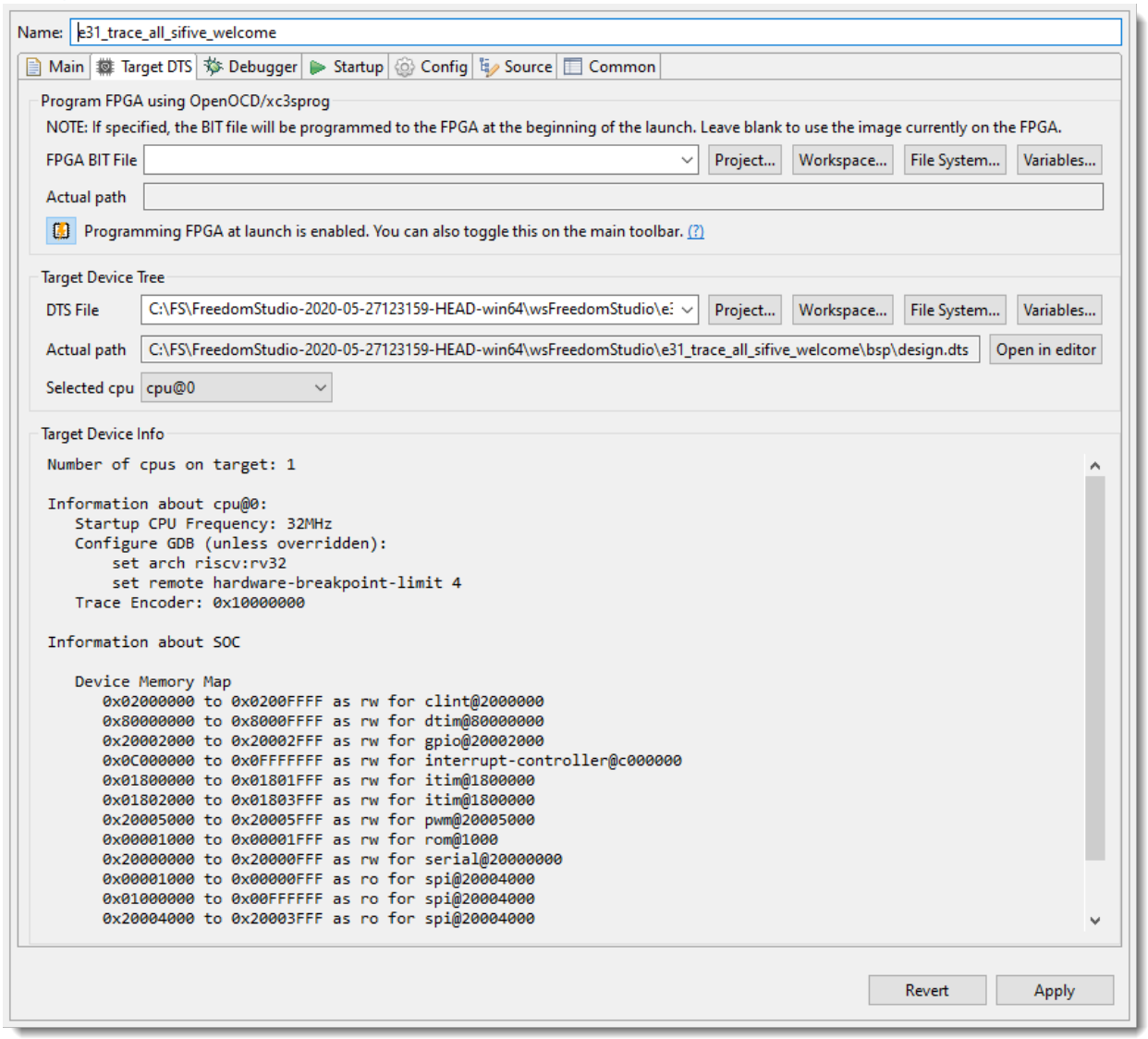
Debug Launch Configurations

Main Tab

Generally, there is no reason to adjust settings on this tab unless you are creating a new launch configuration from scratch.

See the [CDT Reference Documentation]

Target Tab



Debugger Tab

Connection Status [Windows Only]

On Windows Freedom Studio can automatically monitor the target connection and warn you if either target connection is missing. Uncheck the monitoring checkboxes if you are using a custom target connection.

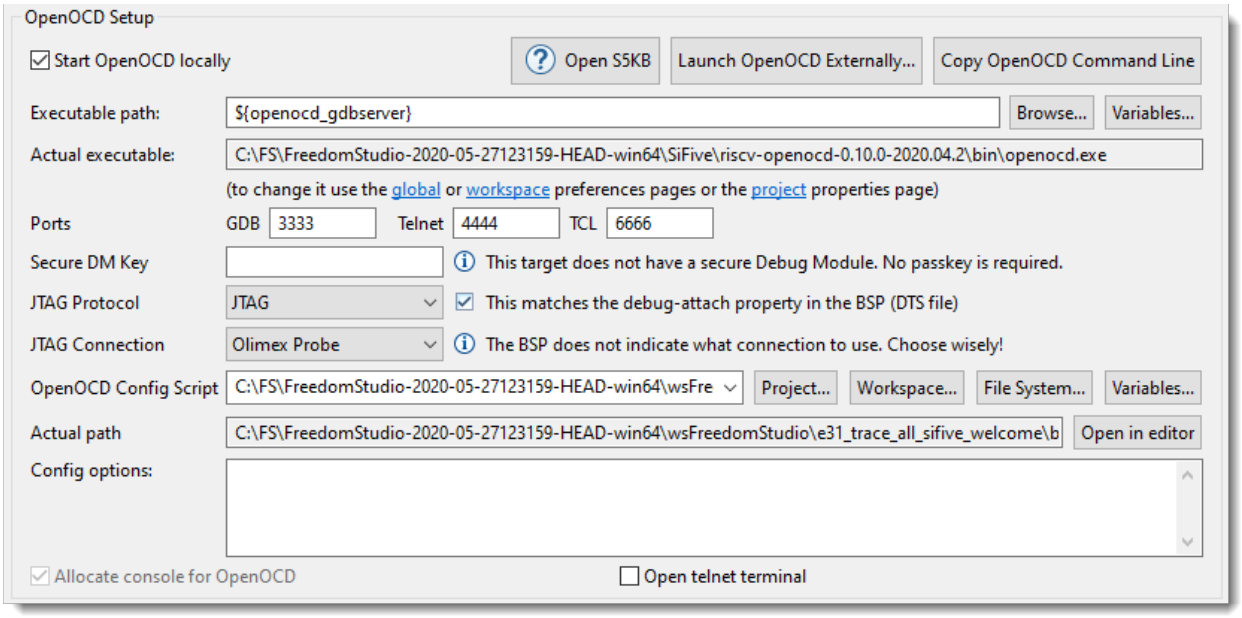
OpenOCD Setup

The default settings in this section should work.

See these Knowledge Base Articles for additional information:

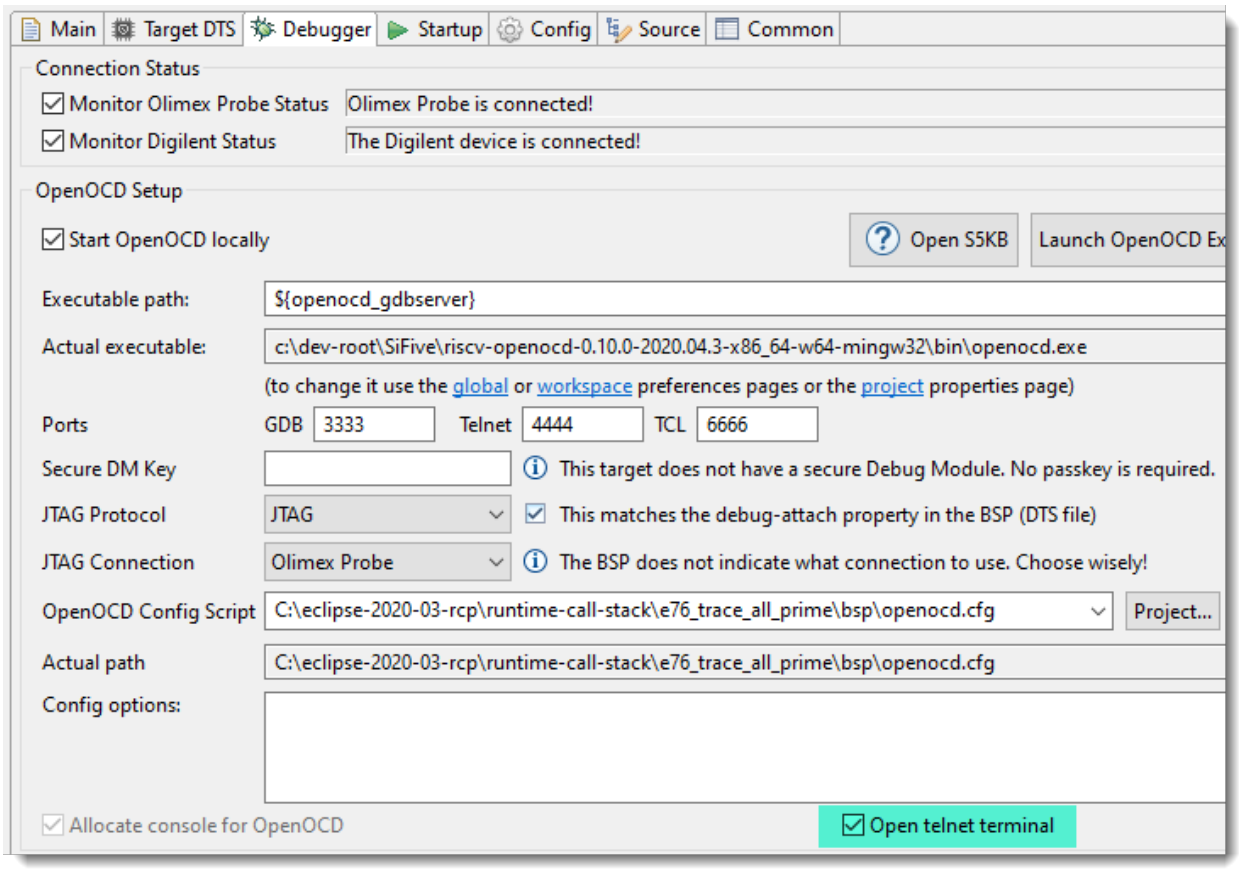
- Logging Open OCD Output in Freedom Studio
- OpenOCD in an IDE
- Connecting to an RTL Simulator using OpenOCD

Newer Knowledge Base articles may have been written since this manual was published. Click here to open the Knowledge Base and check.



Open Telnet Terminal

When using the OpenOCD debug connector, you can automatically open a telnet session to the OpenOCD TCL console.

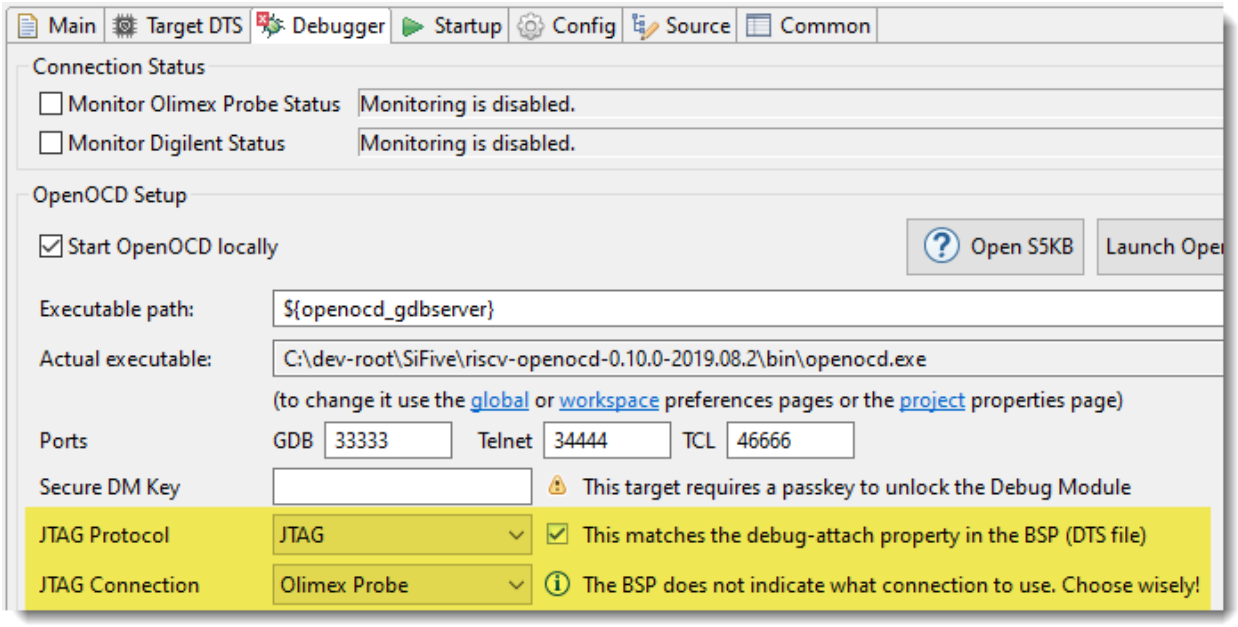


Specifying JTAG/cJTAG/BSCAN

Users no longer need to edit the openocd config script to enable cJTAG support. The launch configuration UI allows you to specify the type of connection (JTAG/cJTAG/BSCAN) and the default openocd script adapts accordingly.

This feature is only supported with IP packages and freedom-e-sdk instances newer than 2019.08. Older releases still require manual configuration of the openocd configuration script.

The Debugger Tab in the Launch Configuration Dialog is updated to more easily specify the debugger protocol and connection:

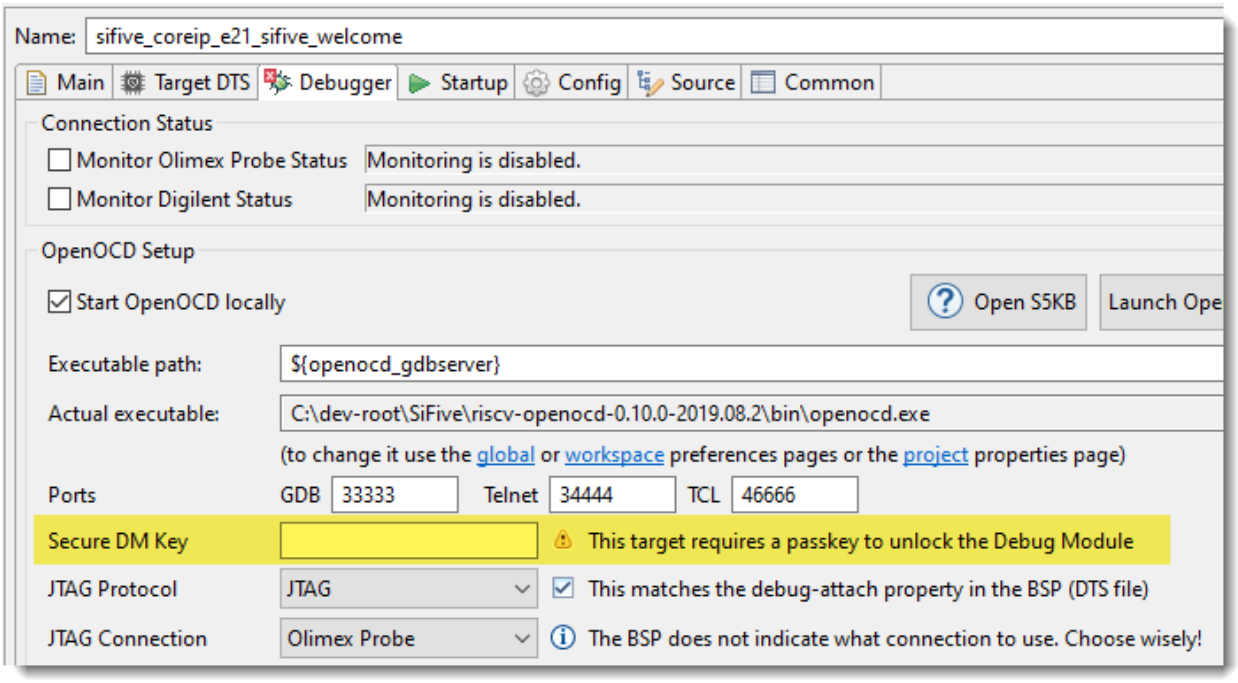


Newer IP packages (post 19.08) include information about the JTAG protocol implemented on the target. When this information is included Freedom Studio will automatically select the correct protocol. When the IP package does not contain this information, you need to ensure that the correct protocol is selected.

IP packages do not yet specify the implemented JTAG connection. You need to choose the correct connection.

Secure DM Key

Freedom Studio allows for specifying a key to unlock the debug module on cores that have a secure debug module.



The secure DM key is an 8 digit hexadecimal number (without a leading '0x'). The DTS file is examined to determine if a secure DM is present and Freedom Studio will print an appropriate message. If the DTS file is not provided you will need to know if a key needs to be provided. *For all SiFive generated FPGA designs, the key is hardcoded to 12345678*.

Auto Loading TCL Scripts

Sometimes it can be useful to have TCL scripts loaded into the OpenOCD TCL interpreter. While you can do this interactively using a telnet session to OpenOCD, doing so on every launch becomes quickly tiresome.

Freedom Studio can automatically load TCL scripts located in certain special folders. Before placing a TCL script into an autoload location make sure that the script does not have any syntax errors that would prevent the script from loading. Errors in the script will cause the debug launch to fail.

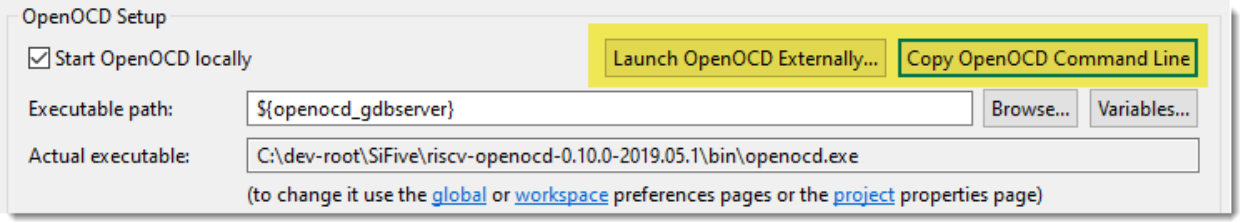
The following locations are examined for files ending in "*.tcl*". Found files will be loaded.

- `<project-folder>/scripts/tcl/autoload`
  Use this location to capture scripts that need to follow a project.
- `<openocd-root>/share/openocd/scripts/autoload`
  Use this location to capture scripts that need to follow OpenOCD
- `<fs-install-root>/SiFive/scripts/tcl/autoload`
  Use this location for scripts that may be needed for all projects.

- `FS_AUTOLOAD_TCL_FOLDER`
  Define this environment variable to point to any filesystem location. Any scripts found in this folder will be loaded.

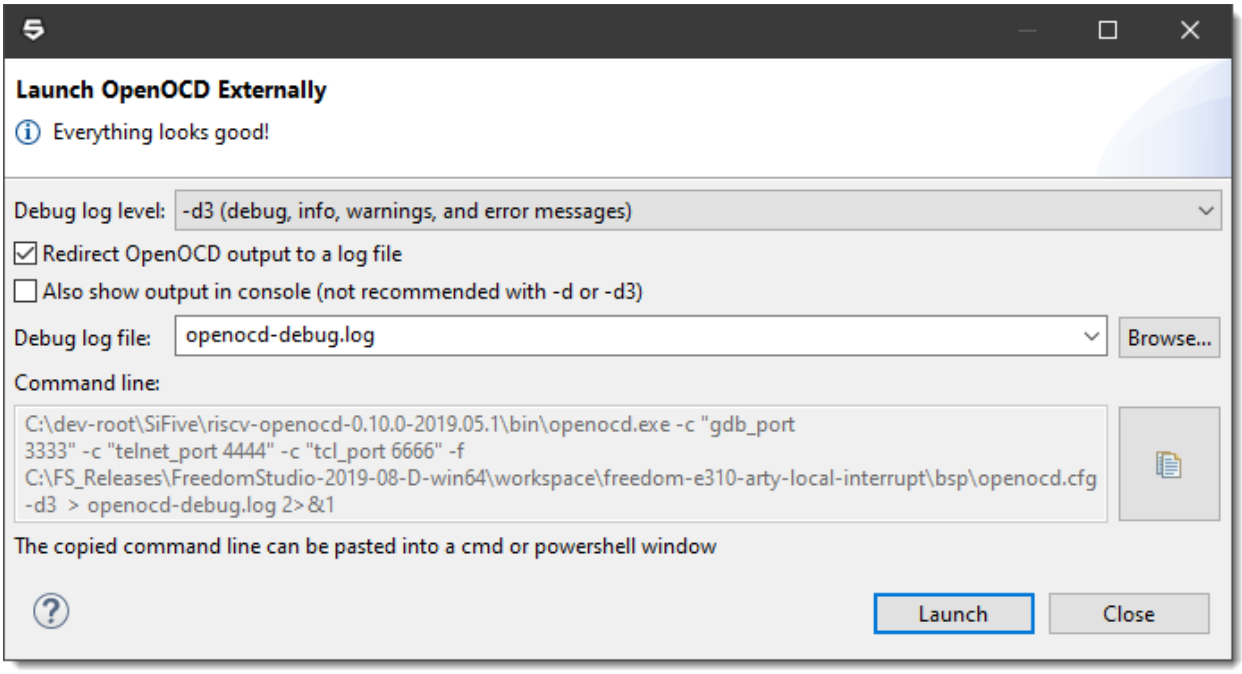
Launch OpenOCD Externally

Sometimes you need to launch OpenOCD as an external process. This is most useful when you need to capture an OpenOCD debug log to a file. Freedom Studio now has helper features to make this process simpler. The OpenOCD launch configuration dialog gains two new buttons that make it very easy to launch OpenOCD as an external process (i.e. a process not managed by Freedom Studio).



The "Copy OpenOCD Command Line" button copies the exact command line that Freedom Studio will use when launching to the system clipboard. You can paste (and edit, if desired) this command line in a shell (cmd prompt or powershell in Windows, or a terminal shell in Linux and MacOS).

The "Launch OpenOCD Externally..." button opens a new dialog box where you can configure the OpenOCD process with a custom debug level and optionally redirect the output to a file.



The “Debug log level” combo box lets you select a custom log level for the session. When preparing a debug log to send to support@sifive please use “-d3”.

If the “Redirect OpenOCD output to a log file” checkbox is not checked, OpenOCD log output only goes to the console. Checking this box will output the log to the specified file.

If you check the “Also show output in the console” checkbox then the out will go to both the console and the log file. It is recommended that you not check this box when using the “-d3” log level.

The “Debug log file” specifies where to create the log file. If a relative path is specified, then the path is relative to the project directory.

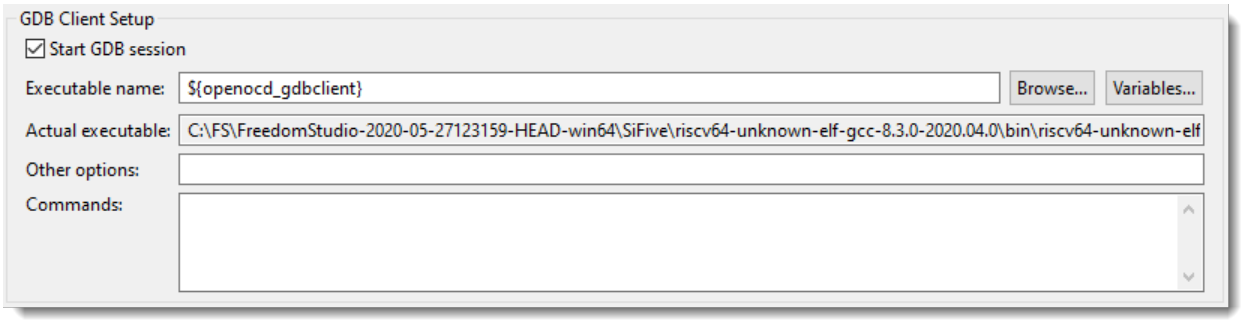
You can use the “Copy” button (to the right of the command line box) to copy the command line to the clipboard.

When launched as an external (unmanaged) process it is your responsibility to terminate the OpenOCD process when it is no longer needed. Note that you do not have to terminate the OpenOCD process and restart it between successive debug launches. Freedom Studio will happily use the running process multiple times.

Windows Only: When the “Launch” button is pressed, a new Command Window is opened and the OpenOCD process is started and the “Start OpenOCD locally” checkbox is automatically unchecked. [Linux and MacOS do not show the “Launch” button, but the command line can still be copied and pasted into a terminal shell. Be sure to uncheck the “Start OpenOCD locally” checkbox.]

GDB Client Setup

The default values in this section should cover almost all use cases. You should not have to change anything here except in special circumstances.

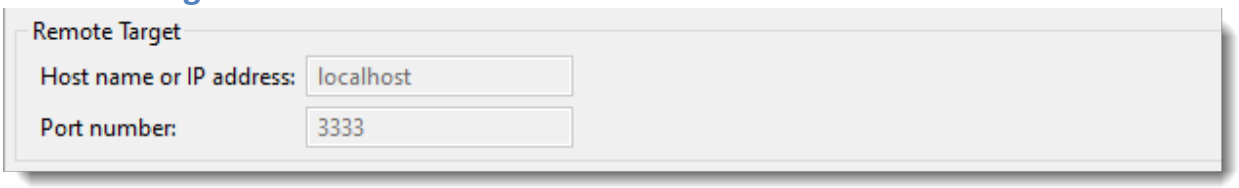


- **Start GDB Session**
  When checked, Freedom Studio will start the gdb executable. We recommend leaving this box checked.
- **Executable name**
  The default variable shown will automatically use the gdb exec bundled with

Freedom Studio. If you need to use a different gdb executable, use the Browse button to locate and select it.

- **Other Options**
  Additional command line options to be passed to the gdb client.
- **Commands**
  Additional command that will be sent to gdb upon startup. These commands are executed just before the ".gdbinit." sequence and can be used to set target state when needed. Commands prefixed with "monitor" will be sent to OpenOCD. You can content assist (Ctrl-Space) to get a list of common commands.

Remote Target



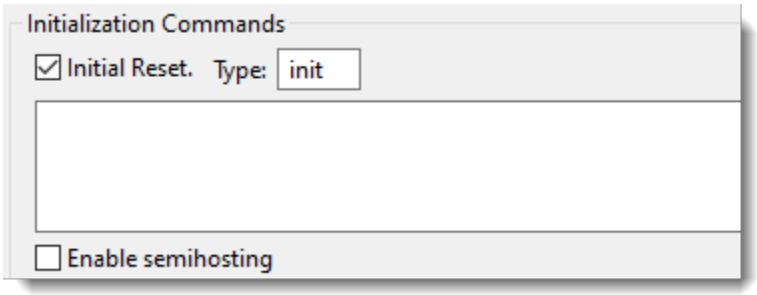
These settings are used when you want to connect to an external OpenOCD process. The external process can be running on the local machine, or any other machine that is accessible from the local network.

Other Settings

- Force Thread List Update on Suspend
  Force all threads to be updated on every suspend. Usually not required.

Startup Tab

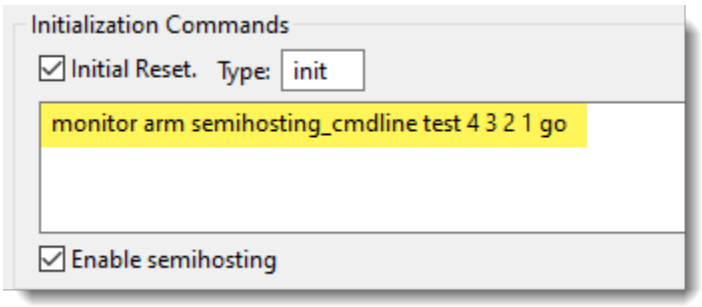
Initialization Commands



- **Initial Reset**
  Perform an initial reset and halt; this will take the processor out of whatever state it was and prepare it for programming the flash. Normally the GDB server performs a reset when starting, so this is especially useful when a specific reset type is required. It is disabled when the 'Connect to running' option is used. The generated command is 'monitor reset <type>'.

- **Reset Type**
  Can be one of:
  - 'run' Let the target run,
  - 'halt' Immediately halt the target,
  - 'init' Immediately halt the target and execute the reset-init script.
- **Initialization Commands**
  Additional or alternate initialization commands. To reach the GDB server, the commands should be prefixed with 'monitor'.
- **Enable semihosting**
  Enable support for semihosting. The generated command is 'monitor arm semihosting enable'.

To provide argument to main, add the following command with the desired arguments specified:

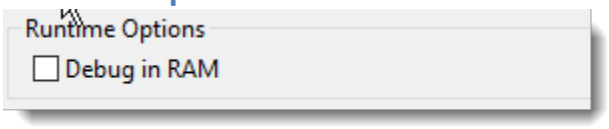


This example will result in

```
argc == 6

argv = ["test", "4", "3", "2", "1", "go"]
```

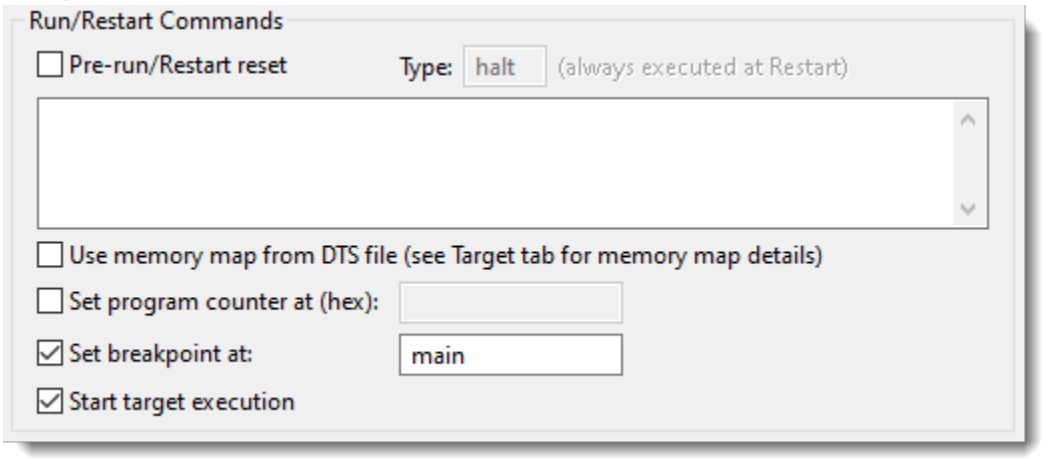
Load Symbols and Executables

Runtime Options



When checked, load the executable and run the entire debug session in RAM. The main effect is that the executable will be loaded after each reset/restart, not only once after the initial reset.

Run/Restart Commands



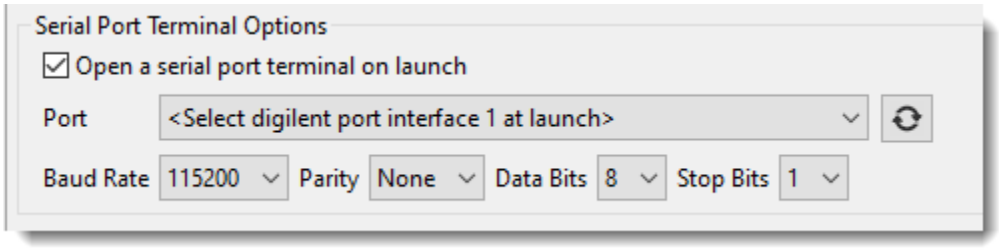
- **Pre-run/Restart reset**
  We recommend that this be left unchecked unless you have a specific need for a second reset. In general freedom-e-sdk software example projects do not require this. When check and additional `monitor reset 'type'` is issued. Doing so may affect the target state that has been previously set up.
- **Run/Restart Commands**
  Add any additional command that should be run on a restart.
- **Use memory map from DTS file**
  When checked gdb will be configured with the memory map extracted from the DTS file. Only use this option if your target cannot handle potentially memory accesses outside of existing memory.
- **Set program counter at (hex)**
  If you need the PC set to a specific location that is not indicated in the ELF file, specify that location here. Otherwise the start address in the ELF file will be used.
- **Set breakpoint at**
  The default value is "main". Sets an initial (temporary) breakpoint at this location. Common values are "main" (for debugging application code) and "_enter" (for debugging startup code)
- **Start target execution**
  When checked Freedom Studio will start target execution. Otherwise the target will remain halted at the first instruction.

Initial Trace Setup

See Initial Trace Setup

Serial Port Terminal Options

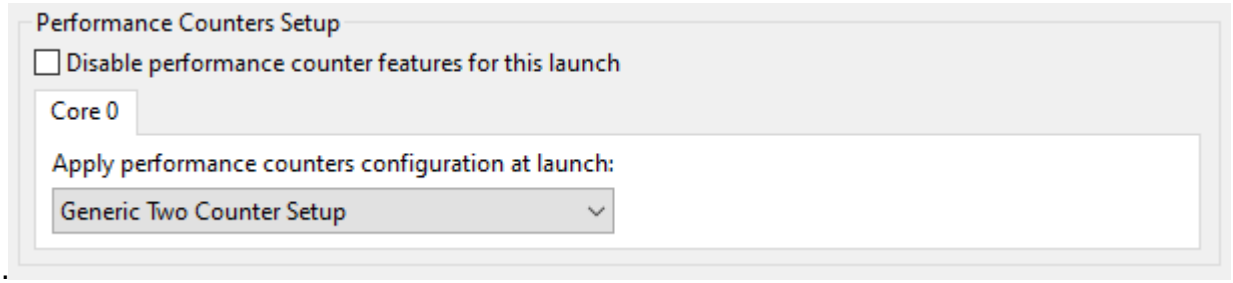
Launch Configurations can automatically open a serial console when the launch starts. Use the following controls to specify the serial port parameters.



In most cases Freedom Studio can automatically determine the correct serial port to use. If you find that the correct port is not being used, use the drop-down to select the correct port. See also: UART List View

Performance Counter Setup

This section allows for specifying the initial configuration for performance counters at launch



.

You may disable the performance counter feature by checking the checkbox. It is recommended that you check this box if you are not actively using performance counters while debugging the target, especially on targets that have many performance counters. Management and discovery of performance counters can significantly slow down the launch process.

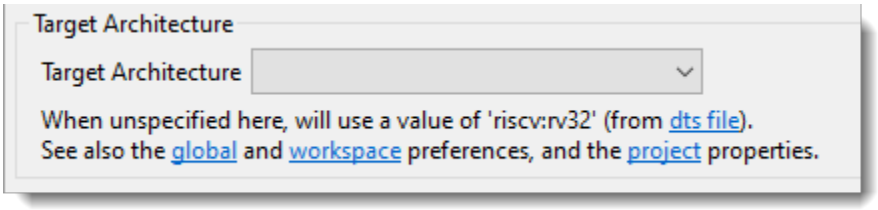
See Performance Counter Setup

Config Tab

Register List
See Register List Management

Hardware Breakpoints
See Managing Hardware Breakpoint Resources

Target Architecture
The target architecture is usually extracted from the DTS file in the BSP folder. If, for some reason, you are not using a BSP, the setting can be manually specified in Preferences or directly in the debug launch configuration.

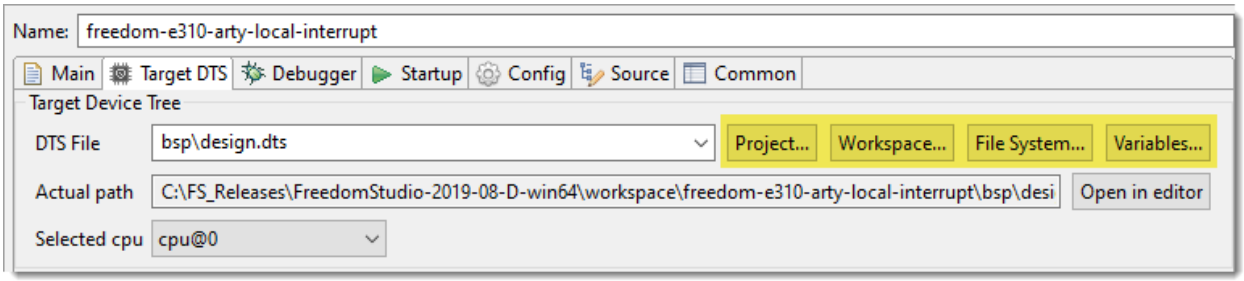


Source Tab
See the CDT Reference Documentation

Common Tab
See the CDT Reference Documentation

Automatic Removal of Temporary Breakpoints
At the start of a debug launch any existing temporary breakpoints are removed. This prevents temporary breakpoints from a previous launch or different project from accidentally interfering with a debug launch or inadvertently using precious hardware breakpoint resources.

Selecting File Resources
Freedom Studio uses a common UI for selecting file resources. You can select from Project, Workspace, or File System scopes, and use Eclipse variables to build expressions.

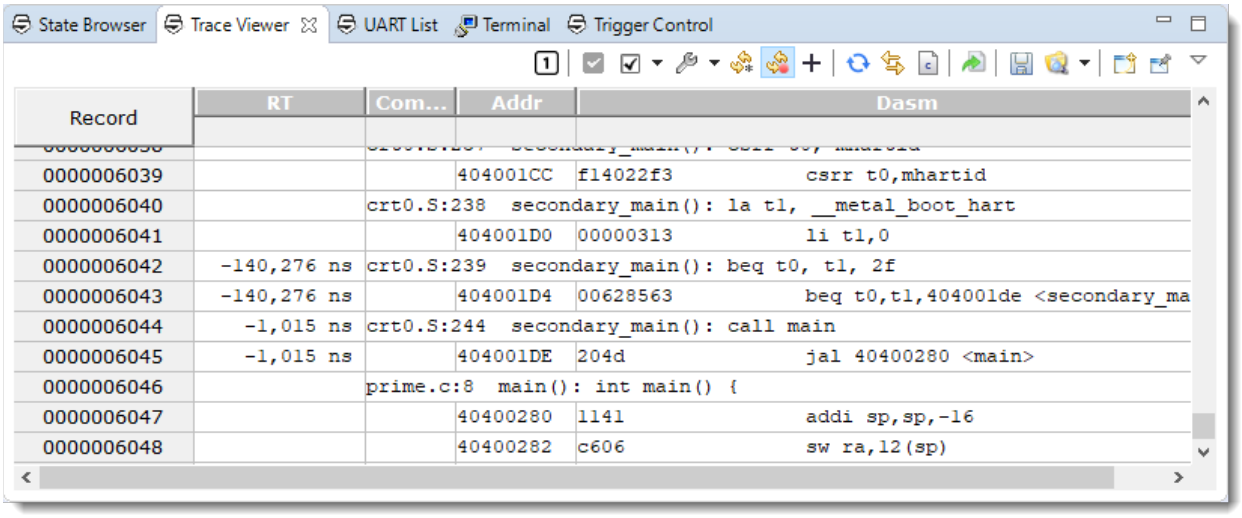


The “Open in Editor” button that will open the selected resource in a Freedom Studio editor window. You will need to close the Debug Language window to use the editor.

Processor Trace

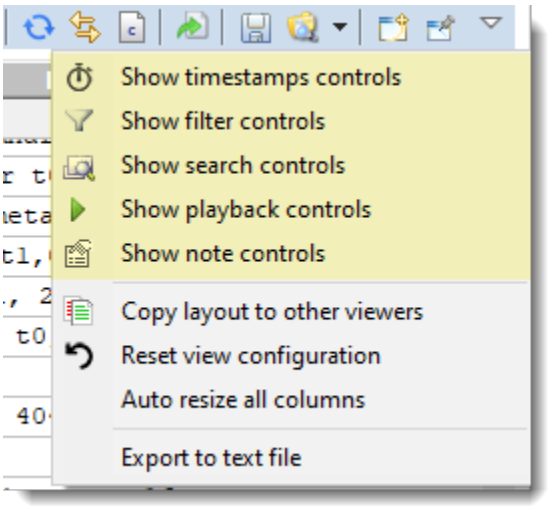
Trace Viewer

The Trace Viewer is the primary interface to the SiFive Trace system. This view shows trace data and allows for full control of the trace system. This is the Trace Viewer:



Trace Viewer Control Bar

That is a lot of buttons. The buttons on this bar control almost all aspects of the trace system. Trace control groups can be hidden when not being used, helping to reduce the clutter of unwanted controls. The view dropdown menu (on the far right) contains switches that control enablement of several command groups in the toolbar.



Each of the control groups is described below.

Primary Trace Control Commands

This control group cannot be hidden. It is the primary interface to the trace system.

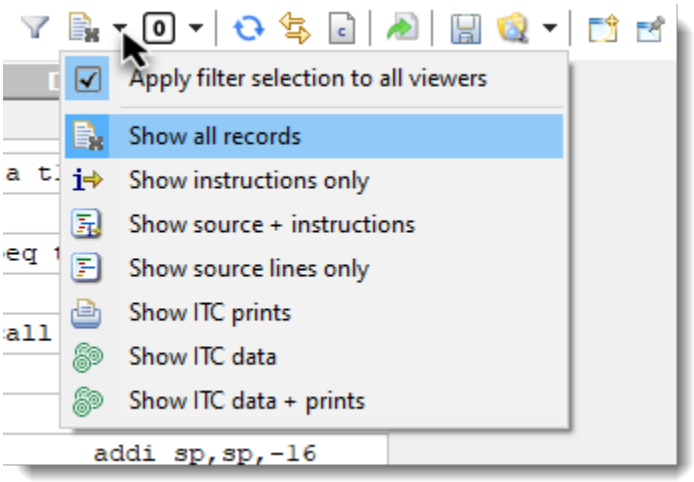
1. **Funnel Trace Enable**
   On multi-core systems with a trace funnel, this toggle button enables or disables the trace funnel. It is, in effect, a master switch for multi-core trace enable/disable. If this toggle is disabled, no trace from any core will be produced.
2. **Core trace enable**
   This toggle button enables or disables trace on a specified core. Clicking the checkbox will toggle the active core (that is the core currently selected in the Debug thread view). Using the drop-down button to the right of the checkbox allows you to toggle the enable state for other cores without having to select the core in the Debug view first.
3. **Trace Control**
   Pressing this button will open the Trace Control Dialog for the active core (that is the core currently selected in the Debug thread view). Using the drop-down button to the right of the checkbox allows you to configure trace for other cores without having to select the core in the Debug view first.
4. **Load trace data from target**
   Pressing this button will cause all trace data on the target to be loaded into Freedom Studio. This is a "manual" load. There are two primary uses cases:
   a. If you choose not to have trace loaded on each halt, this button allows you to load trace on demand.
   b. If you have a large trace buffer on the target, Freedom Studio will only load part of the trace buffer when halting (for performance reasons). Pressing this button will cause the entire buffer to be loaded.
5. **Load Trace on Halt**
   This toggle button controls loading trace when the target halts (either due to breakpoint, or user suspend request). When enabled, trace will be loaded at each halt. If you have more than 4KB of trace, only the first or last 4KB will be loaded on halt. [This size can be changed in Freedom Studio preferences.]
6. **Accumulate Mode**
   This toggle button controls how trace is accumulated when collected. When toggled off, the previous trace data is discarded before collection begins. [Internally, this

toggle controls when the trace write pointer gets reset.] When toggle on the trace buffer on the target will continue where it left off on the previous halt.

Filtering Commands

The filter command, and the filter row header in the trace data view table, control how trace data is filtered in the view.

1. **Toggle filter row visibility**
   This toggle button shows or hides the filter row in the Trace Data table. The filter row is located just below the column headers. When enabled, the filter row allows you to type regular expressions in the column filter to control which records are displayed.
2. **Predefined Filter Select**
   This dropdown menu allows you to select from several predefined filters.



Options include:

a. **Apply filter selection to all viewer**
   This toggle button, when checked, will apply any filter selection to all open Trace Data Views. When not checked, filter selection will only apply to the current view.
b. **Show all records**
   This is the default filter. It shows all trace records with no filtering applied.

c. **Show instructions only**
   Shows only instruction execution records
d. **Show source + instructions**
   Show instruction records with source lines interleaved. Each source line will be shown followed by all the instructions that make up the source line.
e. **Show source lines only**
   Shows only source lines. Instruction records are not displayed.
f. **Show ITC prints**
   Shows ITC print records only.
g. **Show ITC data**
   Show ITC data records only (ITC prints will not be displayed)
h. **Show ITC data + prints**
   Show both ITC data and ITC print records.

3. **Core Context Selector**
   Use this dropdown selector to select which core to display.

Timestamp Commands

The timestamp section controls how timestamp information is reported.

1. **Reset the timestamp counter on the target**
   Write 0 to the timestamp counter register on the target. Not required for general timestamping of trace. Provided for special cases, when needed.
2. **Set clock frequency for timestamps**
   Opens a dialog so that you can specify the clock frequency of the timestamp counter. This is required to translate timestamp values into real time. The timestamp clock can originate from different clock sources so it is important to know where the source clock is located and what frequency it is running.
3. **Timestamp unit select**
   This button cycles through different time units. You can also use the 't' key in the Trace View to cycle through the units.
   a. picoseconds
   b. nanoseconds
   c. microseconds
   d. millisecond
   e. seconds

f. clock cycles

4. **Set relative timestamp origin**
   Pressing this button with a timestamped trace record selected will set the relative timestamp origin to the selected record. All other timestamp values in the Relative Timestamp [RT] column will be reported relative to this record. You can also use the 'o' key to set the origin.

Search Commands

The search commands provide basic facilities for searching through trace data.

1. Search expression status
   This icon reflects the status of the search expression. If a mal-formed regular expression is entered, the tooltip for this icon will report the expression error. If the search expression is valid a green checkmark is displayed.
2. Search expression
   The search expression is a regular expression that will be used to find matches in the trace data. All displayed trace data is searched.
3. Search expression management
   This dropdown menu contains commands for saving and forgetting search expressions. A saved search expression will become available in the search term dropdown combo box.
4. Previous/Next search result
   Use the previous and next search result to locate the previous or next search result in the trace view. Search result records will be highlighted green for a short duration to indicate the result more clearly.

Sync Commands

The sync command controls how the Trace Data view synchronizes with other parts of Freedom Studio.

1. **Sync viewports**
   When toggled on all views looking at the same trace data set will be synchronized such that scrolling in one view will also cause other viewers to scroll so that the same trace record (or closest match) is selected and shown in all synchronized views.
2. **Toggle sync to editor**
   This toggle button controls synchronization to the editor and disassembly view. When toggled on, selecting a trace record in the trace data view will also show the selected source line in an editor and show the corresponding instruction in the disassembly view.
3. **Goto source**
   If "toggle sync to editor" is turned off, then this button will manually sync the selected record to the editor and disassembly view. You can also do this by double-clicking any trace record.

Note Commands

Notes are a simple way of marking and noting trace records in a trace capture. Notes are extremely ephemeral. Any notes created are lost on the next trace capture. Notes are intended for navigating around a trace capture in real time during a debug session. Using notes requires enabling the Note column in the Trace Data View. When enabled you can enter a simple note in the note column for a given trace record.

1. **Note selector**
   This is a dropdown menu that will list all notes created in this trace capture. Selecting a note in the menu will take you to the noted trace record.
2. **Goto previous note**
   Press this button to go to the previous noted record.

3. **Goto next note**
   Press this button to go to the next noted record.

Save Trace Commands

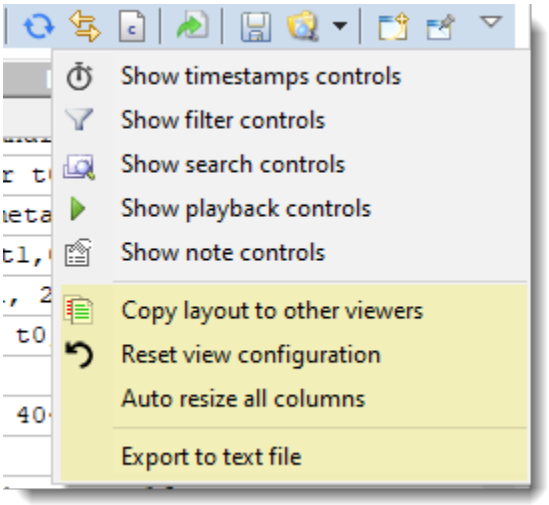
These commands provide the ability to save trace data and reload saved data.

1. **Save trace data**
   Saves the current trace data so that it can be reloaded at a later time. Saved trace data consists of multiple files including the raw trace data, a metadata file that captures target state, and a copy of the ELF file. These files are required to decode the trace. All files are saved to a directory that constitutes the saved trace.
2. **Load trace data**
   Loads a previously saved trace data set.

View Management Commands

Other Commands

The Trace Data view dropdown menu contains a few additional commands.



- **Copy layout to other views**
  This command copies the column layout of this view to any other trace viewers.
- **Reset view configuration**
  This command resets the trace data view to the default configuration.
- **Auto resize all columns**
  This command automatically sizes all displayed columns to completely fit displayed data.
- **Export to text file**
  This command exports the current trace capture to a text file. The text file content will reflect the currently displayed columns in the Trace Data view.

Trace Data Columns

The trace data view is highly configurable. Showing and hiding columns is controlled by:

- Right-click on any column header and select "hide column" to hide a displayed column.
- Right-click on any column header and select "manage column". This will bring up a column manager dialog where you can add and remove columns.
- From the view drop-down menu, select "reset view configuration" to only show columns that are displayed by default.

The following columns can be displayed (or hidden):

These columns are shown in the default configuration:

- **RT**
  Reports the relative timestamp for a trace record that includes a timestamp. The reported time is relative to the "origin" record. By default, the origin record is the first time-stamped record when "stop trace on buffer full" is enabled, and the last trace record then "stop trace on buffer full" is disabled.
- **Composite Output**
  This column is a general purpose column that reports data relevant to the trace record type. Source lines are reported here, as well as ITC prints, and formatted ITC data records, including channel information.
- **Addr**
  Shows the address of executed instruction records.
- **Dasm**
  This column shows the opcode and disassembly for all executed instructions.

The following columns are not shown by default. These must be manually turned on.

- **Note**
  The note column contains manually entered notes for a trace record.
  Core
  Shows the core index of the trace record.
  ISR

Shows the ISR nesting level. A nesting level of 0 indicates normal program execution.

ISR Time

If timestamps are configured to mark every branch message, then this column will report total time spent in each ISR. The time is reported on the last instruction of the ISR before returning from the interrupt.

This column is not shown by default.

- **C/R Type**
  This column shows the Call/Return type of a record.

- **Type**
  This column shows the "type" of trace record (i.e., instruction, source, ITC…). Mostly useful for debugging trace.
- **DT**
  This column reports delta time between timestamped records.
- **Opcode**
  This column reports the instruction opcode. This data is already included in the Dasm column, so this column is not shown by default.
- **Chan**
  Shows the originating ITC channel for an ITC message. This information is included in the Composite Output column.
- **ITCData**
  Shows ITC data record data values. This information is included in the Composite Output column.

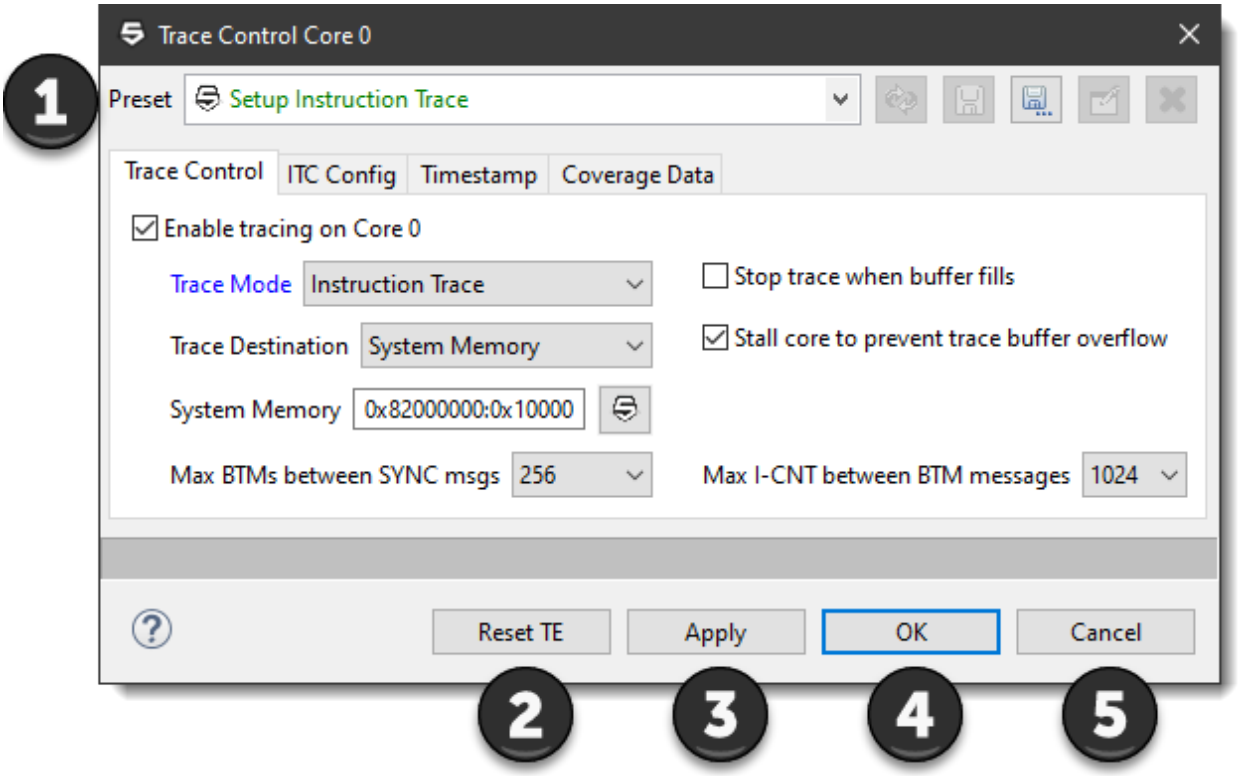
Trace Control Dialog

The trace control dialog is where all trace configuration happens. When the trace configuration dialog opens, all trace control states from the target system are loaded. This ensures that the dialog will always reflect the actual target state when opened. This is especially important if the software running on the target actively manages trace control.

The trace control dialog is generally organized as tabs for each major functional block. If a target system does not contain the functional block (for instance: timestamps), the corresponding tab will not be displayed.

Changes in the Trace Control dialog are not written to the target until either the Apply button of the OK button is pressed. You can cancel any changes by pressing the Cancel button.

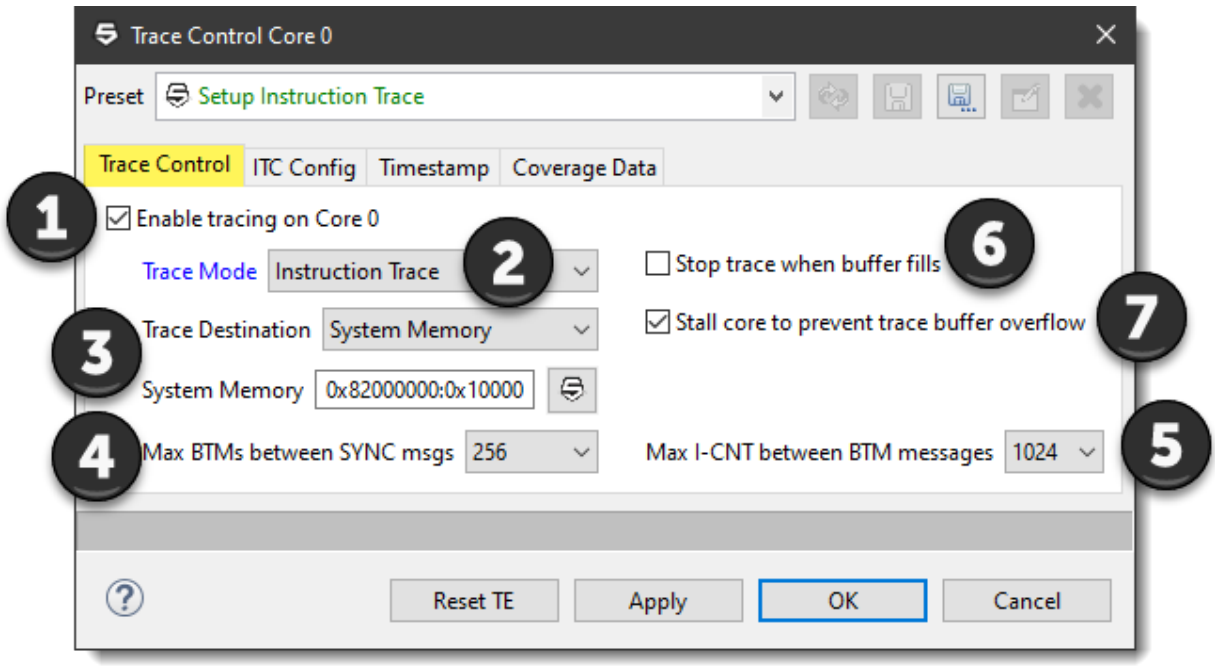
Before describing each of the function block tabs, let us go over the general trace controls (those controls outside of the functional tabs)



1. **Trace Presets**
   Trace presets are described in detail in the Trace Control Presets section.
2. **Reset TE Button**
   This button resets the trace encoder on the target system. This button is applied immediately to the target and is provided to easily reset the trace system when needed.
3. **Apply Button**
   Write any changes you have made to the target without closing the dialog.
4. **OK Button**
   Write any changes you have made and close the dialog.
5. **Cancel Button**
   Closes the dialog without writing any changes to the target.

Trace Control Tab

The Trace Control tab is the primary control center for the trace system. This tab is always present.



1. **Enable Tracing on Core X**
   This check box control trace encoder enablement on that active core. This is the same as the Core Enable checkbox on the toolbar. Replicated here for convenience.
2. **Trace Mode Selector**
   Select the trace mode. There are three modes available:
   a. **Instruction Trace**
      Generates instruction trace capturing instructions executed on the core. ITC messages are also generated.
   b. **Off (ITC w/o Instructions)**
      No instruction trace or SYNC messages are generated, but ITC messages are generated.
   c. **Periodic PC Sampling**
      Generates SYNC messages periodically as a trace-based form of PC sampling. Each SYNC message contains a full PC value. When this mode is selected the "Max I-CNT between BTM messages" control changes to become "Cycles between PC samples".
3. **Trace Destination/System Memory**
   Control where trace data is stored. Only options present of the target core will be displayed. Options include:
   a. **SRAM**
      SRAM is a dedicated chunk of RAM on the core. This option must be selected at design time. This option is not displayed if the target core does not have a dedicated SRAM buffer. The buffer size is also specified at design time.
   b. **System Memory**
      Trace can be captured in system memory. Using system memory allows for specifying where and how much trace should be captured. When this destination is selected the System Memory control must specify both the

destination address and size of the trace buffer. In the simplest form this can be specified as "0x<buffer-start-addr>:0x<length>". This can also be specified symbolically. See the section Specifying the System Memory Buffer for more detail.

c. **PIB**
Probe Interface Block. Specify this option when trace should be sent to the PIB block. This is used to send trace data to external probes that can capture trace directly to probe or host memory. Read about the PIB Config tab below for information on additional options that apply when the PIB option is selected here.

d. **ATB**
Advanced Trace Bus Select this if the target uses an ATB to route trace to another device.

4. **Max BTMs between SYNC Msgs**
Maximum number of BTMs between periodic Sync messages. A Sync emitted for another reason will reset this timer. For small trace buffers a smaller value should be used otherwise a wrapped trace buffer may not provide much, if any trace, since a SYNC message is required to start decoding trace.
5. **Max I_CNT between BTM message/Cycles between PC Samples**
   a. **Instruction trace mode**
   The maximum number of instruction messages between BTM messages. A BTM emitted for another reason will reset this timer. The maximum setting is dependent on the configuration.
   b. **Periodic PC Sampling mode**
   Specifies the number of clock cycles between PC (SYNC) messages.
6. **Stop trace when buffer fills**
Present only in systems with SRAM or System Memory sinks. When checked, disable trace when the trace buffer fills. Use this mode when you want to capture trace starting at the current PC and ending when the buffer fills. Let us call this "trace-from" mode. When unchecked the trace buffer will "wrap" overwriting older trace data with new trace data. This mode ensures that you capture trace up to the current PC when the target halts. Let us call this "trace-to" mode.

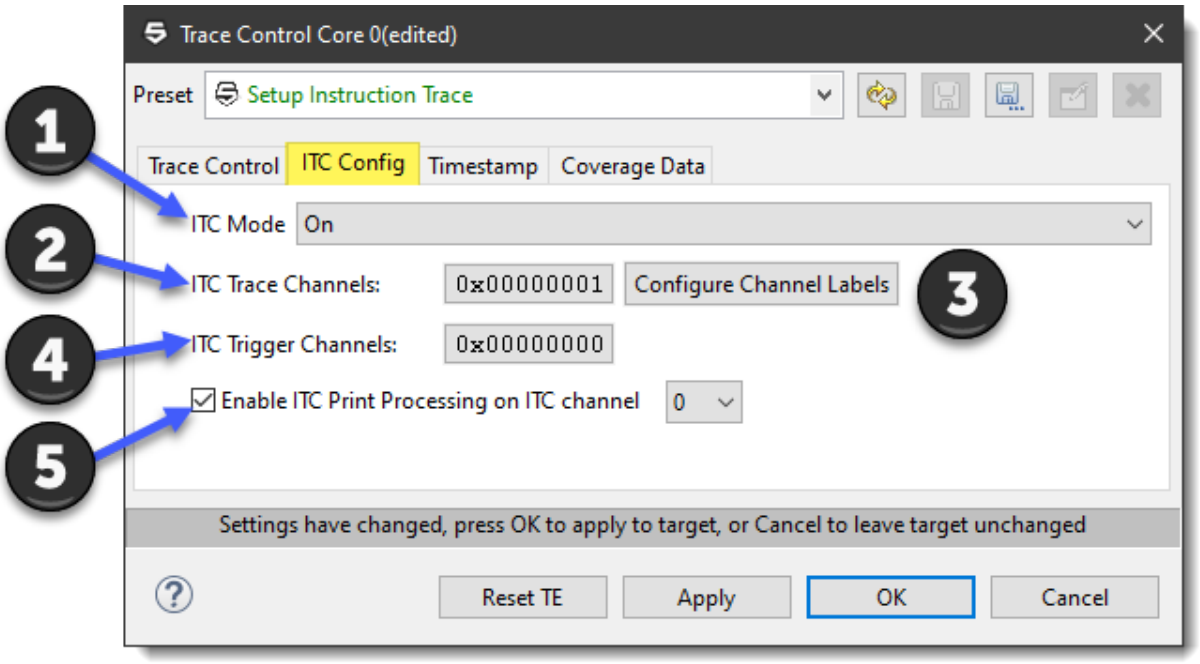
   The state of this checkbox affects what part of trace is displayed when the target halts. When checked the trace viewer will download and show the beginning of trace. When unchecked the trace viewer will download and show the end of trace.
7. **Stall core to prevent trace buffer overflow**
When checked the core will be stalled in the event the trace encoder cannot accept a new message. This mode ensures that every instruction is captured in trace. This mode may not be acceptable in time-critical applications where stalling the core is unacceptable. When unchecked, an overflow message is generated and trace data up to the next SYNC message will be lost. The Trace Data viewer will indicate when this happens with an overflow record.

ITC Config Tab

The Instrumentation Trace Component allows software running on the target system to inject data into the trace stream. This can be used, for example, to capture the change history of a variable, or to send important variable values along with related trace. ITC can also be used to send "printf" style output.

ITC messages are transmitted on "channels". There are 32 channels (only 16 physical channels, where channels 16-32 map directly to channels 0-15, but with slightly different behavior.) Data transmitted on channel 0 will also show up on channel 16.

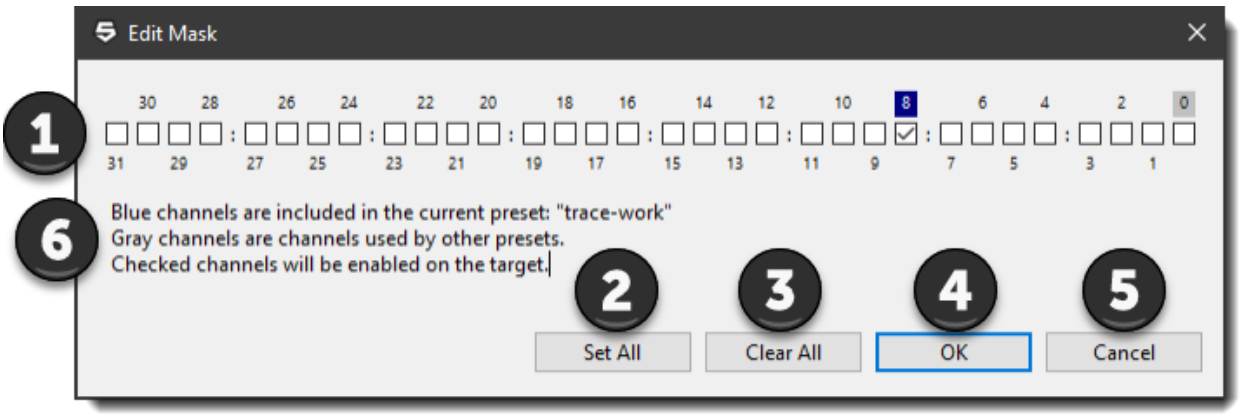
The ITC tab controls everything related to the Instrumentation Trace Component. This tab is not displayed if your target does not include ITC.



1. **ITC Mode Selector**
   The following ITC modes are provided:
   a. **None**
      ITC message generation is turned off. Instrumented code will not produce ITC messages in trace.
   b. **On**
      ITC message generation is on. Enabled ITC channels will generate ITC messages in trace.
   c. **Only ownership messages**
      All ITC Channels generate ownership messages.
   d. **On and 15/31 generate ownership messages**
      Channel 15 generates ownership message with no timestamp
      Channel 31 generates ownership messages with a timestamp
      All other channels generate not ITC data messages

2. **ITC Trace Channel Enables**
   This field controls which ITC data channels are enabled. The code running on the target may use many ITC channels, but only enabled channels will be encoded in the trace output. This allows for controlling output at debug time rather than compile time.

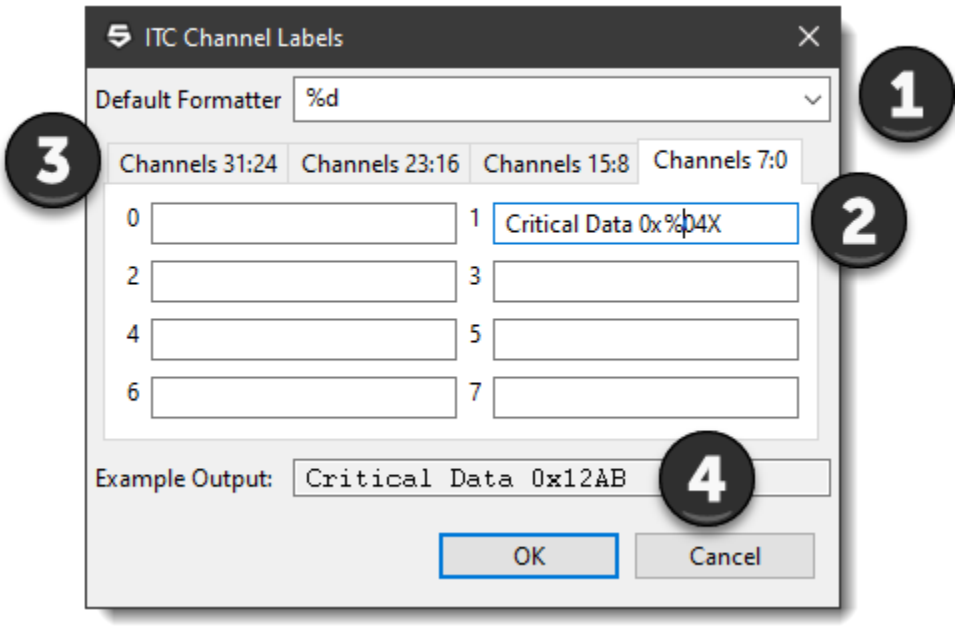
   The channel enable is expressed as a 32bit hex value where each '1' bit denotes an enabled ITC channel. Pressing the button will open the ITC channel editor:



   a. (1) Channels enable checkboxes. A checked channel will generate ITC messages in trace output.
   b. (2) Set All: Enables all channels
   c. (3) Clear All: Disables all channels
   d. (4) Close the dialog, applying the channel enables.
   e. (5) Close the dialog, ignoring any changes made.
   f. (6) Notes describing visual indicators on the dialog.

3. **ITC Channel Formatters**
   Channel formatters can be used to label and format ITC data in the trace viewer. You can assign a different format and label to each channel, as well as a default format and label to any channel that does not have a specific formatter. Formatters are simply printf-style format strings with a single "%d" or "%x" value specifier.

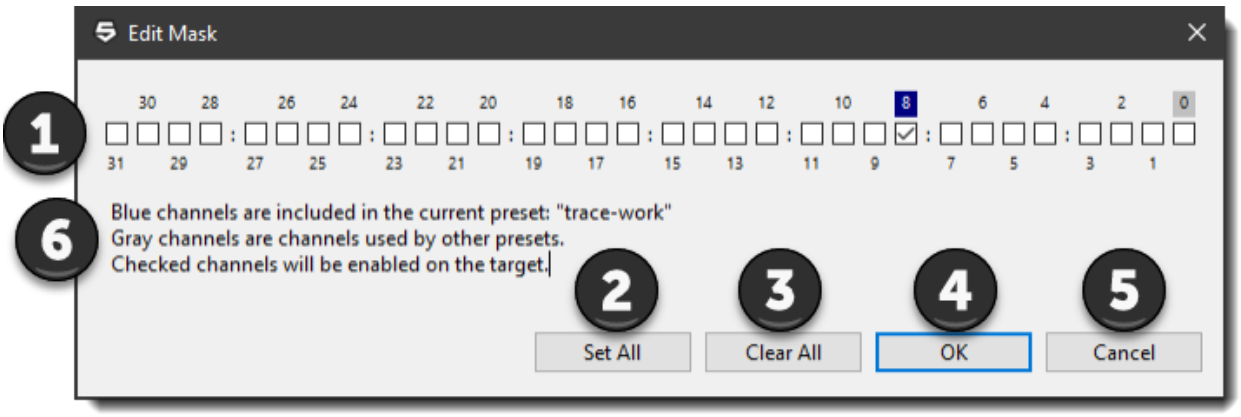
Pressing the button opens the ITC Channel Labels dialog box:



a. (1) Specify a default formatter. Use the dropdown to select from some predefined formats or type your own custom format. The example field will show how your formatter will display.
b. (2) An example formatter for a specific ITC channel.
c. (3) Channel tabs. Use these tabs to navigate among the 32 ITC channels.
d. (4) Example output. This box shows how the currently editing channel formatter will be displayed using some sample data. If there is an error in the formatter string, the error message will be displayed here.

4. **ITC Trigger Channels**
This field controls which ITC trigger channels are enabled. The code running on the target may use many ITC channels, but only enabled channels will be encoded in the trace output. This allows for controlling output at debug time rather than compile time.

The channel trigger enable is expressed as a 32bit hex value where each '1' bit denotes an enabled ITC trigger channel. Pressing the button will open the ITC channel mask editor:

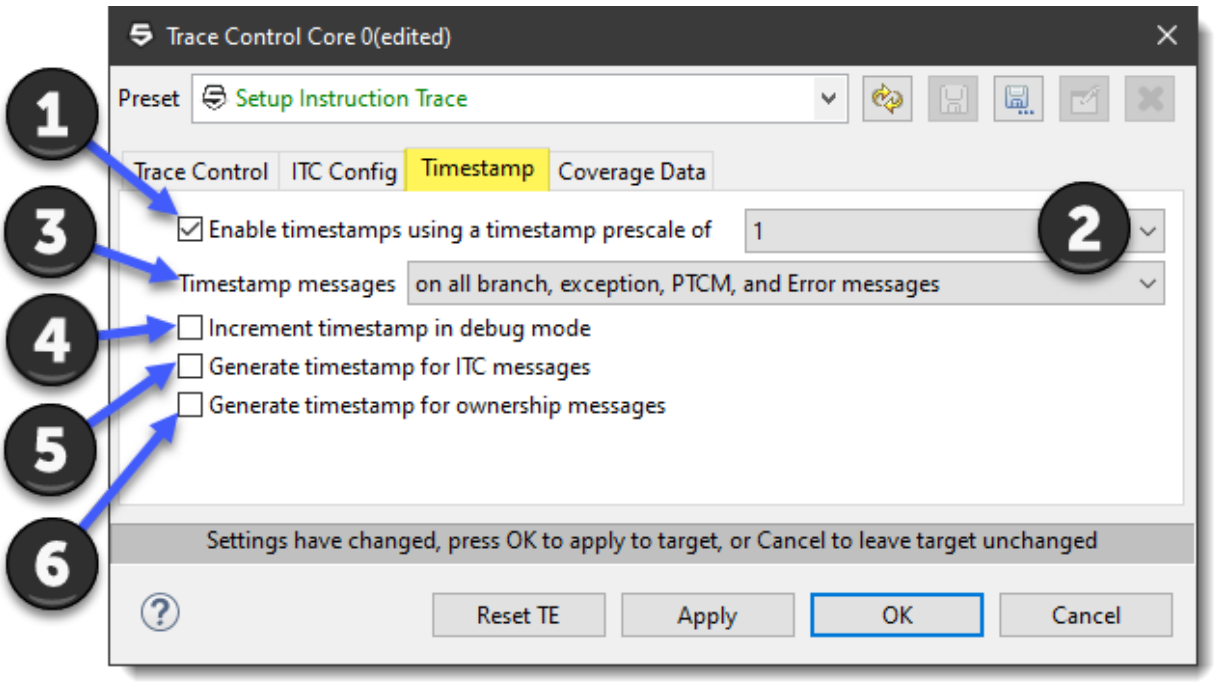


a. (1) Channels enable checkboxes. A checked channel will generate ITC trigger outputs.
b. (2) Set All: Enables all channels
c. (3) Clear All: Disables all channels
d. (4) Close the dialog, applying the channel enables.
e. (5) Close the dialog, ignoring any changes made.
f. (6) Notes describing visual indicators on the dialog.

5. **Enable ITC Print Processing on ITC Channel**
   Use this control to specify which ITC channel will be used for "print" processing. Only a single channel can be used to send "print" style messages.

Timestamp Tab

The Timestamp tab controls how and when timestamps are generated on the target. This tab is not displayed if your target system does not include timestamping.

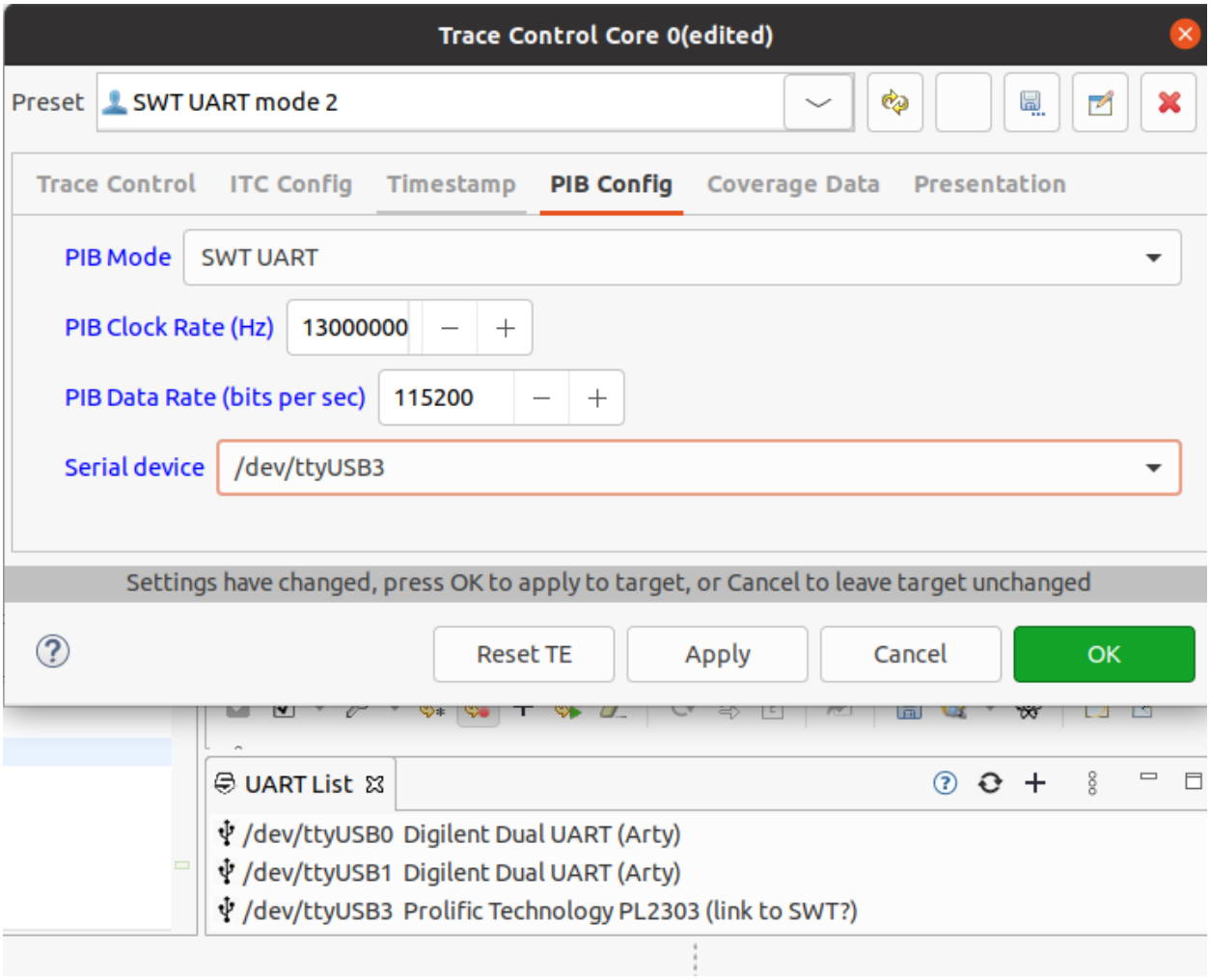


1. **Enable timestamp generation**
   The master switch that controls generation of timestamp data in the trace output for the selected core. When checked, timestamp data will be generated as specified by the remaining controls.
2. **Timestamp prescale selection**
   Prescale timestamp clock by the selected value. The timestamp clock is divided by the selected scale factor, effectively slowing the timestamp clock. This can be useful for measuring longer intervals (the timestamp clock will run longer before wrapping) and allow for using a narrower timestamp (which conserves power).
3. **Timestamp message generation mode**
   This setting determines when (on what types of messages) timestamps are generated. Options are:
   a. **are not generated for branch messages**
   b. **are generated on all indirect branch and exception messages**
   c. **are generated on all branch, exception, PTCM, and Error messages**

4. **Increment timestamp in debug mode**
   When checked the timestamp clock will continue to increment while in debug mode. When not checked, the clock will stop while in debug mode.
5. **Generate timestamp for ITC messages**
   When checked, timestamps will be generated for all ITC messages on channels 16 to 31 (Channels 0-15 do not generate timestamps). When not checked, ITC messages will not include timestamps.
6. **Generate timestamp for ownership messages**
   When checked, timestamps will be generated for all ownership messages. When not checked, ownership messages will not include timestamps.

PIB Config Tab

The PIB Config tab controls the trace hardware configuration details that affect the operation of the trace block when PIB is active as the trace destination (see Trace Destination selector on the Trace Control tab). Currently, the only PIB mode that Freedom Studio works with is the Serial Wire Trace (SWT) UART mode, that involves connecting a USB/serial adapter cable between the host machine running Freedom Studio and the target board. Below is a snapshot of this tab as it appears when running Freedom Studio on Ubuntu host.



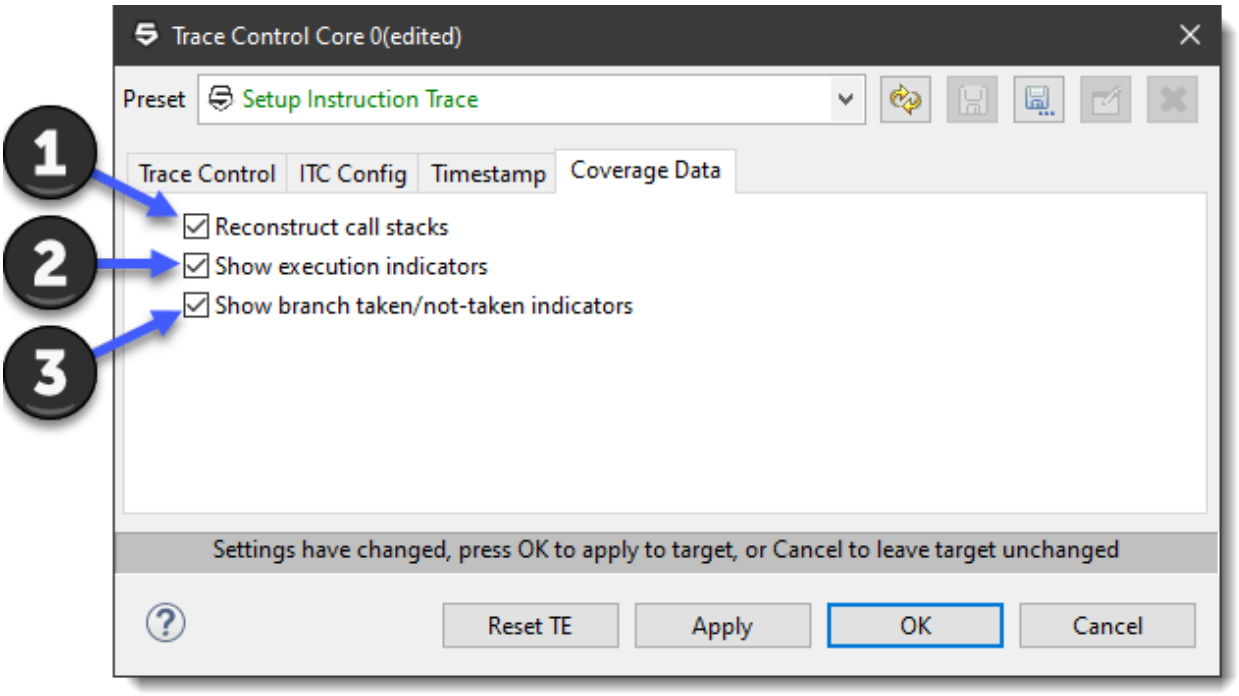
1. **PIB Mode**
   This option controls the physical signaling and protocol that the PIB uses to stream trace through external pins of the board to external hardware. Freedom Studio currently supports only a mode called SWT UART (SWT stands for single wire trace), which streams trace with signaling characteristics that work well with commodity USB/serial adapters (e.g. PL2303 or CP210x chipsets). 3rd party

toolchain partners may have solutions that work with other PIB modes that involve higher-performance streaming of trace from SiFive cores. Consult with your SiFive sales representative for more details.

2. **PIB Clock Rate**
   This is the clock rate driving the PIB itself. On ARTY FPGAs, for instance, this is the same as the bus clock frequency. Freedom Studio needs to know this in order to tune the clock divisor for the UART output to match your desired baud rate (see below).
3. **PIB Data Rate**
   This specifies the baud rate to use. Commodity USB/serial adapters on the various host OSes seem to like 115200 as a least common denominator, though some of the CP210x-based adapters seem happy with 921600. The PIB on the SiFive core can support baud rates up to the PIB clock rate itself, but the limiting factor will be on the side of the USB/serial adapter. One recommendation would be to start with 115200, and if higher baud rates are desired, try 921600 and beyond. Some combinations of adapter, and OS driver support for that adapter, may establish a practical limit of 115200.
4. **Serial Device**
   This identifies the COM port (Windows), or /dev/ttyUSBn device (Linux or MacOS) to which Freedom Studio should connect, when offloading trace in an SWT UART tracing session. Consult Freedom Studio's UART List view for strong hints at which device to choose (or use Windows Device Manager, or Linux lsusb, etc). It will correspond to the particular USB/serial adapter you have acquired and connected.

Coverage Tab

The coverage tab controls generation of code coverage and call stack reconstruction data. The controls on this tab do not directly control any target state. The controls are described here. Trace based code coverage is described in more detail in a later section.



1. **Reconstruct call stacks**
   When checked, call stack data is reconstructed from trace data. Call stack reconstruction lets you examine the call stack from any trace record. This data is required to use the Freedom Studio Call Stack views.
2. **Show execution indicators**
   When checked, execution indicators are displayed in the source editors and disassembly view.
3. **Show branch taken/not-taken indicators**
   When checked, branch taken indicators are displayed in the source editors and disassembly view. Branch taken indicators show, for every branch, whether the branch was always taken, never taken, or fully covered.

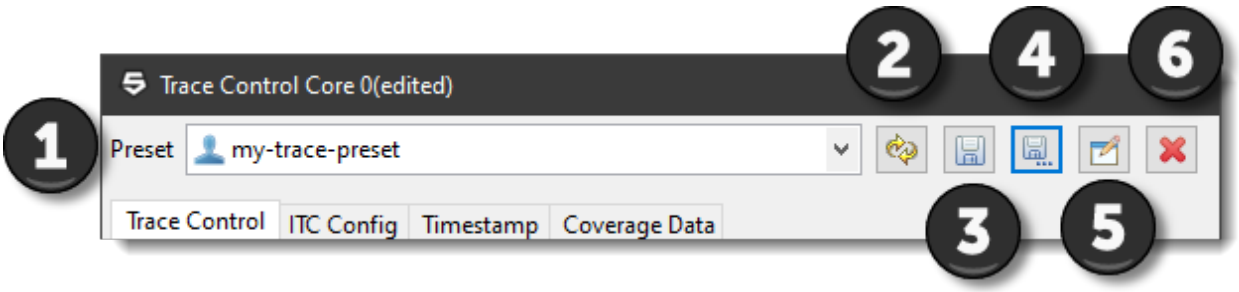
Trace Control Presets

Trace control presets allow you to set up custom trace configuration and save them for future use. Trace presets can also be applied at debug launch time to fully configure trace from the start of code execution.

Freedom Studio includes a selection of built-in trace presets that can be used as-is, or as starting points for new presets.

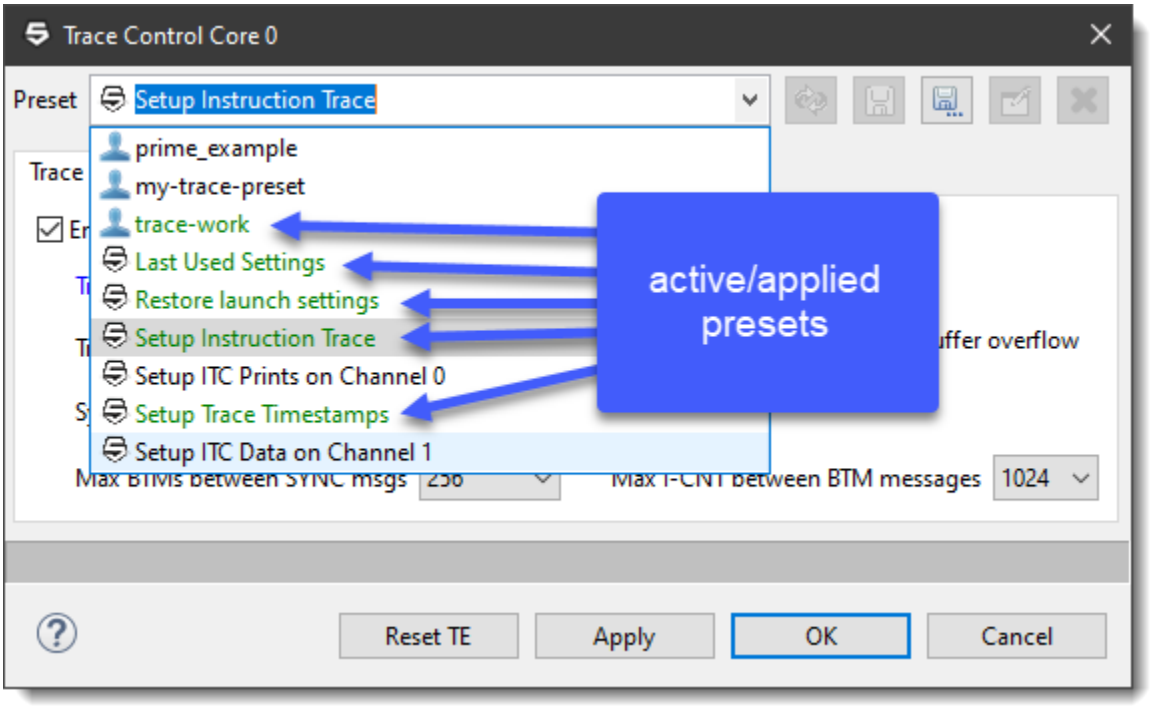
Trace Presets are “cumulative”, meaning that any given preset may only specify a subset of trace settings, and that multiple presets can be applied simultaneously. This allows you to build various custom presets for different use-cases and then apply one or more of them depending on your immediate needs while debugging.

Presets are managed using the control at the top of the trace control dialog.



1. **Preset selector**
   The preset selector is a drop-down list of all known presets. Selecting a preset from the list will update the trace configuration dialog with the setting included in the preset. The selected preset (that is, the one shown in the box) is considered the "Active" preset for editing. Changes made to settings included in this preset can be saved to the preset.
2. **Undo preset changes**
   This button reverts any changes you've made to the selected preset settings while the Trace Configuration dialog is open. This button is disabled for built-in presets as they cannot be edited.
3. **Save preset**
   This button saves any changes you made to the selected preset. This button is disabled for built-in presets as they cannot be modified.
4. **Save a new preset**
   Use this button to create a new preset. See the Editing & Creating Trace Presets
5. **Edit Preset Mode Toggle**
   Edit mode is a special mode for editing which trace control settings are included in a preset. See the Editing & Creating Trace Presets
6. **Delete User Preset**
   Deletes the selected user preset. This button is disabled for built-in presets.

When dropping down the Preset Selector you may notice that some presets are green while others are black. Green presets are those that match the current trace configuration settings and values. That is, all settings from a blue preset are active and current.



This provides a quick way of verifying which presets have been applied.

Presets with a "person" icon are user defined presets. Those with a SiFive icon are built-in presets.

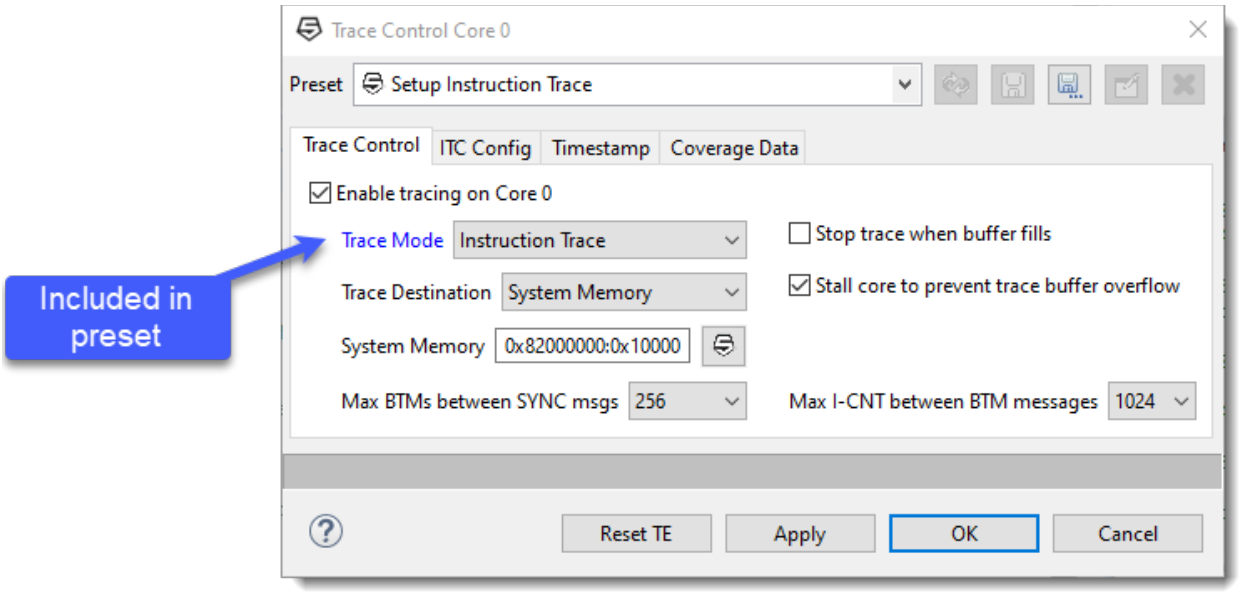
Special Built-in Presets

There are a selection of special built-in presets that may be handy in some use-cases.

- **Last Used Setting**
  This built-in preset is created each time the Trace Configuration dialog is closed. It contains all of the trace configuration settings that were just written to the target.
- **Launch Settings**
  This preset captures all of the trace configuration settings that were applied at the start of the debug session. It is a handy way of "reverting" to the launch settings.

What is included in a Trace Preset

The Trace Control dialog indicates settings that are included in a preset using blue labels. For instance, in this example the built-in "Setup Instruction Trace" preset includes the "Trace Mode" setting because the Trace Mode label is blue.



All other settings (those with black labels) are not included in this preset. Their values remain unchanged when this preset is selected.

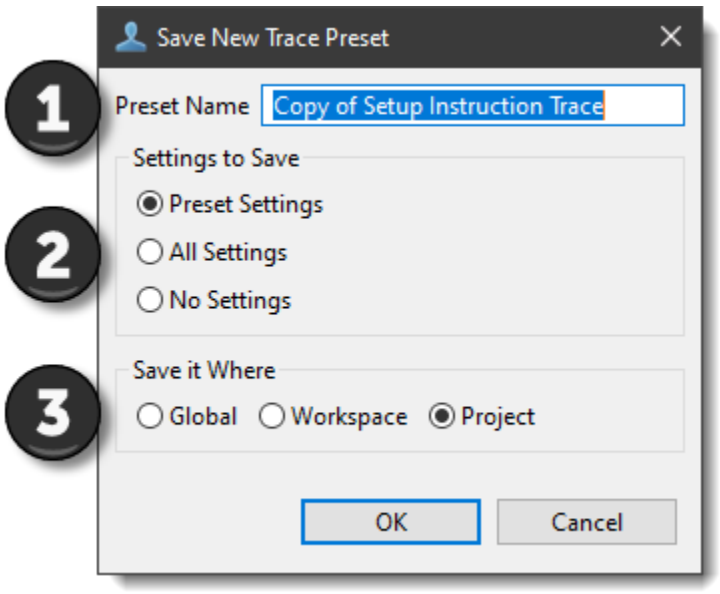
Editing & Creating Trace Presets

Creating Trace Presets

Freedom Studio ships with a selection of basic built-in trace presets. You will quickly want to develop your own custom presets. When creating presets you can start with the settings of an existing preset, start with a blank preset, or start with a preset that includes all trace control settings.

Changes to built-in presets cannot be saved back to the built-in preset. To save these changes you will need to create a new custom preset.

To create a new preset press the New Preset button to open the new preset dialog.



1. **Preset Name**
   Give your new preset a meaningful name. This is the name that will be listed in the preset selector drop-down.
2. **Settings to Save**
   These radio button determine which setting will initially be included in the preset:
   a. **Preset Settings**
      If a preset was selected when this dialog is opened, then this option will be enabled and selecting it will include all settings from the current preset into the new preset. Once created you can simply add or remove settings, and change setting values as needed.
   b. **All Settings**
      Selecting this option will include all trace control settings (and values) in the preset. Use this if you want to simply save everything into a single preset. Once created you can easily remove specific settings if desired.
   c. **No Settings**
      Start with a blank preset that includes no setting (and no values). Use this when you want to create a new preset that does not easily derive from an existing preset. After you have created a blank preset you need to add settings to the preset before it will be useful.
3. **Save it Where**
   Trace presets are saved as simple text files. Trace preset can be saved to different locations depending on the desired scope of usage:
   a. **Global**
      Global presets are saved in a subfolder of the Freedom Studio installation folder. These presets are available in any workspace associated with the Freedom Studio installation.
   b. **Workspace**
      Workspace presets are saved in the workspace metadata folder and are

available to all projects in the workspace. They are not available to projects in other workspaces.

c. **Project**
   Project presets are saved in the project .settings folder and are only available in the project. This is the best option if you want to commit a trace preset setting so a revision control system so that other users will have access to it.

Once you save your new preset it becomes the selected and active preset in the Trace Control dialog. You can now add or remove settings and update the values of included settings. Remember to save any changes you make to settings inclusion or setting values.

Editing Trace Presets

There are two aspects to editing trace presets. The first is simply editing the value of a trace preset setting. This is simple. Just change the setting value to the desired value and save the preset.

The second aspect is adding or removing settings from the preset. There are three ways to do this:

1. **Ctrl-click on the Settings Label**
   Ctrl-clicking on a setting label will toggle inclusion of the setting in the selected/active preset. You will see the label color toggle between the blue (indicating inclusion) and black (not included). This is the easiest way to manage which settings make up a preset.
2. **Use Edit Mode**
   Pressing the Edit Mode button in the Preset Control area causes the Trace Control dialog to enter "edit" mode. In edit mode all trace settings gain a checkbox next to the label. The checkbox state indicates inclusion (check) or exclusion (not checked). While in edit mode, setting values cannot be changed.
3. **Edit the Preset Settings File in a Text Editor**
   For adventurous users only! If you hold the Ctrl key down when pressing the Edit Mode button Freedom Studio will open the preset file in a text editor. Now you can edit the preset file directly. Be forewarned, it is easy to render the preset unusable by making bad edits. Many comments are provided to help guide you, but you are really on your own here.

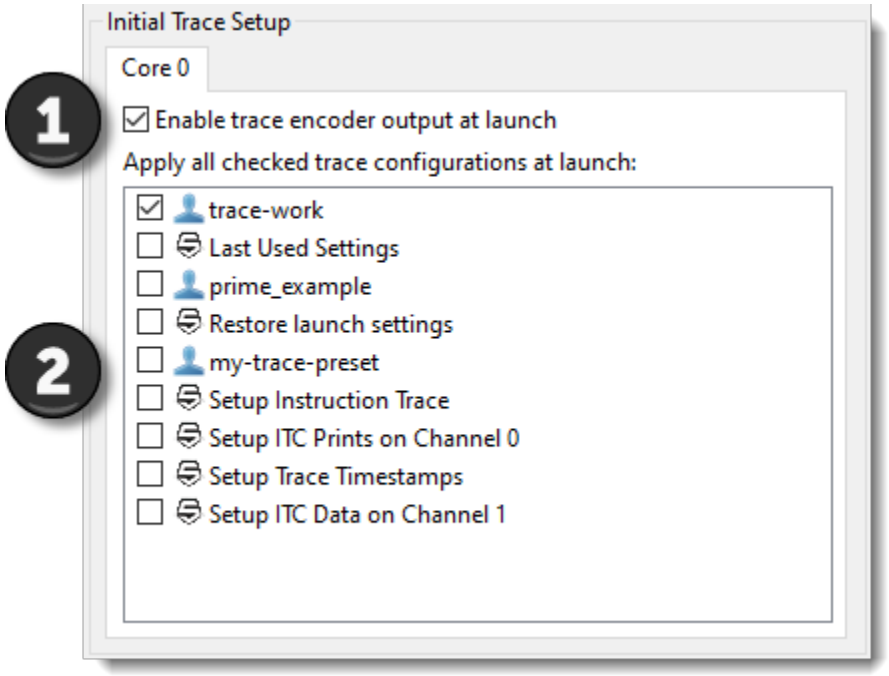
Every setting in the Trace Control dialog can be included in a preset. ITC Trace Channels and ITC Trigger Channels can be included/excluded in a preset on an individual channel basis. So you can have one preset that configures channels 1,3, and 5; and another preset that configures channels 2 and 4. These presets can be applied simultaneously without interfering/overwriting each other. Configuring preset channels is done the same way as any other preset setting, but you do need to open the channel editor to do so.

ITC Channel Formatters are also stored in the preset. In this case a given ITC channel formatter is considered included in the preset when the corresponding ITC channel is included.

Tracing at Startup

Trace can be configured at the start of a debug launch so that no user intervention is required to configure and enable tracing.

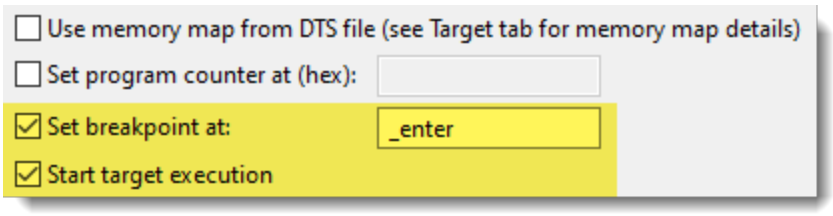
The Debug Launch Configuration dialog Startup Tab has controls for enabling and configuring trace on all cores. Look for the **Initial Trace Setup** section:



On a single core target there is only one tab (for Core 0). Multi-core target will include a tab for each core. Each core's trace can be configured independently. On a multi-core target you have the option to apply the same trace configuration setting to all cores, or different settings to each core.

1. **Enable trace encoder output at launch**
   This checkbox tells Freedom Studio to enable (or disable) the trace encoder for the indicated core at launch.
2. **Preset List**
   Any checked presets will be applied to the target at launch. Note that if you do not check any presets, then the current target settings will be used.

The trace configuration is programmed to the target after the target halts and hands control to the debugger. Typically this will be after hitting the temporary breakpoint at main(). If you want to trace the code execution starting at the program entry point the debug launch needs to be configured differently. On the Debug Launch Configuration dialog Startup tab, make these settings:



Now the target will halt at `_enter` and the trace configuration will be programmed. Go ahead and set a normal breakpoint at the entry to main() and run the target. You will capture the entire startup sequence in trace. This is a great way to get familiar with the startup sequence.

Specifying a System Memory Buffer

When storing trace data to system memory the trace encoder must be configured to know where the buffer starts and how big the buffer is. This can be done in code (requiring code to do so). Or it can be done at the beginning of a debug session using a trace preset.

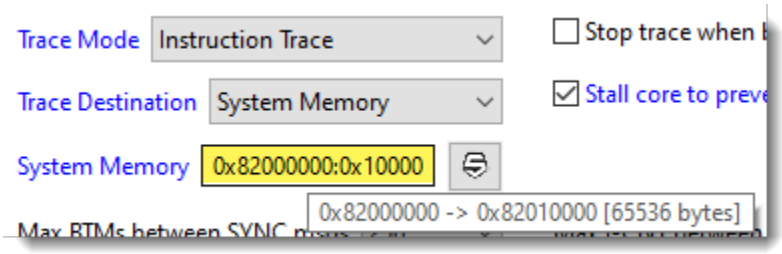
Below are three methods of specifying a system memory buffer for the trace-encoder using Freedom Studio.

Use a hard-coded address and length

This is perhaps the easiest method to configure, but it does require detailed knowledge of the target memory map. Use this method to specify that absolute start address and length of the trace buffer. Just be very sure that no other code or data will be placed in this region.

Specify the region as 0x<start-address>:0x<length>

For example:



In this example I'm absolutely sure that no code or data will be located in the specified memory.

Reserve a Section using the Linker Script

This is the best method as it ensures (automatically) that the linker will not place code or data into the section dedicated for trace capture. It is also the most complicated method to set up initially. Using this method you will create a dedicated section in the memory map, define linker symbols that denote the start and end address of the buffer, then use these symbols in the trace configuration to configure the trace encoder.

First (optionaly) create a copy of the linker script. In most cases this will be a copy of `metal.default.lds` (it is located in the bsp folder of any example software project). Call the copy something like `metal.trace.lds`

Now modify your makefile to use this linker script. In an example software project you will want to edit the Makefile in the project root folder. For instance, make the following edit:

```
ifeq ($(LINK_TARGET),)
LINK_TARGET = ~~default~~trace
endif
```

Now open metla.trace.lds in the text editor and add the highlighted text to the "MEMORY" block:

```
MEMORY
{
      itim (airwx) : ORIGIN = 0x1800000, LENGTH = 0x8000
      ram (arw!xi) : ORIGIN = 0x80000000, LENGTH = 0x10000000
      rom (irx!wa) : ORIGIN = 0x40400000, LENGTH = 0xc00000
      /*
      * Reserve 64KB for trace-encoder
      */
      trace (arw!xi): ORIGIN = 0x82000000, LENGTH = 0x10000
}
```

Adjust the address and length to suit your target system.

Just below the "MEMORY" block is the "PHDRS" block. Add the highlighted text:

```
PHDRS
{
      rom PT_LOAD;
      ram_init PT_LOAD;
      tls PT_TLS;
      ram PT_LOAD;
      itim_init PT_LOAD;
      /*
      * Do not load this section
      */
      trace PT_NULL;
}
```

At the very bottom of the file (just before the last closing brace) add this highlighted text:

```
.heap (NOLOAD) : ALIGN(4) {
      PROVIDE( __end = . );
      PROVIDE( __heap_start = . );
      PROVIDE( metal_segment_heap_target_start = . );
      /* If __heap_max is defined, grow the heap to use the rest of
RAM,
      * otherwise set the heap size to __heap_size */
```

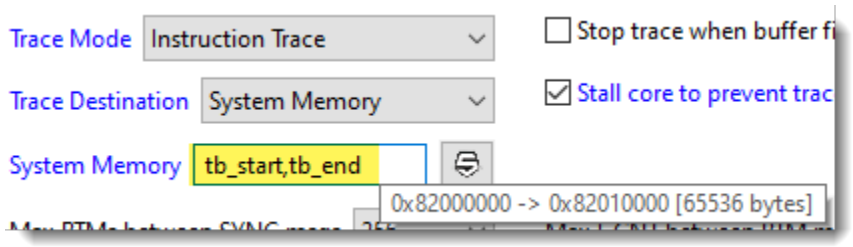
```
        . = DEFINED(__heap_max) ? MIN( LENGTH(ram) - ( . - ORIGIN(ram)) ,
0x10000000) : __heap_size;
        PROVIDE( metal_segment_heap_target_end = . );
        PROVIDE( _heap_end = . );
        PROVIDE( __heap_end = . );
} >ram :ram

.tracebuffer (NOLOAD)   : ALIGN(4) {
        PROVIDE( tb_start = . );
        . += LENGTH(trace);
        PROVIDE( tb_end = . );
} >trace
```

You are done with the linker script. Save it and go back to the makefile. We will configure the build to not remove the tb_start and tb_end symbols. Add the highlighted text:

```
RISCV_LDLIBS += -Wl,--start-group -lc -lgcc -lm -lmetal
$(LIBMETAL_EXTRA) -Wl,--end-group

# Dont remove this symbol
RISCV_LDLIBS += -u tb_start -u tb_end
```

A full rebuild is required. Save the makefile and do a full clean and rebuild.

Now, in the Trace Configuration dialog you can specify the trace region as:



Use a static char buffer allocated in code

You can define a large static char[] in your code. For instance:

```
char trace_area[4096] __attribute__ ((aligned (8)));
```

This defines a 4KB buffer that can be used to capture trace. But the linker will likely remove this symbol since it will be detected as unused. You need to configure the build to not remove this symbol. You can do this in code by simply creating a "use" of it, like this:

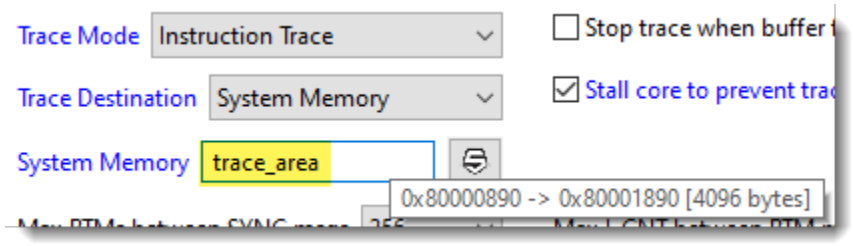
```
int main() {
    trace_area[0]=0;
    primes();
}
```

Where simply writing a 0 to the first element is enough to convince the toolchain not to remove the symbol.

Or you can modify the Makefile to provide "-u trace_area" to the linker, for instance:

```
# Do not remove this symbol
RISCV_LDLIBS += -u trace_area
```

With such an array defined (and not removed), you can specify this in the trace configuration dialog as:



When applied to the target, trace will be written to the specified memory region. Hovering the mouse pointer over the box will display a tooltip with the resolved address and length of the buffer.

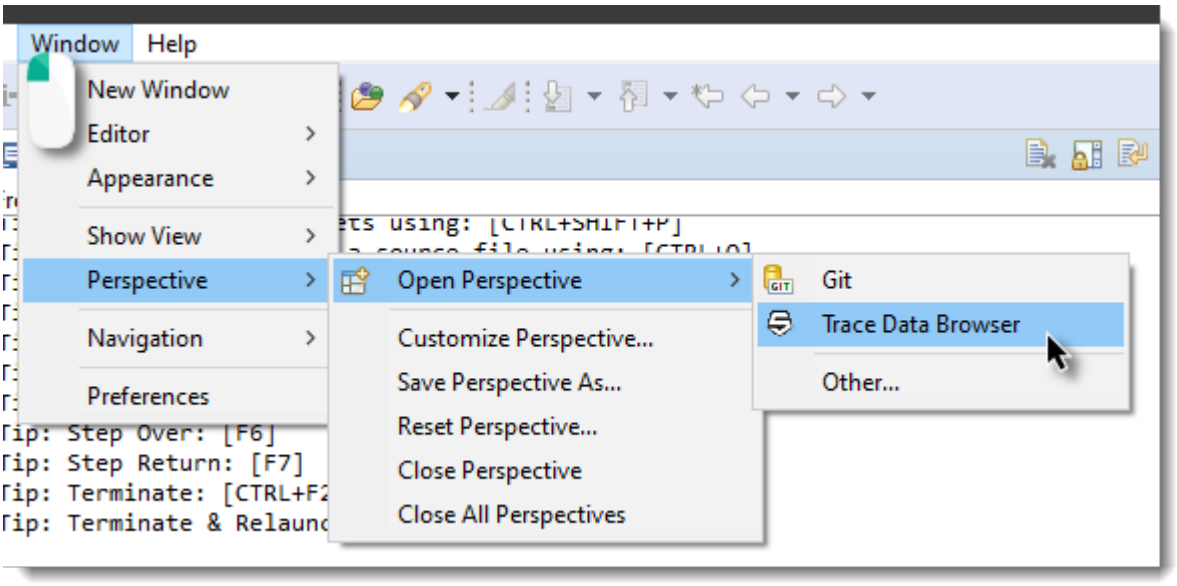
WARNING

The main drawback of this approach is that you cannot use it to trace startup code. A buffer allocated in this way gets initialized to 0s during startup. If the trace encoder is writing to the buffer while the startup code is also writing to the buffer, the trace data will be corrupt.

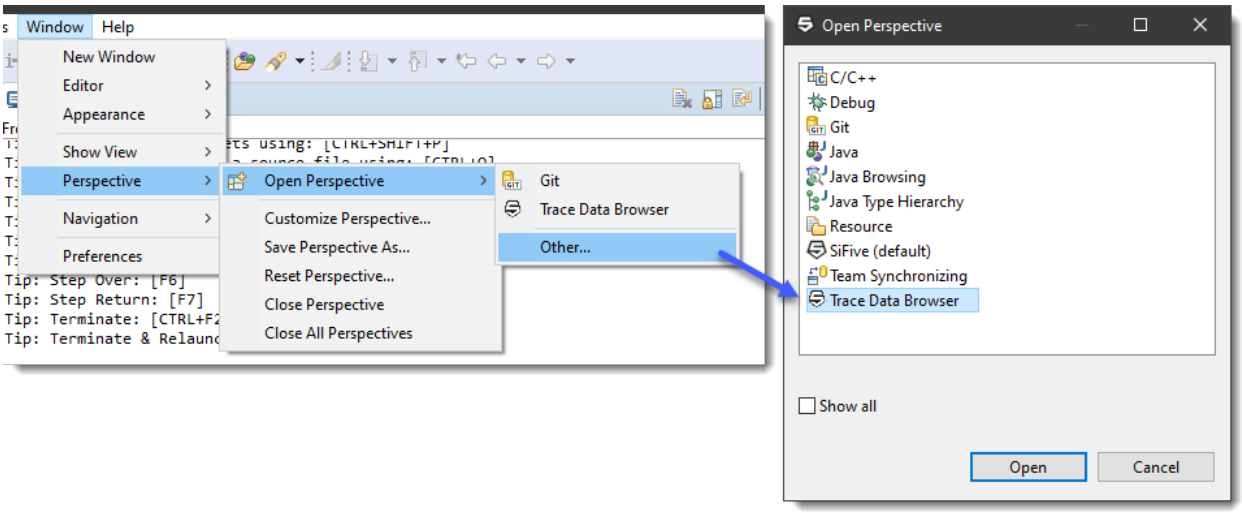
Trace Perspective

Freedom Studio provides a dedicated perspective for examining trace data. Using this perspective is entirely optional, but provides a cleaner workspace with most of the normal debug views removed.

If you are in the SiFive or Debug perspectives, you can open the Trace Data Browsing perspective from the Window → Perspective → Open Perspective → Trace Data Browser



From any other perspective, you can open the Trace Data Browser using Window → Perspective → Open Perspective → Other… and selecting the Trace Data Browser from the list.



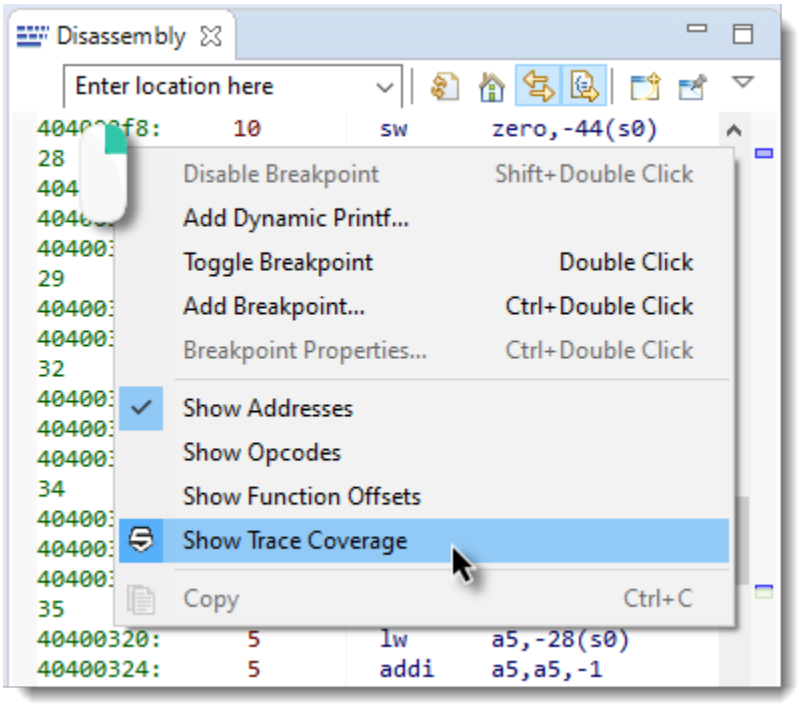
The perspective toolbar provides quick access to the Open Perspective dialog.

Trace-based Code Coverage

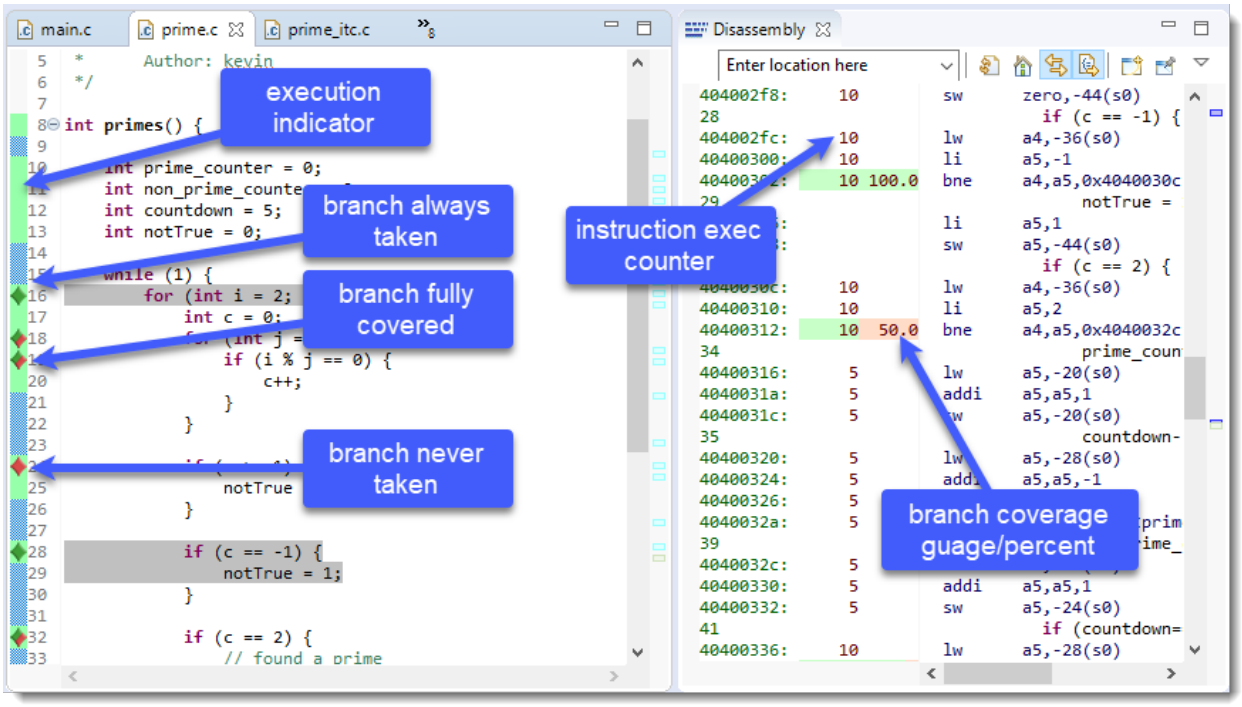
Code execution and branch taken/not-taken coverage data can be reconstructed from trace data. The following Trace Configuration settings are required:

- **Trace Control : Trace Mode : Instruction Trace**
- **Coverage Data : Show execution indicator**
- **Coverage Data : Show branch taken/not-taken indicators**

By default the Disassembly view does not display the coverage ruler. You can turn this on by right-clicking on the "address" column and selecting "Show Trace Coverage"



This screenshot shows examples code execution and branch coverage indicators:



Executed source lines are indicated (by default) using a green gutter marker on the left hand gutter in source editors. The color can be changed in Preferences.

Executed instructions in the disassembly view are indicated using counts.

Branch coverage in source files is indicated with gutter icons that indicate always-taken, never-taken, and fully-covered (as shown in the screenshot above).

Branch coverage in the disassembly view is indicated using a bar gauge where green indicates the percentage taken, red indicates the percentage not-taken. And a percentage taken value is displayed at the right edge of the gauge.

Trace Call Stacks

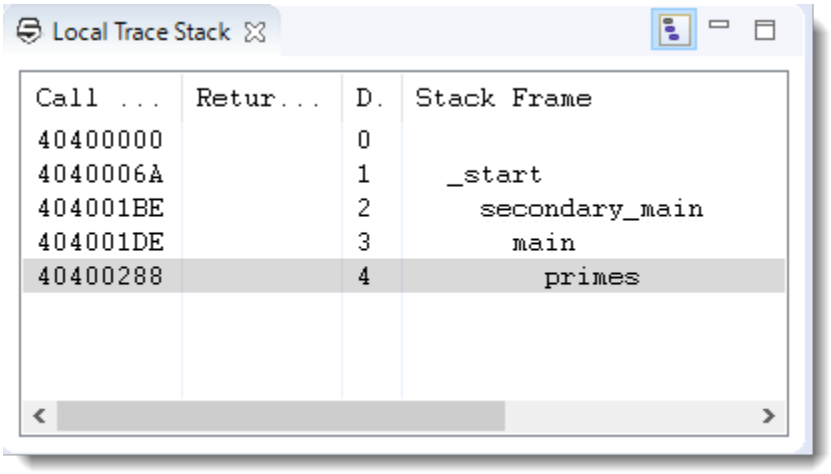
Call stack information can be reconstructed from trace data. The following Trace Configuration settings are required:

- **Trace Control : Trace Mode : Instruction Trace**
- **Coverage Data : Reconstruct call stacks**

Freedom Studio includes two viewers to display reconstructed call stack information. Both call stack views react to selection of trace records in the Trace Viewer.

Local Call Stack View

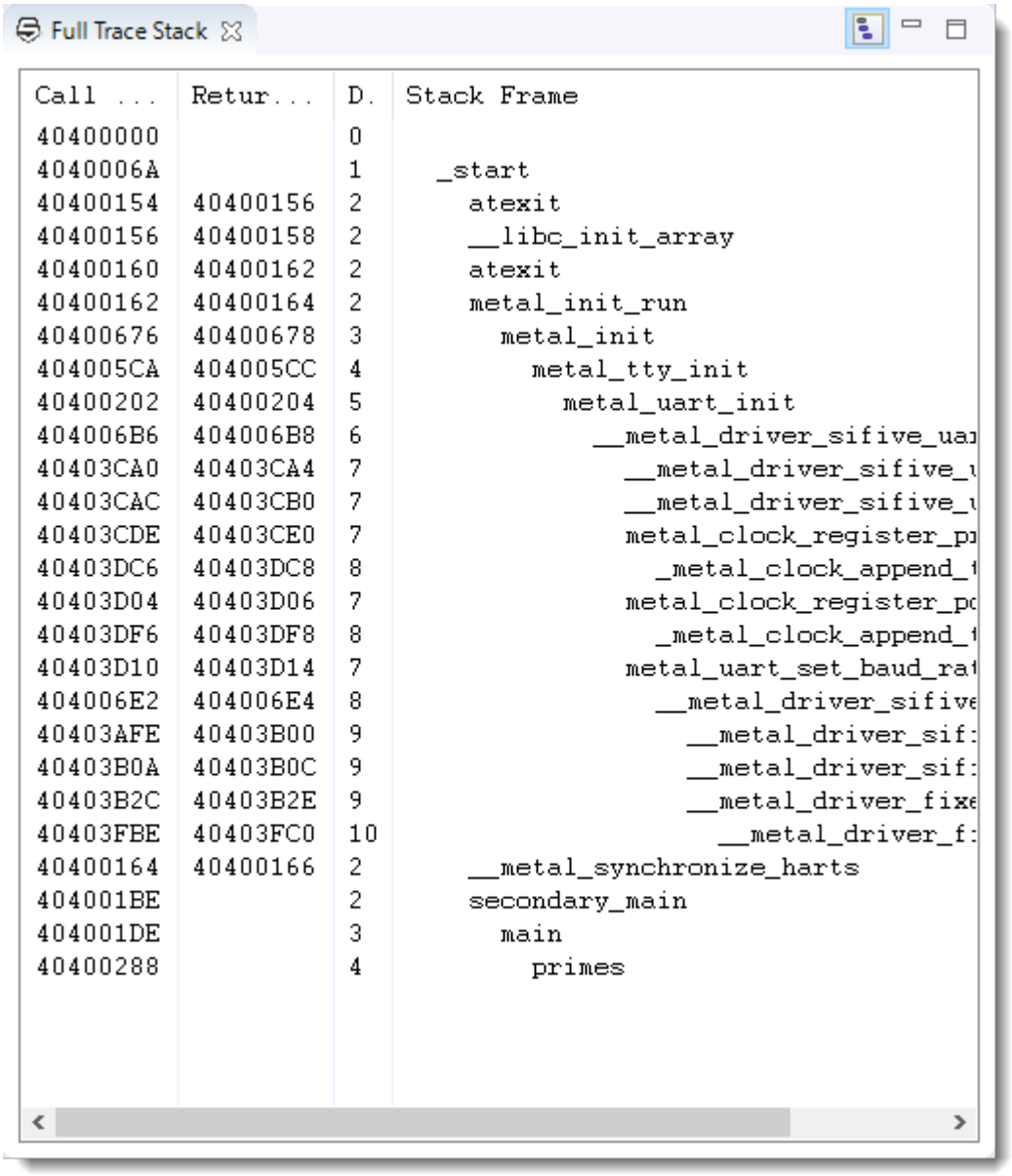
The Local Call Stack view shows the call stack above the currently selected trace record. Each frame in the call stack also shows the address where the frame was called from, and the address where the frame returns to. You can click on the called-from and return-to addresses and the corresponding trace frame will be selected in the Trace Viewer (and if editor synch is enable the source and disassembly views will also show the context of the trace record)



The stack frame column can be displayed with an indent or flat using the toolbar button.

Full Call Stack View

The Full Call Stack view shows the entire call-stack reconstruction from trace. You can follow the calling hierarchy up and down the call stack from the beginning of trace to the last instruction traced. Each frame in the call stack also shows the address where the frame was called from, and the address where the frame returns to. You can click on the called-from and return-to addresses and the corresponding trace frame will be selected in the Trace Viewer (and if editor synch is enable the source and disassembly views will also show the context of the trace record)

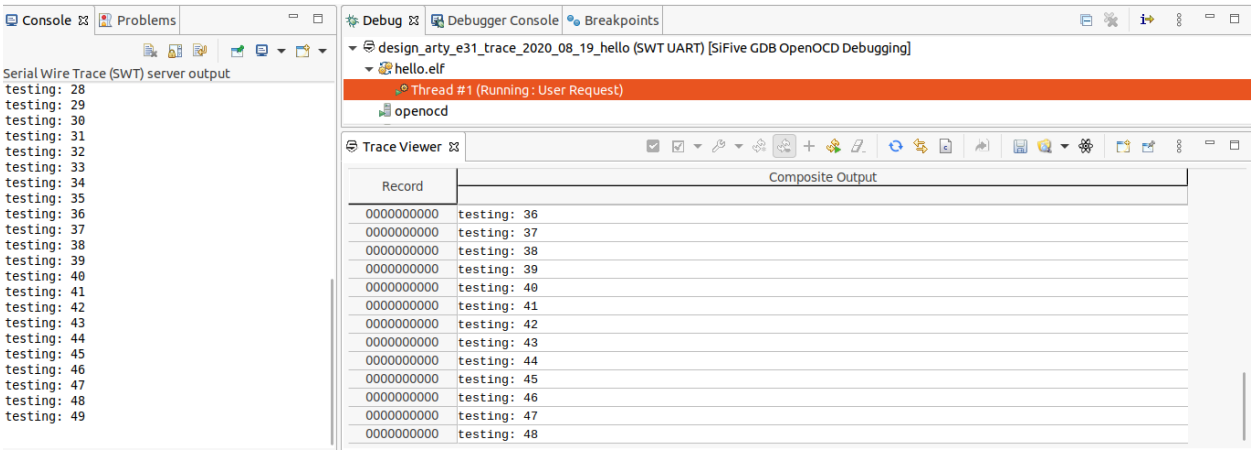


The stack frame column can be displayed with an indent or flat using the toolbar button.

SWT Output Console

When a trace session has been launched, and the trace configuration has been set to use Serial Wire Trace mode of the tracing hardware, then in Freedom Studio's Console tab, there is a console page present for displaying the runtime instrumentation trace output from the target. Refer to the trace decoder software package for more information on using instrumented-trace-based printf output (e.g. the itc_printf() function for which source code is provided by the trace-decoder package). Below is an example of a simple program using instrumentation trace printf, running on a SiFive core on an ARTY 100T board, and connected to the host running Freedom Studio using an inexpensive PL2303-based USB/serial adapter. Refer to the section on the Trace Control dialog box for more

information on setting up a serial-wire trace session. Note that when the Trace Viewer is open, then the SWT-acquired trace information is rendered there too.



Hardware Triggers

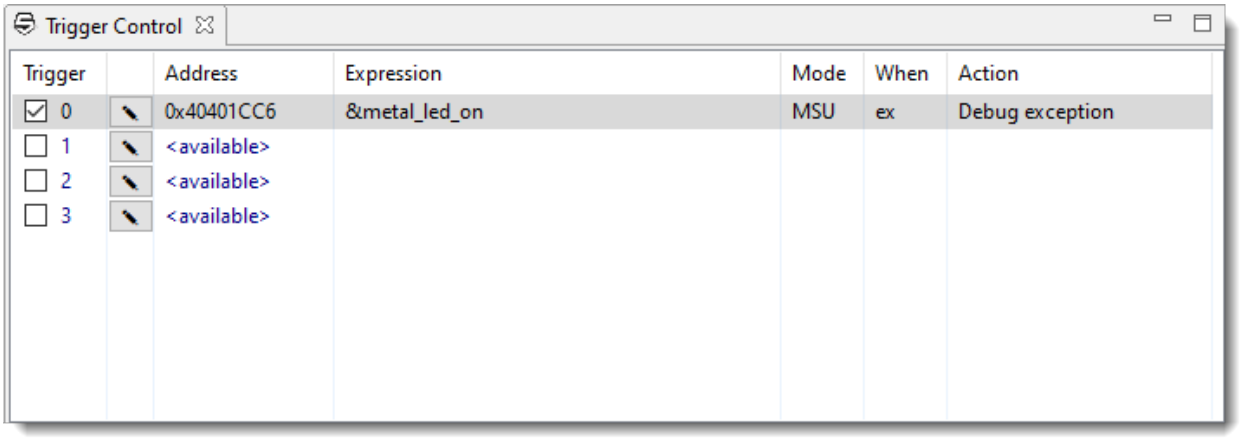
Hardware triggers are supported via three separate configuration views:

1. Trigger Control
2. Ext Trigger Inputs
3. Ext Trigger Outputs

Note: Earlier versions of Freedom Studio combined all three of these into a single view.

Trigger Control

The Trigger Control view allows configuration of target platform hardware address triggers. The view lists all address trigger resources available on the target.



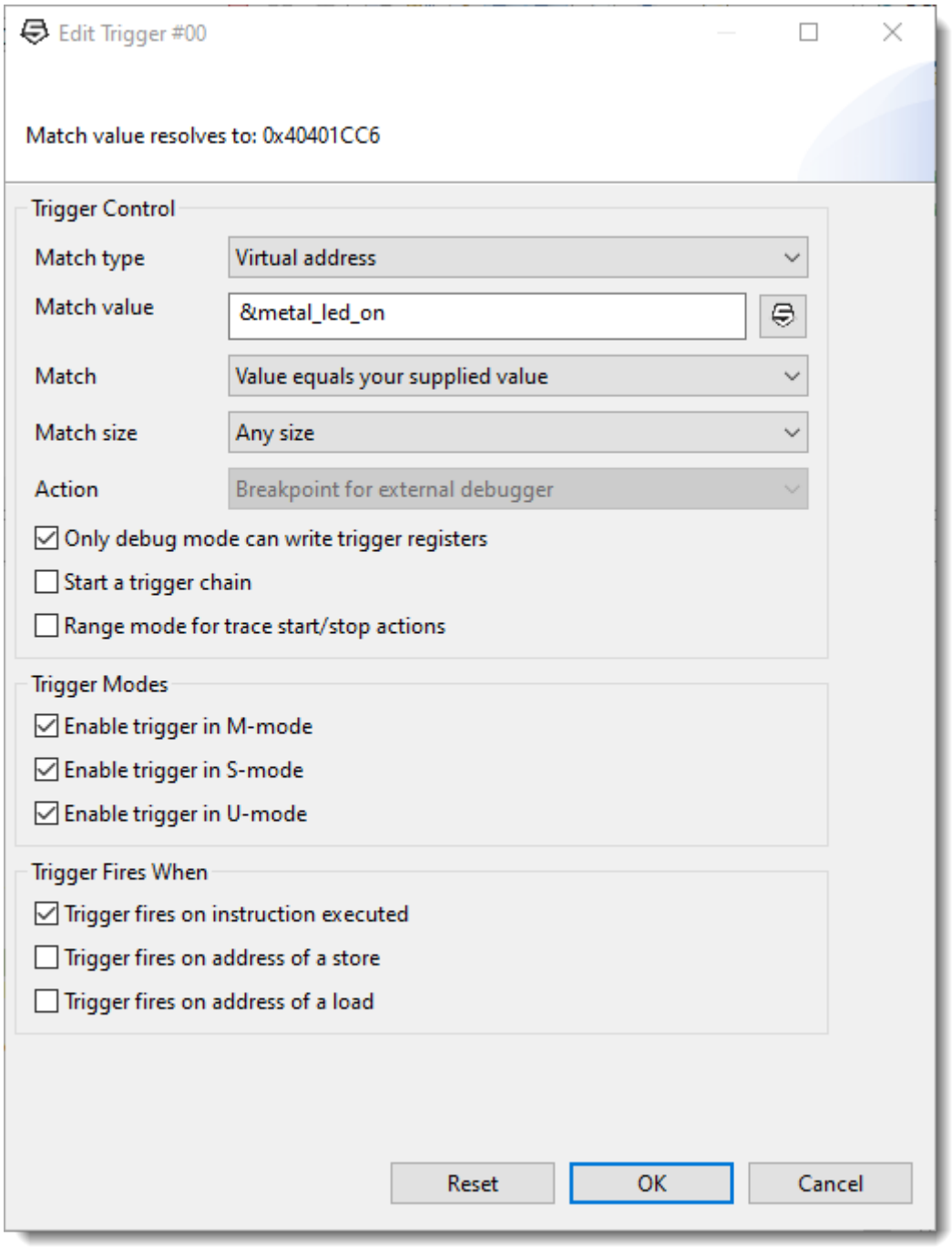
The Trigger Control view shows the current state of each trigger.

You can open the trigger configuration editor by:

- Clicking the pencil icon on the trigger you want to configure
- Double-clicking on the trigger you want to configure

Configuring Triggers

While simple hardware breakpoints are automatically supported via normal breakpoints, advanced configuration of hardware triggers must be done using the Trigger Configuration Dialog.



- **Match Type**
  Determines the type of match required for this trigger. Only options available on the target will be selectable. Options are:

    - **Virtual Address**
      A trigger is considered matched when the virtual address is matched.
    - **Data loaded/stored, or instruction executed**
      A trigger is considered matched when a load or store occurs, and the data address value being loaded or stored is equal to the match value. It is also considered matched when an instruction is executed, and the instruction address is equal to the match value.
- **Match Value**
  This is the address that must be matched to fire an address trigger. You can express the address in a few ways:
    - You can enter an expression. Freedom Studio will use gdb to evaluate the expression. The expression must evaluate to an instruction or data address.
    - You can use the Symbol Picker dialog (via the SiFive button) to find and select a symbol from the ELF file. Your selection from this dialog will be an expression (not a hardcoded address).
    - You can enter an absolute hex address
- **Match**
  This field determines how a match is made. Only options supported on the current target will be selectable. Options are:
    - Value equals your supplied value
    - Top M bits of value match those of your supplied value
    - Value is greater than or equal to your supplied value
    - Value is less than your supplied value
    - Lowerhalf(value) & upperhalf(yourvalue) == lowerhalf(yourvalue)
    - Upperhalf(value) & upperhalf(yourvalue) == lowerhalf(yourvalue)
- **Match Size**
  Determines the size of the match required to qualify the trigger. Only options available on the target will be selectable. Options are:
    - Any Size
    - 8 bit
    - 16 bit
    - 32 bit
    - 48 bit
    - 64 bit
    - 80 bit
    - 96 bit
    - 112 bit
    - 128 bit
- **Action**
  Determine what action is taken when this trigger fires. Only options supported on the current target will be selectable. Options are:
    - **Breakpoint for on-target debugger**
    - **Breakpoint for external debugger (i.e. OpenOCD, JLink)**
      Use this action for normal Freedom Studio debug session triggers
    - **Start trace**

    - **Stop trace**
    - **Record program trace Sync message**
    - **Generate an External Trigger Out**
- **Only debug mode can write trigger registers**
  Selecting this option reserves the trigger registers exclusively for the external debugger driving the Debug module; machine mode software will not be able to write to the trigger registers. This is the recommended option except for very specialized and uncommon cases (e.g., the software being debugged is a machine-mode debug agent).
- **Start a trigger chain**
  Form a chain between the current trigger with the trigger at the next highest trigger index, which itself may in turn be chained with its successor (although cores may limit maximum chain length, in which case the checkbox will refuse an attempt to 'check' it). The largest-indexed trigger in a chain is the trigger whose actions will be fired if and only if all triggers in the chain have met their match criteria.
- **Range mode for trace start/stop actions**
  Select this option to set up trace start/stop matched pair action across the range of addresses associated with this trigger (and predecessor if part of a chain) and specify "Start trace" as the trigger action. This scenario does not require an explicit "Stop trace" action, but rather the corresponding trace stop will be implicit when execution leaves the trigger match area. Another possibility is to leave this option unchecked and specify "start trace" and "stop trace" actions on separate triggers or chains, explicitly, although that will generally tie up more trigger slots.
- **Tigger Modes**
  Triggers can be configured to trigger only in the selected modes. Check the modes that you want to trigger to fire in.
- **Trigger fires on address or instruction executed**
  Fire the trigger if an instruction fetched from an address of interest, or with a specific encoding of interest, executes.
- **Trigger fires on address or data of a store**
  Fire the trigger if data is stored to an address of interest, or if a particular value of interest is being stored to any address.
- **Trigger fires on address or data of a load**
  Fire the trigger if data is loaded from an address of interest, or if a particular value of interest is being loaded from any address.
- **Clear**
  The [Clear] button simply clears the three mode checkboxes, effectively disabling the trigger.

Trigger Expressions

Trigger match addresses can be expressed using expressions. Expressions are evaluated using gdb's expression engine. Expressions are evaluated at launch, and when edited.

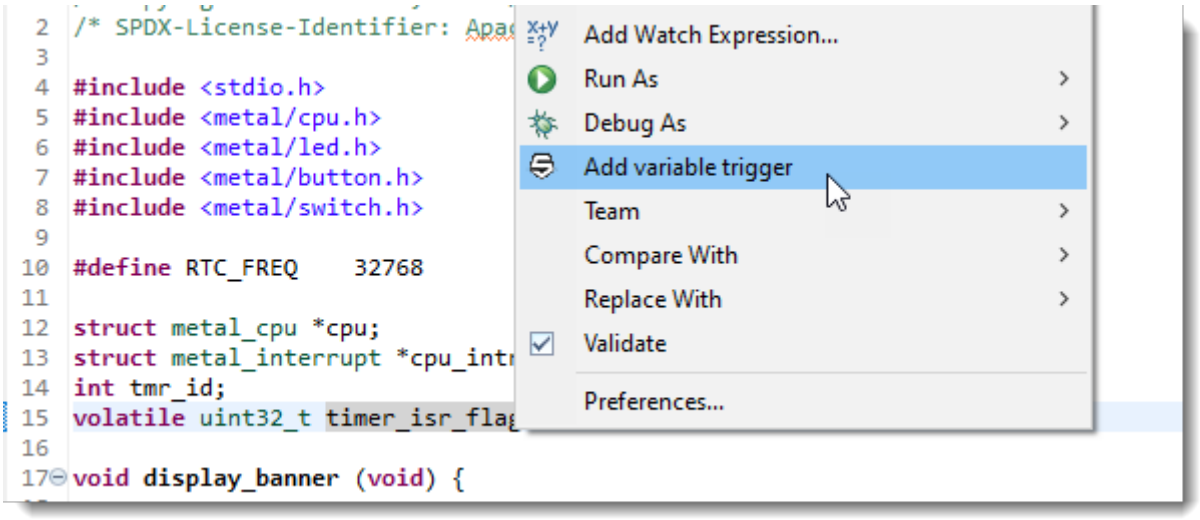
Expressions are preserved. Using expressions is the recommended way of expressing trigger addresses since the address may change from session to session as the code is edited and recompiled.

Expressions can be complex, allowing you to match on structure/union members, specific array elements, etc...

Expressions are evaluated dynamically while you edit the expression, providing feedback on the computed address and/or any errors that result in a failed evaluation.

Add Data Trigger from Editor

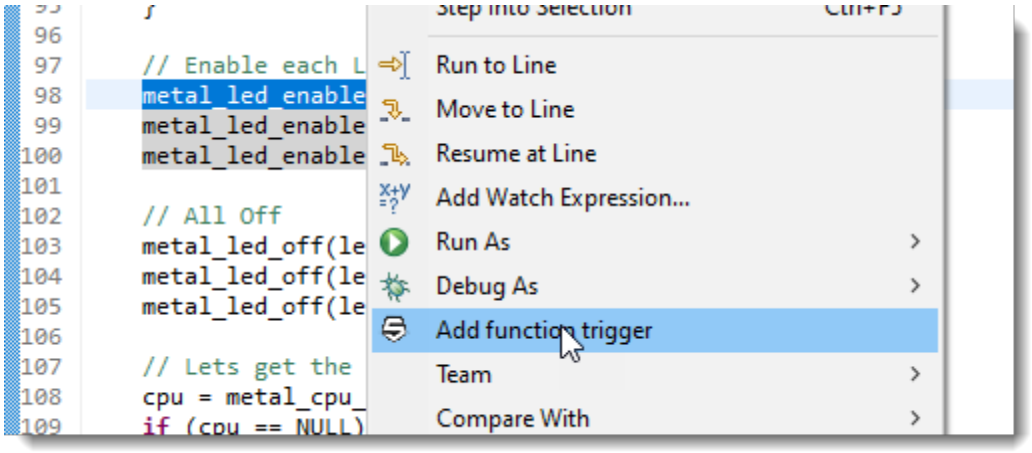
You can add a new data address trigger from the editor context menu. To do so, place the cursor on a data symbol (it must be a global or static symbol, stack variables are not eligible since the address may be different on every function call), open the context menu and select **Add Variable Trigger**



The trigger configuration dialog will open and be preconfigured for the selected symbol. Review the trigger settings before closing the dialog and installing the trigger.

Add Function Trigger from Editor

You can add a new function trigger from the editor context menu. To do so, place the cursor on a function name (a call, or definition will work), open the context menu and select **Add Function Trigger**



The trigger configuration dialog will open and be preconfigured for the selected symbol. Review the trigger settings before closing the dialog and installing the trigger.

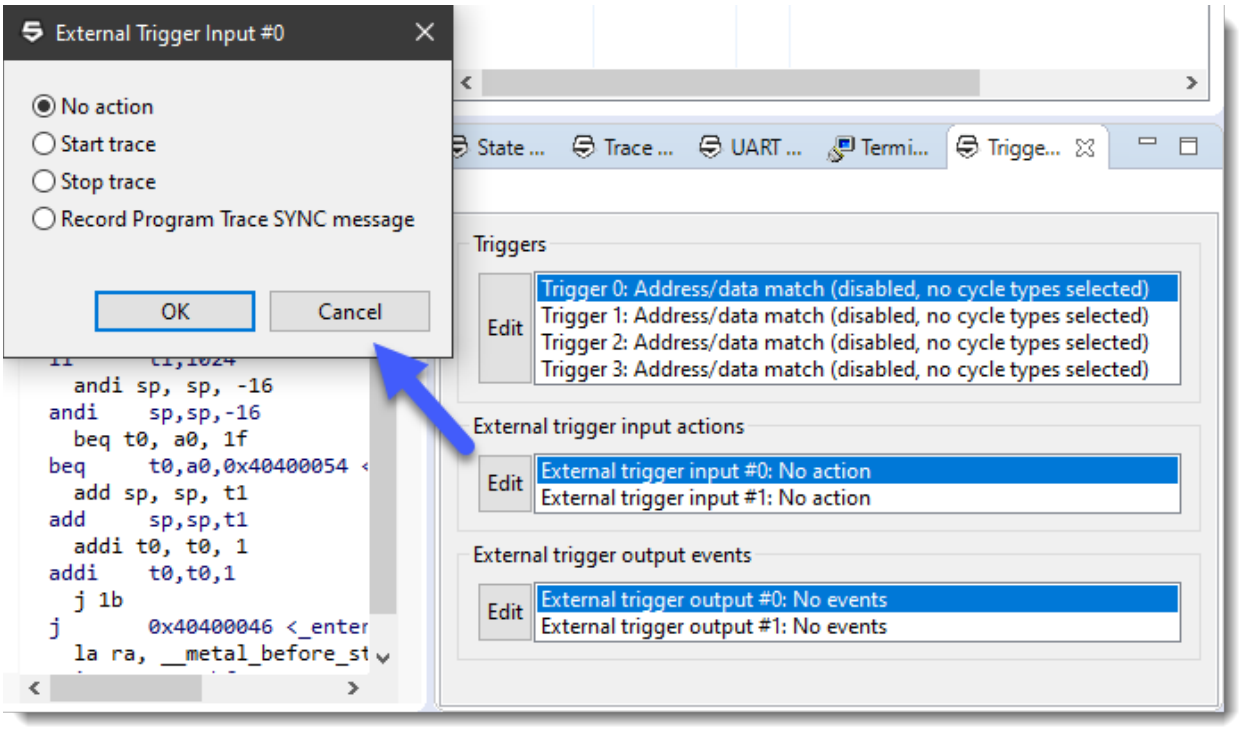
Add Expression Trigger from the Editor

You can add a new expression trigger from the editor context menu. To do so, select the text that will be the expression to evaluate (for instance, a reference to a structure member), open the context menu and select **Add Expression Trigger**

The trigger configuration dialog will open and be preconfigured for the selected expression. Review the expression evaluation result and trigger settings before closing the dialog and installing the trigger.

Configuring External Trigger Inputs

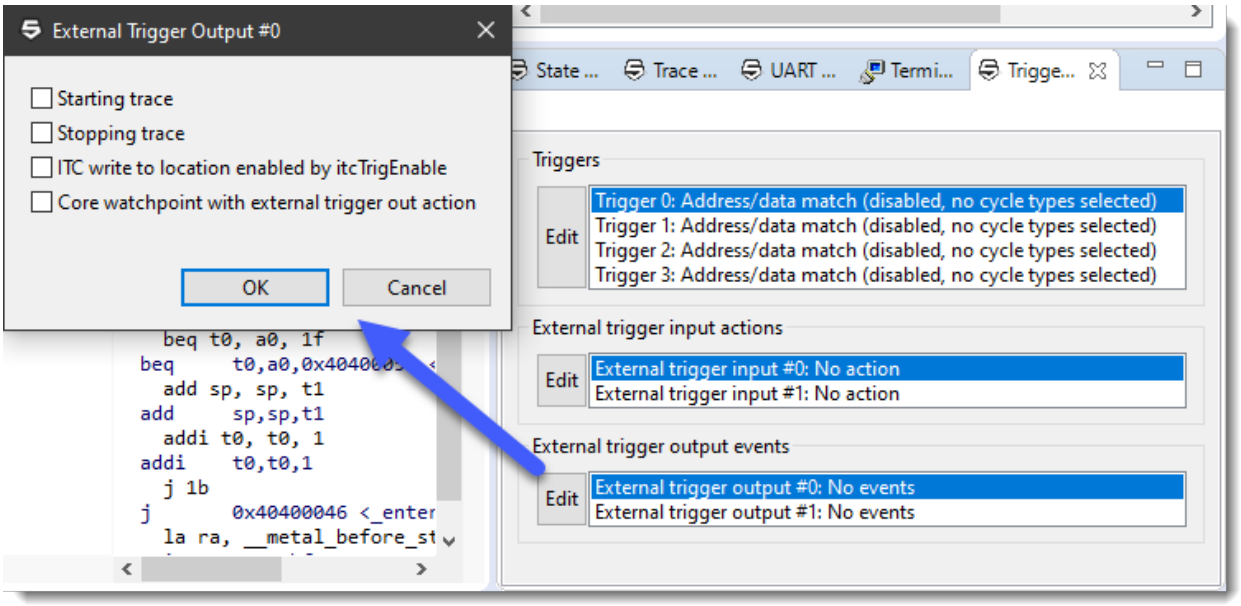
External trigger inputs (if present on the target design) can be configured to start or stop trace; or record a sync in trace output. When configuring an external trigger input you can select the desired action:



- **No action**
  No action is taken, effectively disabling this input signal.
- **Start trace**
  Start tracing when the input signal is asserted.
- **Stop trace**
  Stop tracing when the input signal is asserted.
- **Record Program Trace SYNC message**
  Records a SYNC message into the trace stream when the signal is asserted.

Configuring External Trigger Outputs

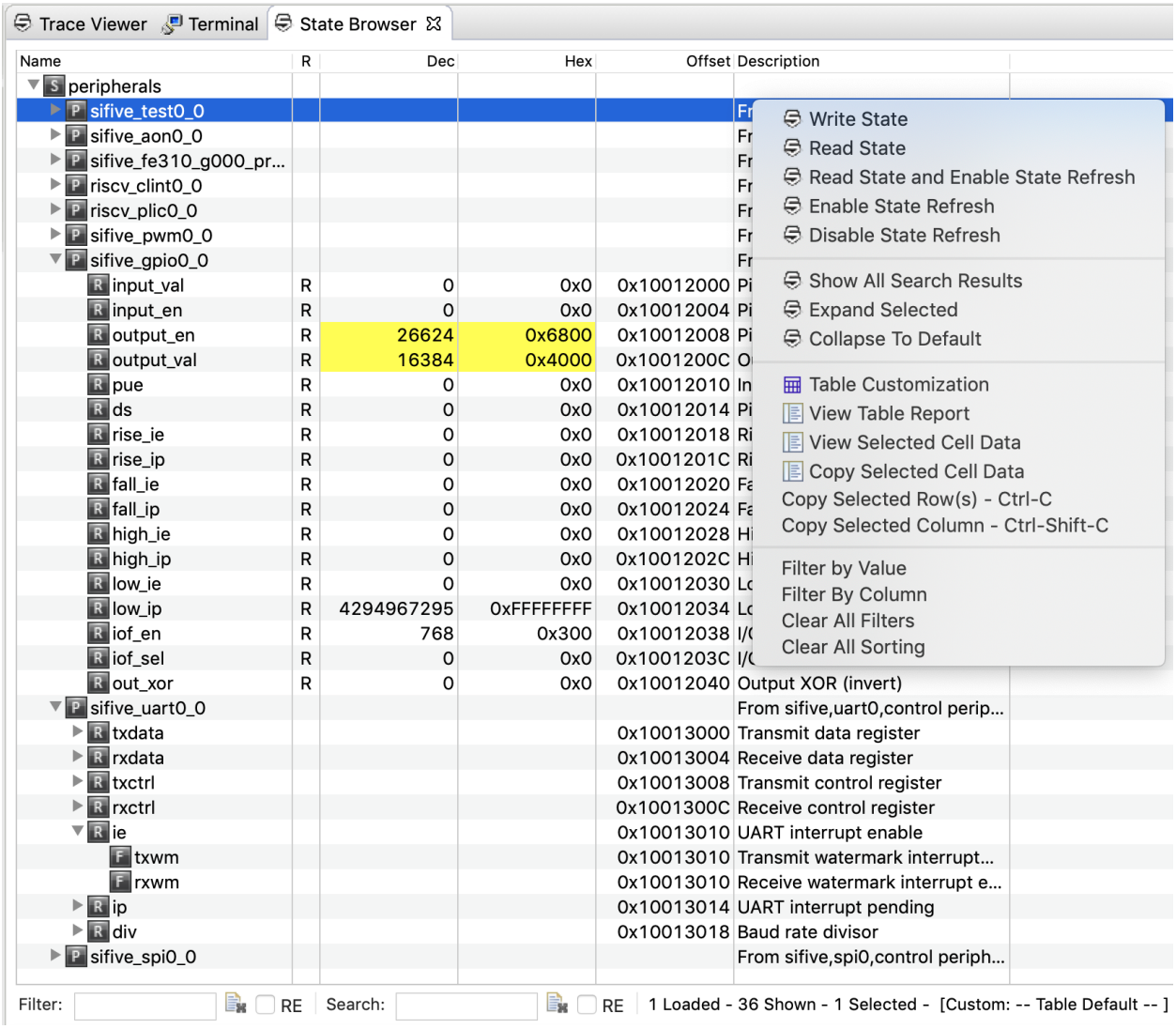
External trigger outputs (if present on the target design) can be configured to assert an external signal when specified conditions are met. The external trigger output dialog allows configuration of these conditions:



- **Starting Trace**
  This option asserts the external trigger signal when trace is started.
- **Stop Trace**
  This option asserts the external trigger signal when trace is stopped.
- **ITC write to location enabled by itcTrigEnable**
  itcTrigEnable is a 32-bit bitmap register with 1's in positions that correspond to ITC stimulus registers that you want to cause an external trigger out when written. For those stimulus registers that are enabled in this way, all external trigger-outs with this box checked will fire when that stimulus is written. You need to program both the source (itcTrigEnable) and sink (xtoControl) to actually generate a trigger.
- **Core watchpoint with external trigger out action**
  When a core watchpoint is hit, and programmed with the action “Generate an External Tigger Out” all triggers with this checkbox checked will fire.

State Browser

The State Browser provides a view for browsing the registers and register fields of all the peripherals in the system under debug and for reading and writing their state.



The contents of the State Browser is available when a system is under debugging, just like for most of the other views in Freedom Studio. The peripherals, their registers and their register fields are accessible through a tree structure with their name in the Name column, memory address in the Offset column and a human readable summary in the Description column. The state of an element is shown in decimal notation in the Dec column and in hexadecimal notation in the Hex column. The Dec and Hex column background turns yellow if the value has changed since the last state update of an element. The activation of state reading and writing is available through the popup menu on any of the elements:

- **Write State:** Rewrite the state shown in the Dec/Hex columns for the selected elements. If a peripheral node is selected then all it's registers are written.

- **Read State:** Read the state for the selected elements and show it in the Dec and Hex columns. If a peripheral node is selected then all it's registers are read.

The state of a register or register field can be updated by clicking in the Dec or Hex cell for it and then write in the new value and press return afterwards. The value in the Dec cell must be in decimal notation and the value in the Hex cell must be in hexadecimal notation (starting with 0x). Press ESC to revert to the old value in the cell.

Refreshing of Register State

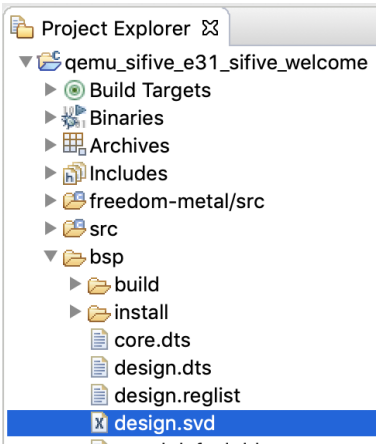
The state of all the registers are not by default automatically updated when the debugging session suspends, like it is in the Registers view. This is because some registers might have side effects when being read, just like the rxdata register in the uart or an interrupt pending register in some other peripherals. So in order to avoid this, it is up to the user to enable state refresh on the elements. The activation and deactivation of state refreshing is available through the popup menu on any of the elements:

- **Enable State Refresh:** Enable automatic refresh at suspend of state for the selected elements. If a peripheral node is selected it's enabled for all it's registers.
- **Disable State Refresh:** Disable automatic refresh at suspend of state for the selected elements. If a peripheral node is selected it's disabled for all it's registers.
- **Read State and Enable State Refresh:** This is a quick way to do a read state and enable automatic refresh at the same time.

If automatic refresh at suspend is enabled for a register it's shown by a 'R' in the R column. The refresh enablement is kept across debug sessions, but not between restarts of Freedom Studio. For more info on the other popup menu entries, the State Browser is based on https://www.eclipse.org/nebula/widgets/xviewer/xviewer.php

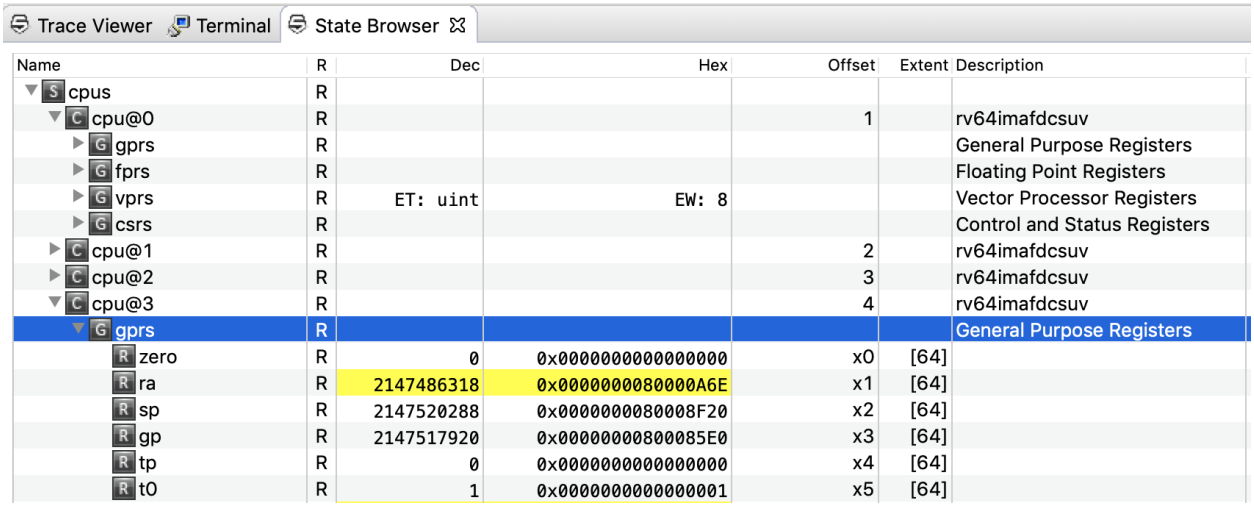
Design.SVD file

The State Browser gets all the description of registers and registers fields in the system from the design.svd file if it exists in the BSP. The format of the file is the standard CMSIS-SVD: http://www.keil.com/pack/doc/CMSIS/SVD/html/index.html



Accessing CPU Registers

In addition to the peripheral registers the State Browser also provides read and write access to the registers of all the CPU's of the system under debugging:



Name	R	Dec	Hex	Offset	Extent	Description
cpus	R					
cpu@0	R			1		rv64imafdcsuv
gprs	R					General Purpose Registers
fprs	R					Floating Point Registers
vprs	R	ET: uint	EW: 8			Vector Processor Registers
csrs	R					Control and Status Registers
cpu@1	R			2		rv64imafdcsuv
cpu@2	R			3		rv64imafdcsuv
cpu@3	R			4		rv64imafdcsuv
gprs	R					General Purpose Registers
zero	R	0	0x0000000000000000	x0	[64]	
ra	R	2147486318	0x0000000080000A6E	x1	[64]	
sp	R	2147520288	0x0000000080008F20	x2	[64]	
gp	R	2147517920	0x00000000800085E0	x3	[64]	
tp	R	0	0x0000000000000000	x4	[64]	
t0	R	1	0x0000000000000001	x5	[64]	

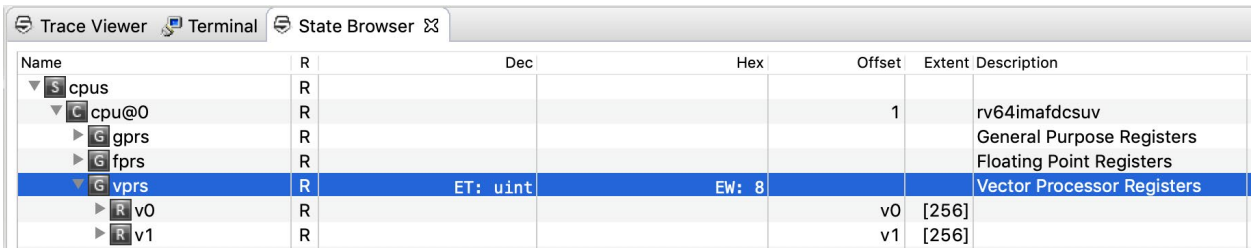
The available CPU registers is determined by the number of CPU's defined in the design.dts file and the ISA spec string of each of them:

- gprs: always available
  - x0-x31, pc
- fprs: available if the ISA string contains 'f' or 'd'
  - f0-f31, fflags, frm, fcsr
- vprs: available if the ISA string contains 'v'
  - v0-v31, vstart, vxsat, vxrm, vcsr, vl, vtype, vlenb
- cprs: always available
  - mstatus, misa, mie, mtvec, mscratch, mepc, mcause, mtval, mip, m*id

In contrast to the peripheral registers, the state of all CPU registers are automatically updated when the debugging session suspends.

Accessing Vector Registers

If a CPU has the vector extension then all the Vector Registers will be visible under the 'vprs' node - the 32 vector registers themselves and the Vector Extension CSR's. The 'vtype' CSR also has the subfields available: vlmul, vsew, vta, vma and vill.

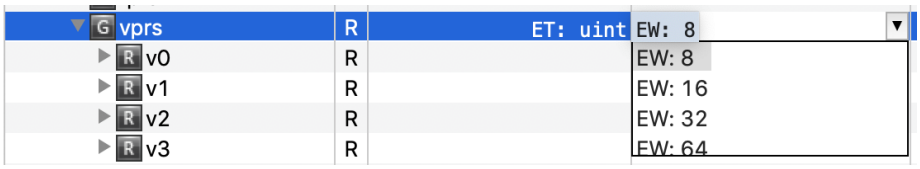


Name	R	Dec	Hex	Offset	Extent	Description
cpus	R					
cpu@0	R			1		rv64imafdcsuv
gprs	R					General Purpose Registers
fprs	R					Floating Point Registers
vprs	R	ET: uint	EW: 8			Vector Processor Registers
v0	R			v0	[256]	
v1	R			v1	[256]	

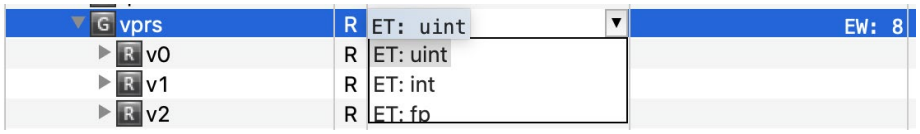
v30	R			v30	[256]	
v31	R			v31	[256]	
vstart	R	0	0x0000000000000000	0x008	[64]	Vector start position
vxsat	R	0	0x0000000000000000	0x009	[64]	Fixed-Point Saturate Flag
vxrm	R	0	0x0000000000000000	0x00A	[64]	Fixed-Point Rounding Mode
vcsr	R	0	0x0000000000000000	0x00F	[64]	Vector control and status register
vxsat	R	0	0x0000000000000000	0x00F	[0:0]	Fixed-point accrued saturatio...
vxrm	R	0	0x0000000000000000	0x00F	[2:1]	Fixed-point rounding mode
vl	R	0	0x0000000000000000	0xC20	[64]	Vector length
vtype	R	0	0x0000000000000000	0xC21	[64]	Vector data type register
vlmul	R	0	0x0000000000000000	0xC21	[2:0]	Vector register group multiplie...
vsew	R	0	0x0000000000000000	0xC21	[5:3]	Standard element width (SEW...
vta	R	0	0x0000000000000000	0xC21	[6:6]	Vector tail agnostic
vma	R	0	0x0000000000000000	0xC21	[7:7]	Vector mask agnostic
vill	R	0	0x0000000000000000	0xC21	[63:63]	Illegal value if set
vlenb	R	32	0x0000000000000020	0xC22	[64]	VLEN/8 (vector register length...

The Vector Register v0-v31 are special in the way that the interpretation of their content is not fixed in HW. Each register can be sliced into elements of a certain width and datatype. The interpretation shown in the State Browser can be selected by the user by clicking the Dec or Hex cell's of the 'vprs' node.

When clicking in the Hex cell, it is possible to change the element width interpretation:



When clicking in the Dec cell, it is possible to change the element data type interpretation (uint: Unsigned Integer, int: Signed Integer, fp: Floating Point, ascii: ASCII):



Vector Register Element Interpretation Examples

When the element width interpretation is changed, the number of sub elements of a vector register displayed in the State Browser changes accordingly.

When the element data type interpretation is changed, the value shown in the Dec column will correspond to both the data type and element width chosen.

Floating Point Element Datatype

vprs	R	ET: fp	EW: 64		
v0	R			v0	[256]
0	R	3.141	0x400920C49BA5E354	v0	[63:0]

Signed Integer Element Datatype

vprs	R	ET: int	EW: 16		
v0	R			v0	[256]
0	R	-42	0xFFD6	v0	[15:0]

8-bit Element Width

vprs	R	ET: uint	EW: 8		
v0	R			v0	[256]
0	R	0	0x00	v0	[7:0]
1	R	0	0x00	v0	[15:8]
2	R	0	0x00	v0	[23:16]
3	R	0	0x00	v0	[31:24]
4	R	0	0x00	v0	[39:32]
5	R	0	0x00	v0	[47:40]
6	R	0	0x00	v0	[55:48]
7	R	0	0x00	v0	[63:56]
8	R	0	0x00	v0	[71:64]
9	R	0	0x00	v0	[79:72]
10	R	0	0x00	v0	[87:80]
11	R	0	0x00	v0	[95:88]
12	R	0	0x00	v0	[103:96]
13	R	0	0x00	v0	[111:104]
14	R	0	0x00	v0	[119:112]
15	R	0	0x00	v0	[127:120]
16	R	0	0x00	v0	[135:128]
17	R	0	0x00	v0	[143:136]
18	R	0	0x00	v0	[151:144]
19	R	0	0x00	v0	[159:152]
20	R	0	0x00	v0	[167:160]
21	R	0	0x00	v0	[175:168]
22	R	0	0x00	v0	[183:176]
23	R	0	0x00	v0	[191:184]
24	R	0	0x00	v0	[199:192]
25	R	0	0x00	v0	[207:200]
26	R	0	0x00	v0	[215:208]
27	R	0	0x00	v0	[223:216]
28	R	0	0x00	v0	[231:224]
29	R	0	0x00	v0	[239:232]
30	R	0	0x00	v0	[247:240]
31	R	0	0x00	v0	[255:248]
v1	R			v1	[256]

16-bit Element Width

vprs	R	ET: uint	EW: 16		
v0	R			v0	[256]
0	R	0	0x0000	v0	[15:0]
1	R	0	0x0000	v0	[31:16]
2	R	0	0x0000	v0	[47:32]
3	R	0	0x0000	v0	[63:48]
4	R	0	0x0000	v0	[79:64]
5	R	0	0x0000	v0	[95:80]
6	R	0	0x0000	v0	[111:96]
7	R	0	0x0000	v0	[127:112]
8	R	0	0x0000	v0	[143:128]
9	R	0	0x0000	v0	[159:144]
10	R	0	0x0000	v0	[175:160]
11	R	0	0x0000	v0	[191:176]
12	R	0	0x0000	v0	[207:192]
13	R	0	0x0000	v0	[223:208]
14	R	0	0x0000	v0	[239:224]
15	R	0	0x0000	v0	[255:240]
v1	R			v1	[256]

32-bit Element Width

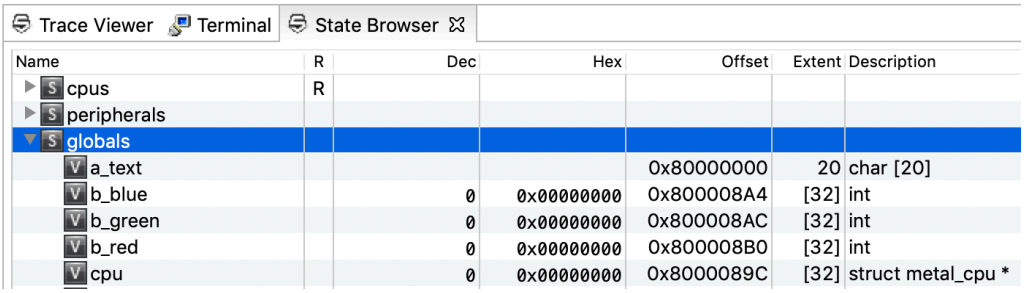
vprs	R	ET: uint	EW: 32		
v0	R			v0	[256]
0	R	0	0x00000000	v0	[31:0]
1	R	0	0x00000000	v0	[63:32]
2	R	0	0x00000000	v0	[95:64]
3	R	0	0x00000000	v0	[127:96]
4	R	0	0x00000000	v0	[159:128]
5	R	0	0x00000000	v0	[191:160]
6	R	0	0x00000000	v0	[223:192]
7	R	0	0x00000000	v0	[255:224]
v1	R			v1	[256]

64-bit Element Width

vprs	R	ET: uint	EW: 64		
v0	R			v0	[256]
0	R	0	0x0000000000000000	v0	[63:0]
1	R	0	0x0000000000000000	v0	[127:64]
2	R	0	0x0000000000000000	v0	[191:128]
3	R	0	0x0000000000000000	v0	[255:192]
v1	R			v1	[256]

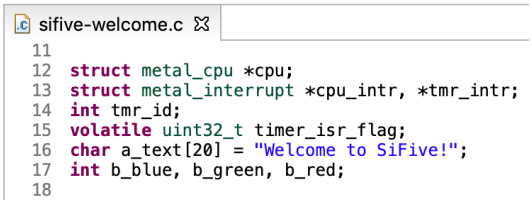
Accessing Global Variables

In addition to the peripheral and CPU registers the State Browser also provides basic read and write access to the global variables (plus live watch) of the system under debugging:



Name	R	Dec	Hex	Offset	Extent	Description
cpus	R					
peripherals						
globals						
a_text				0x80000000	20	char [20]
b_blue		0	0x00000000	0x800008A4	[32]	int
b_green		0	0x00000000	0x800008AC	[32]	int
b_red		0	0x00000000	0x800008B0	[32]	int
cpu		0	0x00000000	0x8000089C	[32]	struct metal_cpu *

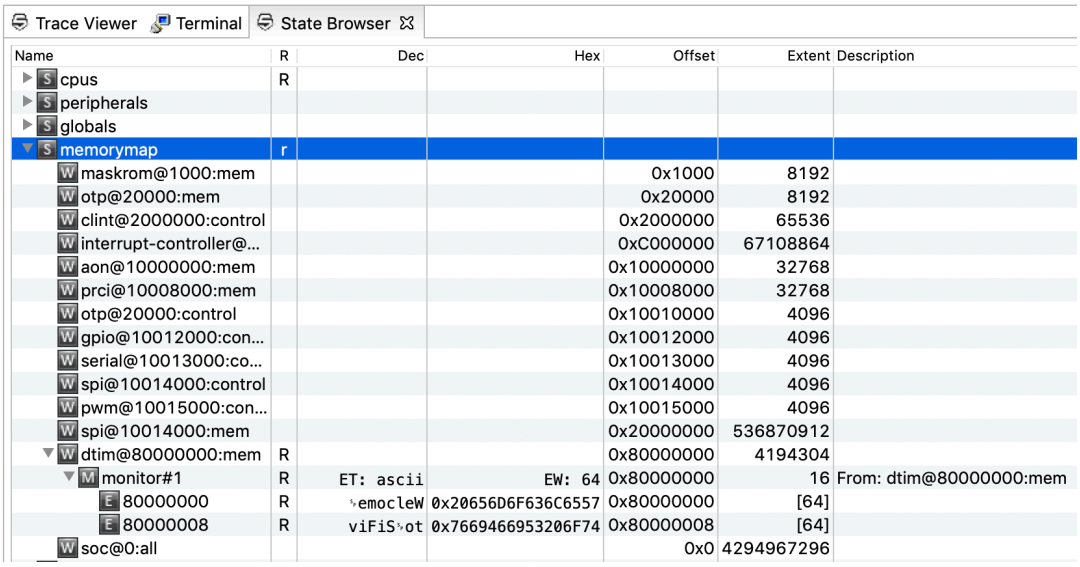
The global variables originates from the C code with address and type as returned from GDB:



Reading and setting the values of global variables works in the same way as for the Peripheral Registers. Global Variables are by default not setup to refresh when the target suspends to keep GUI response time good, and it is usually only a subset of variables that are of interest. Refresh can be controlled in the same way as for the Peripheral Registers.

Creating Memory Monitors

In addition to the peripheral and CPU registers and global variables the State Browser also supports creating memory monitors (plus live watch) for the system under debugging:

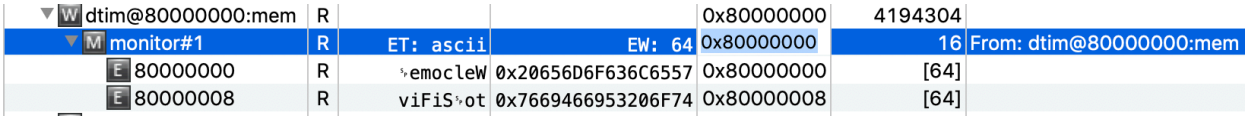


Name	R	Dec	Hex	Offset	Extent	Description
cpus	R					
peripherals						
globals						
memorymap	r					
maskrom@1000:mem				0x1000	8192	
otp@20000:mem				0x20000	8192	
clint@2000000:control				0x2000000	65536	
interrupt-controller@...				0xC000000	67108864	
aon@10000000:mem				0x10000000	32768	
prci@10008000:mem				0x10008000	32768	
otp@20000:control				0x10010000	4096	
gpio@10012000:con...				0x10012000	4096	
serial@10013000:co...				0x10013000	4096	
spi@10014000:control				0x10014000	4096	
pwm@10015000:con...				0x10015000	4096	
spi@10014000:mem				0x20000000	536870912	
dtim@80000000:mem	R			0x80000000	4194304	
monitor#1	R	ET: ascii	EW: 64	0x80000000	16	From: dtim@80000000:mem
80000000	R	␣emocleW	0x20656D6F636C6557	0x80000000	[64]	
80000008	R	viFiS␣ot	0x7669466953206F74	0x80000008	[64]	
soc@0:all				0x0	4294967296	

When starting a new project and debug session the State Browser does not have any predefined memory monitors, but it has the memory map of the target. The memory map consists of the memory ranges for all the peripherals and memories in the system, including an 'soc' node that represents the entire memory space.

Creating a new memory monitor is done by selecting the 'Add Memory Monitor' entry from the popup menu on a memory range element. The memory monitor is created with the start address of the memory range from which it was created and covering 16 bytes.

The start address of the memory monitor can be changed by editing the value in the Offset column of the memory monitor node. The size of the memory monitor can also be changed by editing the value in the Extent column of the memory monitor node:



The start address and size of the memory monitor is in bytes and is forced to be a multiple of 8!

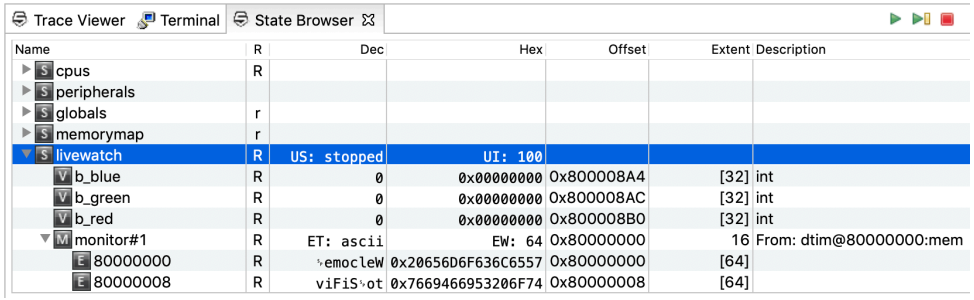
The presentation of the data covered by a memory monitor can be controlled in the same way as for vector registers through selecting the presentation width or the presentation datatype. The presentation shown in the State Browser can be selected by the user by clicking the Dec or Hex cell's of the memory map node.

Memory monitors can also be removed again by selecting the 'Remove Memory Monitor' entry from the popup menu on a memory monitor element.

Mastering Live Watch

Live Watch is a feature that allows continuously watching the values of selected global variables or memory monitors while the target is running or even when it is suspended. The rate at which the elements are read from the target can be controlled by selecting the State Browser view menu entry 'Set Live Watch Update Interval' and entering a new value in the resolution of ms.

Selected global variables and memory monitors can be added to Live Watch by selecting the 'Add To Live Watch' on those type of nodes. When an element is added it is shown as a child of the 'livewatch' node in the State Browser. Elements under Live Watch can be removed from Live Watch by selecting the 'Remove From Live Watch' on a child node of the 'livewatch' node.



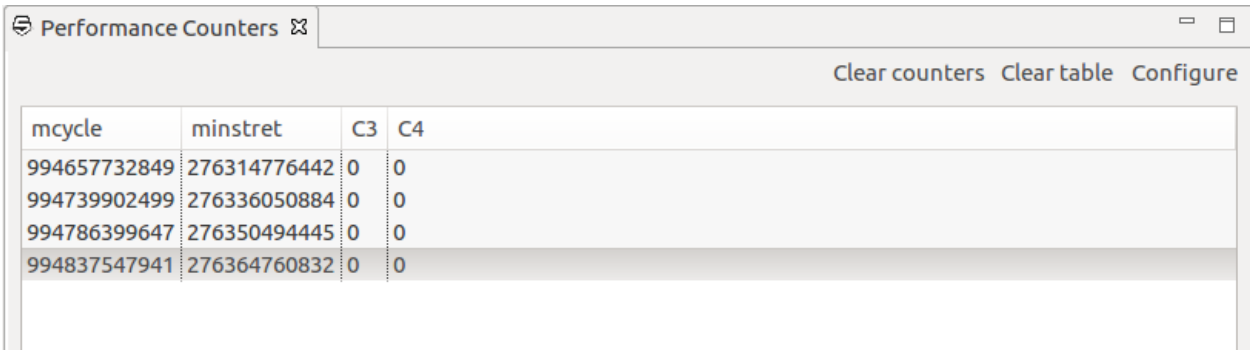
After having added elements to Live Watch it can be started by selecting the ‘Start Nonstop Live Watch’ that will also run while the target is suspended or by selecting the ‘Start Pausing Live Watch’ that will only run while the target is running and pause when the target is suspended.

Live Watch can be stopped again by selecting the ‘Stop Live Watch’. All three actions are available from the State Browser toolbar and popup menu. The mode of Live Watch is presented in the Dec column field of the ‘livewatch’ node and the update interval is presented in the Hex column field.

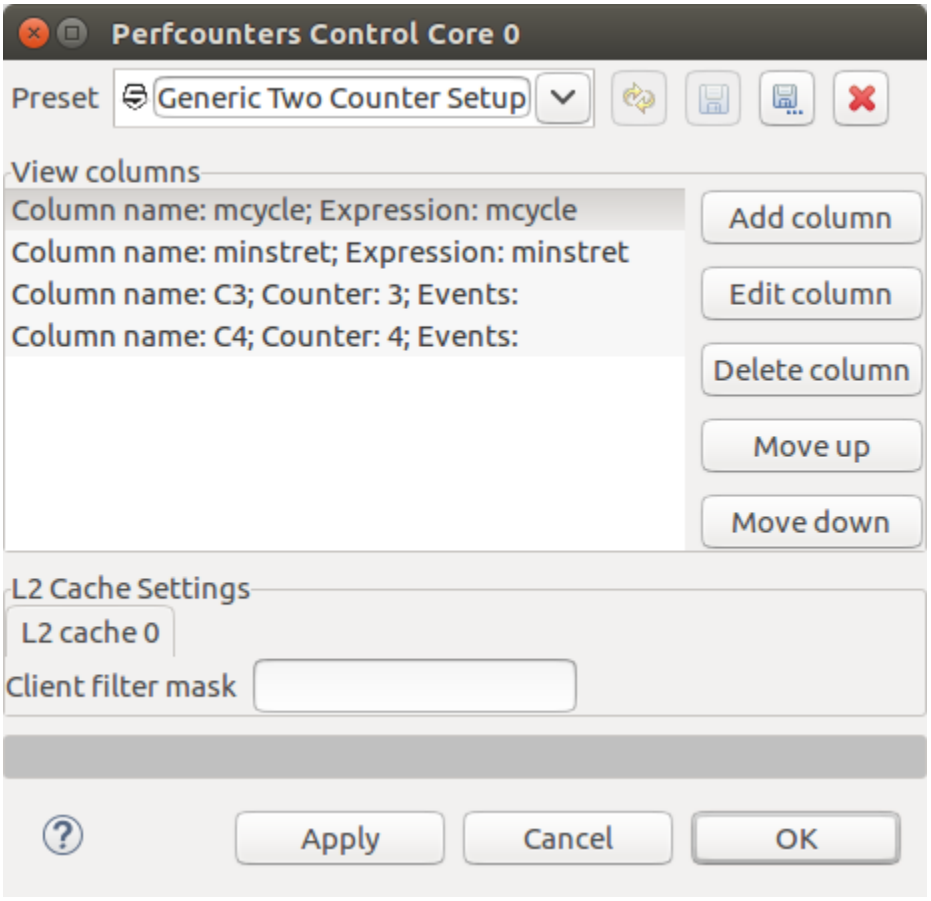
Remember that the Live Watch feature is only available if the target is QEMU or a system that has the Debug Module SBA (System Bus Access) hardware feature available.

Performance Counters View

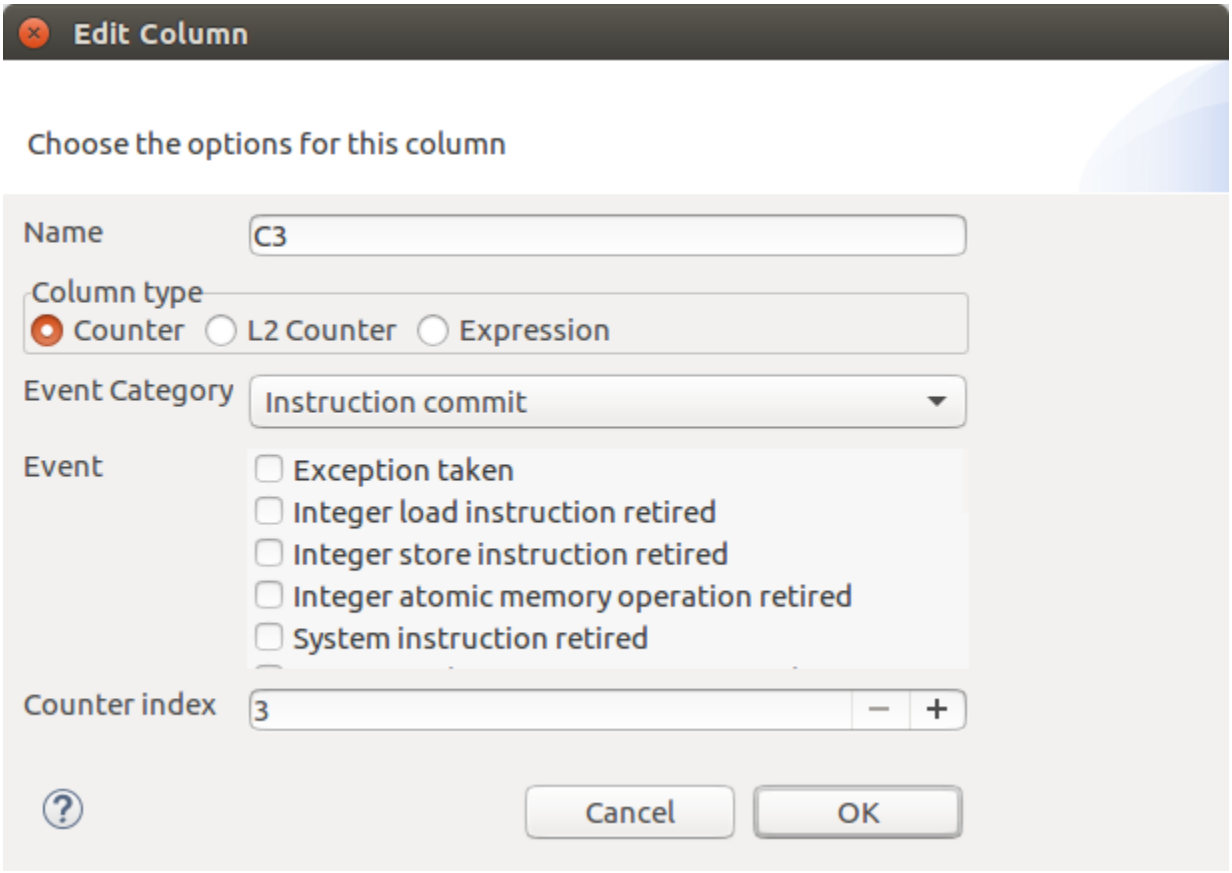
The Performance Counters View displays a history of selected performance counter values, and optionally displays arithmetic expressions based on counter values. The term performance counter can refer to either the performance counters on a SiFive RISC-V core, or performance counters associated with SiFive composable L2 cache instances. At this time, performance counter events are not architecturally standardized throughout the RISC-V ecosystem, and this view is specifically designed to work with SiFive cores. Selected counter values and/or expressions are updated each time processor execution is halted during a debugging session. In an SMP debug session with multiple RISC-V harts, any RISC-V counters reflect the currently highlighted hart in the Debug view; highlighting a different hart in the Debug view will cause any core counters to reflect the newly highlighted hart's counter values. Any L2 cache counter values are independent of the currently highlighted core, since L2 caches exist outside of cores. The performance counters view can be opened from Freedom Studio's main menu: Window -> Show View -> Performance Counters.



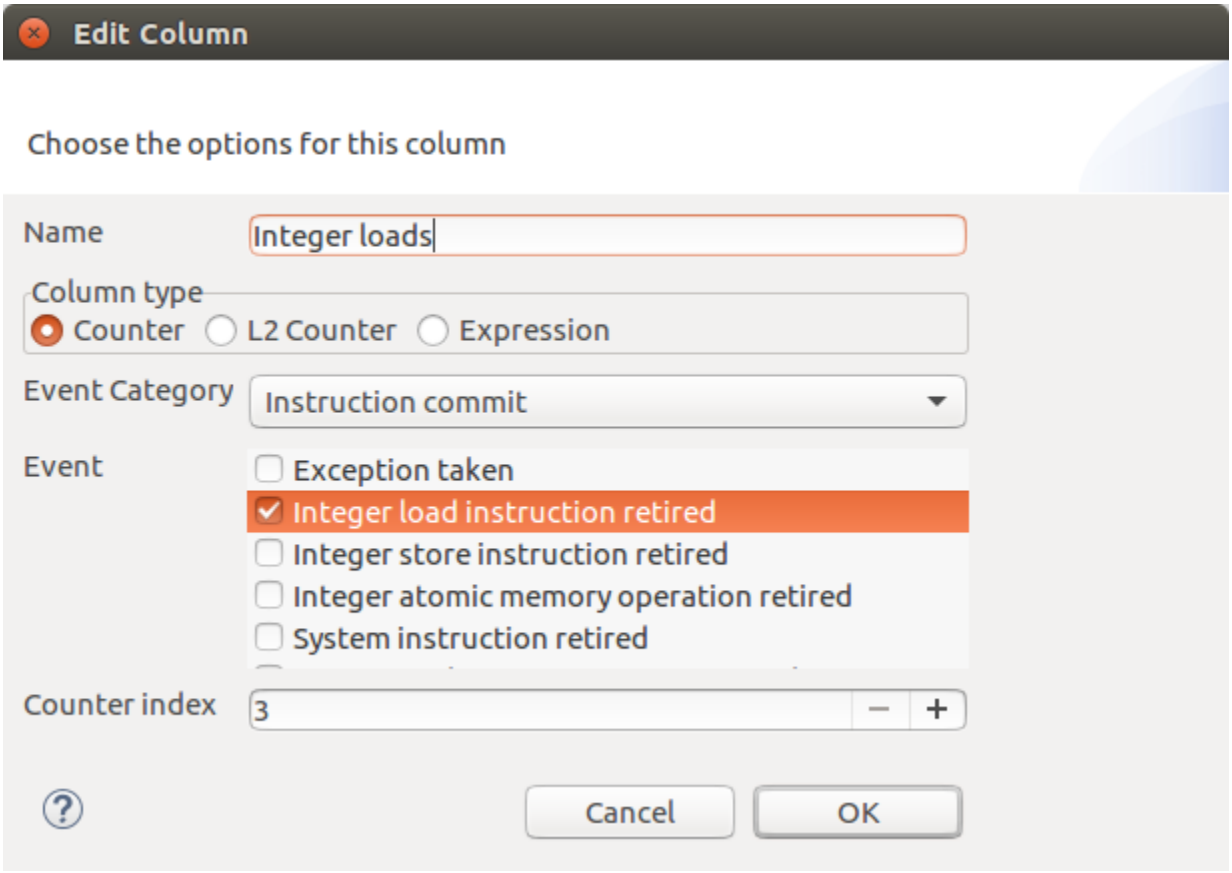
There is a generic default performance counter view setup that includes mcycle, minstret, mhpmcounter3, and mhpmcounter4 (abbreviated to C3 and C4). Performance counter view setups can also be customized and saved as presets. To customize a performance counter view setup, click the "Configure" menu button in the upper right section of the view. This brings up a dialog box that allows columns to be defined, edited, ordered, and deleted (see image below).



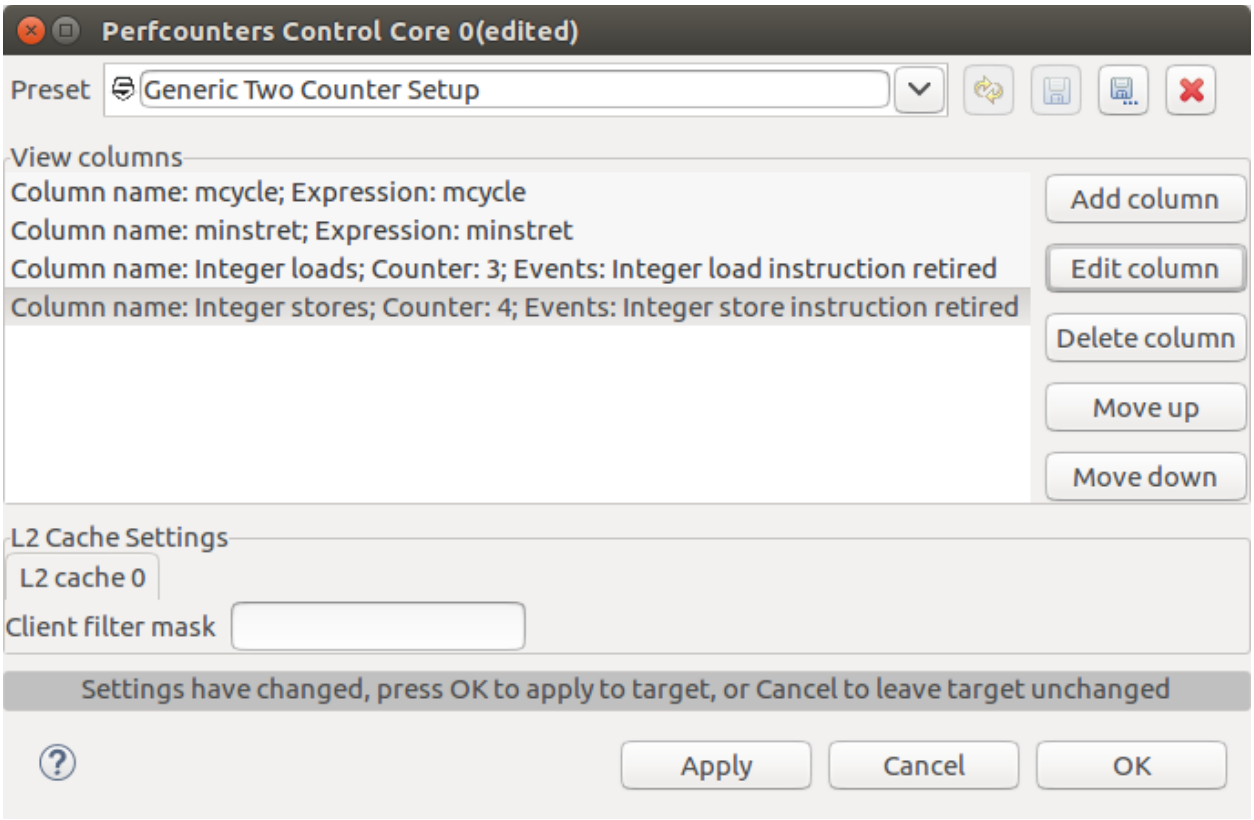
A column can be associated with either a specific single selectable performance counter, or a textual expression that references one or more performance counters. For the sake of example, let's go through the steps involved in creating a customized performance counter preset, based on the "Generic Two Counter Setup" that is included by default. Firstly, let's give the 3rd column a more descriptive name, and choose an event to bind to the counter. To do that, click on the row with column name "C3" and press the "Edit column" button.



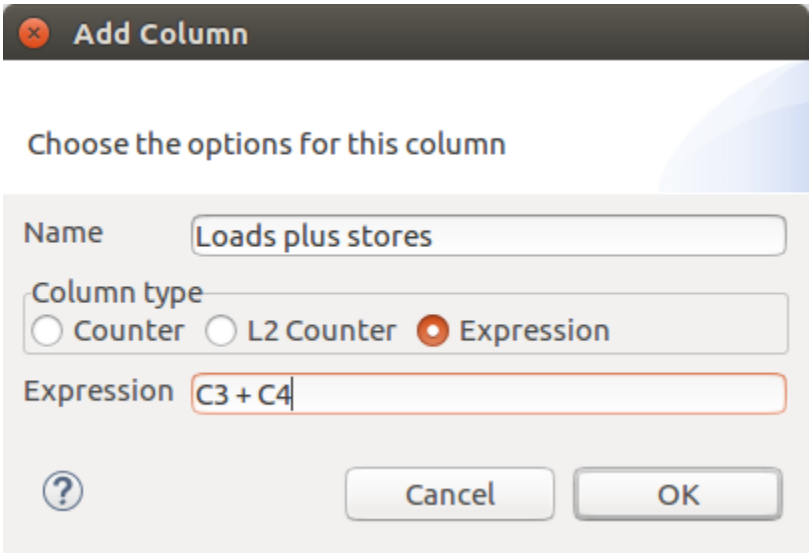
The event categories, and the events within those categories, may differ, depending on the particular SiFive core family that is on the target hardware. Please consult the manual for your particular core, to learn more about the available events, and what they mean. The same recommendation applies to the performance events for the L2 cache. Let's check the box "Integer store instruction retired" and change the column name to "Integer loads".



Then press the OK button, and do the same sort of edit for the column “C4”, except this time check the “Integer store instruction retired” box and assign a name “Integer stores”. At that point, we should see:



Now, let's add an expression column that presents the sum of integer loads and stores. Press the "Add column" button then make the appropriate entries and selections as shown below.



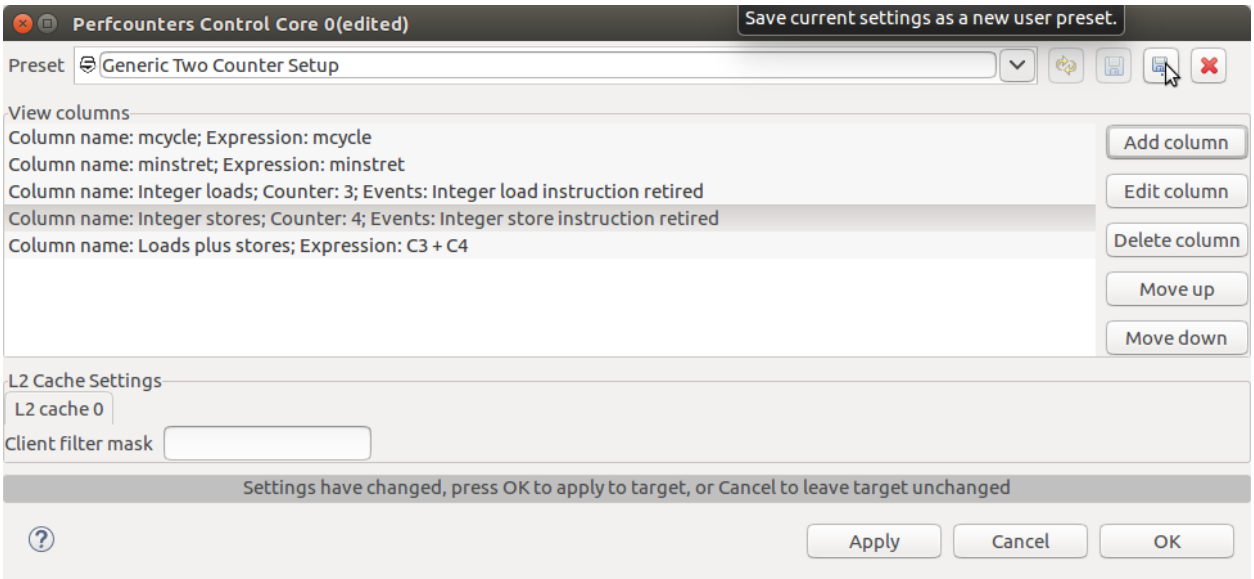
Expressions can reference RISC-V counters by using pseudo-variables "mcycle", "minstret", and "C<idx>" (where <idx> is an integer value greater than or equal to 3 and less than or

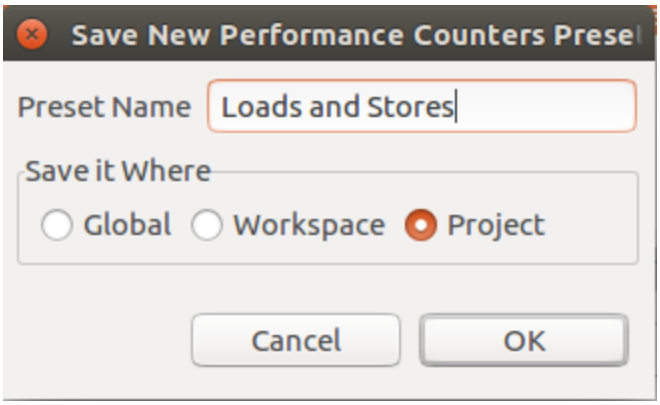
equal to the highest indexed performance counter on the core, which will be 31 or less). Equivalent to "C3" would also be "Core.C(3)" (the latter syntax is more consistent with the L2 performance counter expressions). L2 cache performance counters are referenced using the syntax `L2.C(zero_based_L2_cache_index, zero_based_counter_index);` for instance: "L2.C(0, 0)" would reference performance counter 0 of L2 cache 0. Expressions are evaluated using the Apache Jexl library, and theoretically the full Jexl expression language could be used in this context, but the simple arithmetic expression subset of Jexl can be useful in itself; using conventional infix arithmetic expression syntax to join counter references should just work, without needing to look up more advanced Jexl constructs. For calculations that include division, consider the divide-by-zero possibility to avoid "<evaluation error>" appearing in a performance counter cell later.

For advanced usage of expressions in this context, beyond simple/intuitive arithmetic syntax, please refer to https://commons.apache.org/proper/commons-jexl/reference/syntax.html

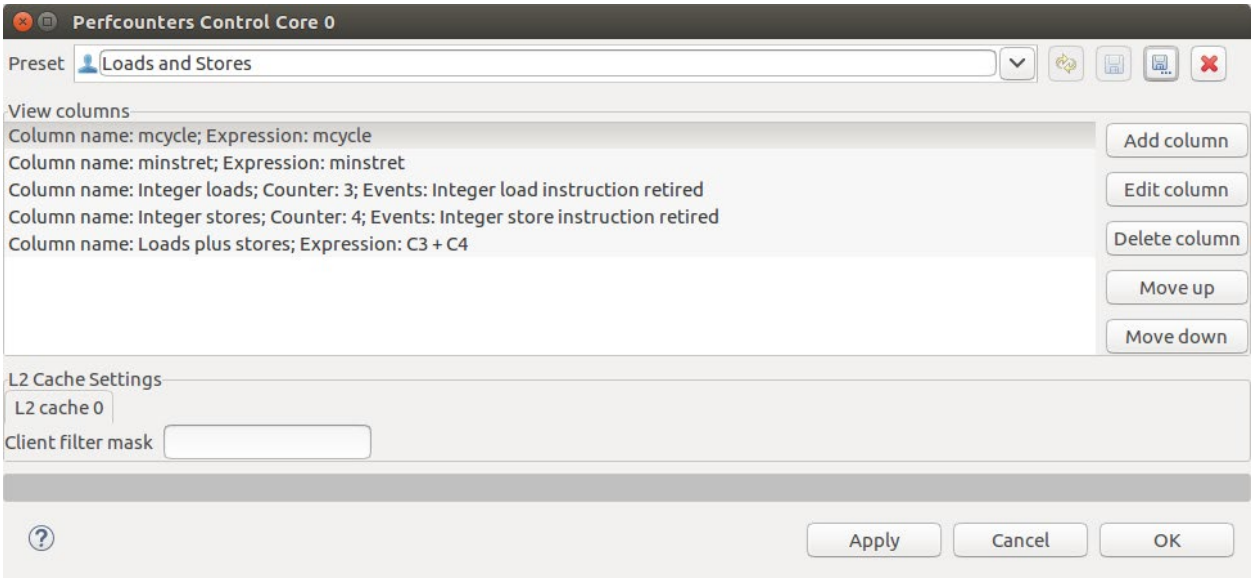
Note that Freedom Studio doesn't keep a stateful Jexl engine running persistently. It instantiates Jexl engines ephemerally as needed to evaluate any expression columns after halting, so trying to influence multiple performance counter columns through any Jexl global variables or state (other than the pseudo-variables mentioned above) probably won't have the desired effect. Sticking to the defined pseudo-variables, and arithmetic expressions based on those, will provide best results at this time.

To save the edited performance counter setup as a new preset, click on the toolbar button as shown below, and in the resulting dialog box supply the desired name of the new preset.

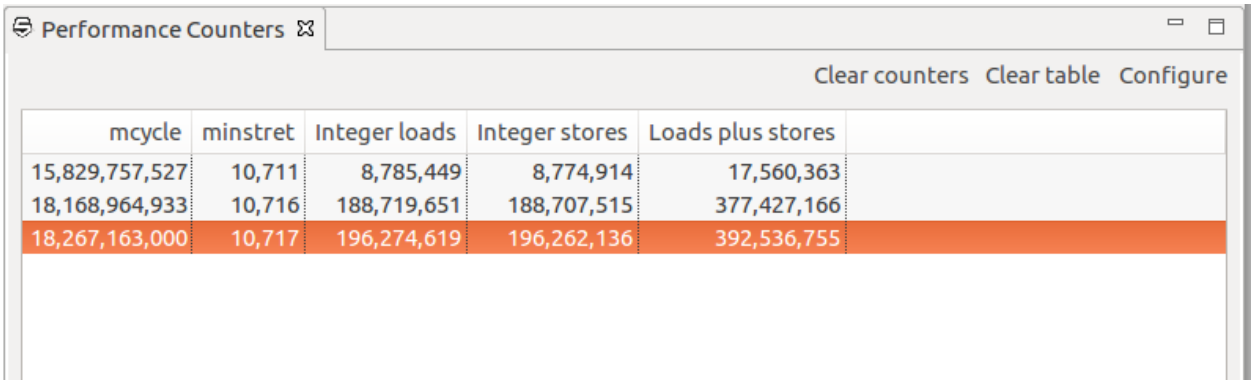




Pressing the OK button of this secondary dialog box brings us back to this:

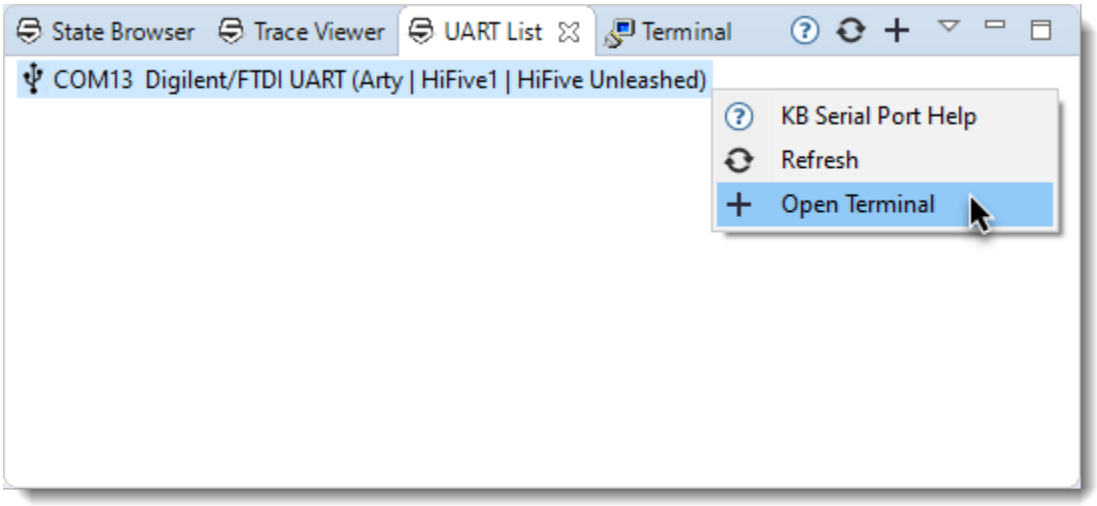


Pressing the OK button of this top-level dialog box applies the settings to the corresponding hart on the attached hardware target, and updates the columns of the performance counter view, which will then append rows of values each time the hart halts during the debug session:



UART List View

The UART List View shows a list of all serial ports (virtual and real) on the host system and will identify the correct serial ports for connected targets.



This view does not automatically refresh the list. Refreshes must be done manually. After connecting a target cable, press the use the "refresh" command to refresh the list.

Once the desired serial port is shown, you can open a serial terminal on the port by using the "Open Terminal" command.

You can also configure a debug launch to automatically open the terminal.

This feature works by running custom shell scripts for each host platform that know how to examine the system serial ports, extracting device ID information from the device. The script uses the device IDs for commonly supported target platforms and can thus identify which port belongs to which target.

Execution Profiler View (PC Sampling)

The Execution Profiler View is used to display PC Sampling data. PC Sampling is a technique where the Program Counter is sampled at a high frequency, processed, then displayed using histograms that can help identify program hotspots.

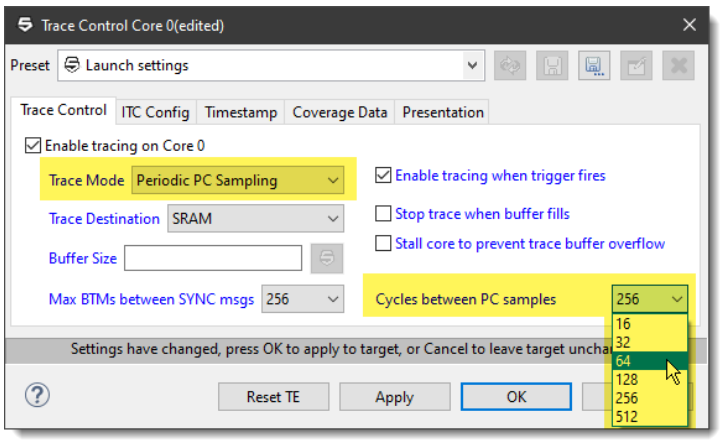
PC Sampling data can be collected in two ways:

1. **Trace-based PC Sampling**
   This method uses the target trace encoder to "sample" the PC every X number of cycles (where X is configurable in the Trace Configuration Dialog). Sample data is sent to a trace sink (SRAM or System Memory) on the target, then uploaded and processed *when the target halts*. This sampling method is very systematic in that samples are collected every X number of cycles. The amount of PC data collected is dependent on the size of the trace sink.
2. **OpenOCD-based PC Sampling**
   This method requires a newer version of OpenOCD that supports a risc-v PC sampling extension. This sampling method is less systematic than trace-based sampling in that samples are collected by OpenOCD using a polling system. The polling rate and target cycle rate are completely different and unrelated. The polling loop timing can also be impacted by "other" work being done in the OpenOCD process. OpenOCD base sampling can be collected continuously *while the target is running*. OpenOCD sampling collects approximately 20K samples/sec with a JTAG clock rate of 29Mhz.

Both types of PC sampling require a trace-encoder on the target system.

Trace-based PC Sampling

Trace-based PC sampling is configured using the Trace Configuration dialog accessed via the Trace Data Viewer.



Use the Trace Mode dropdown to select "Periodic PC Sampling". When you select this, the control labeled "Max I-CNT between BTM messages" will change to "Cycles between PC Samples". Use this dropdown to select the sampling frequency.

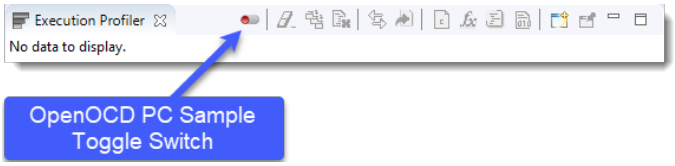
Note: Using a System Memory sink and choosing a very high frequency (i.e. a low number of cycles) can impact target performance because storing the samples to the trace sink requires transactions on the system memory bus that compete with normal program accesses.

Trace-based PC sampling is processed when the target halts.

OpenOCD-based PC Sampling

OpenOCD based PC Sampling requires no configuration. It is enabled and available as long as the target has a trace-encoder (the trace-encoder provides the memory mapped PC sample register used by OpenOCD to read the PC while the target is running) and the version of OpenOCD being used includes the risc-v sampling feature.

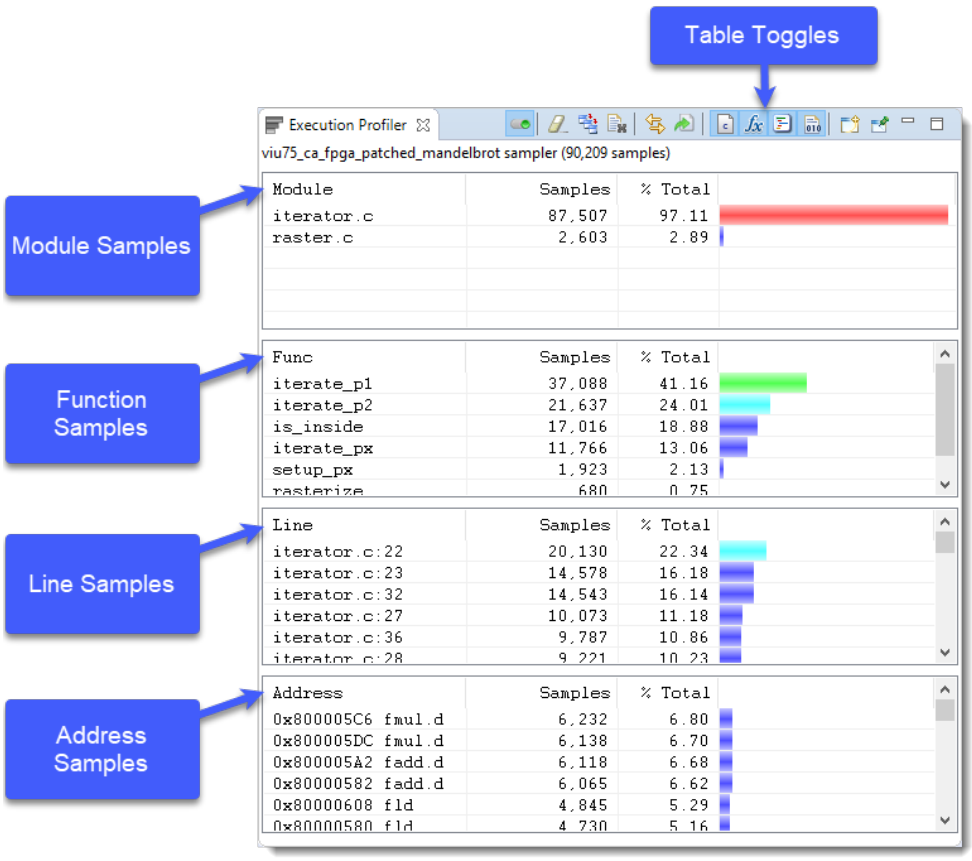
Collecting, processing, and displaying OpenOCD PC Sample data is controlled by s simple toggle switch in the Execution Profiler View:



To collect OpenOCD-based samples, toggle the enable switch on (green). You can leave the switch on while the target is halted, no data will be collected while the target is halted, but will resume collecting when the target is resumed.

Displaying PC Sampling Data

PC Sampling data is displayed in the Execution Profiler View as histogram data. PC Sampling data is further binned into Instruction level counts, Source Line level counts, Function level counts, and Module level counts. Each level of counting is displayed in a separate table.



Individual tables can be toggled on or off using the Table Toggle buttons on the viewer toolbar. Each table contains a list of the sample bins sorted by highest sample count. For each bin, the total number of samples, and the percentage this bin represents against all samples is shown. Hot-spots in code may be identified by those bins that represent abnormally high execution counts. Of course, this is not definitive, and understanding what the code is doing and why it is doing it is critical to using this data to identify hot-spots.

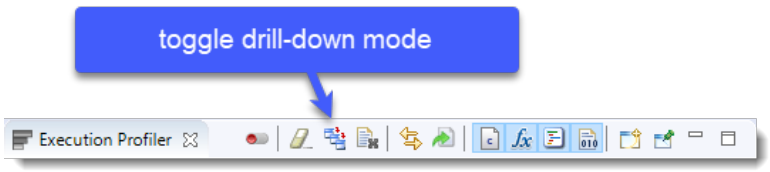
The Address table also shows the instruction at the address to assist in helping understand why a specific address may be sampled often. For example, a heavily sampled load or store instruction may indicate a cache miss situation. If the code is missing the cache often then there may be an opportunity to optimize the code to prevent the cache miss.

Examining Data

Data can be examined in two modes, called **Flat** mode, and **Drill-down** mode.

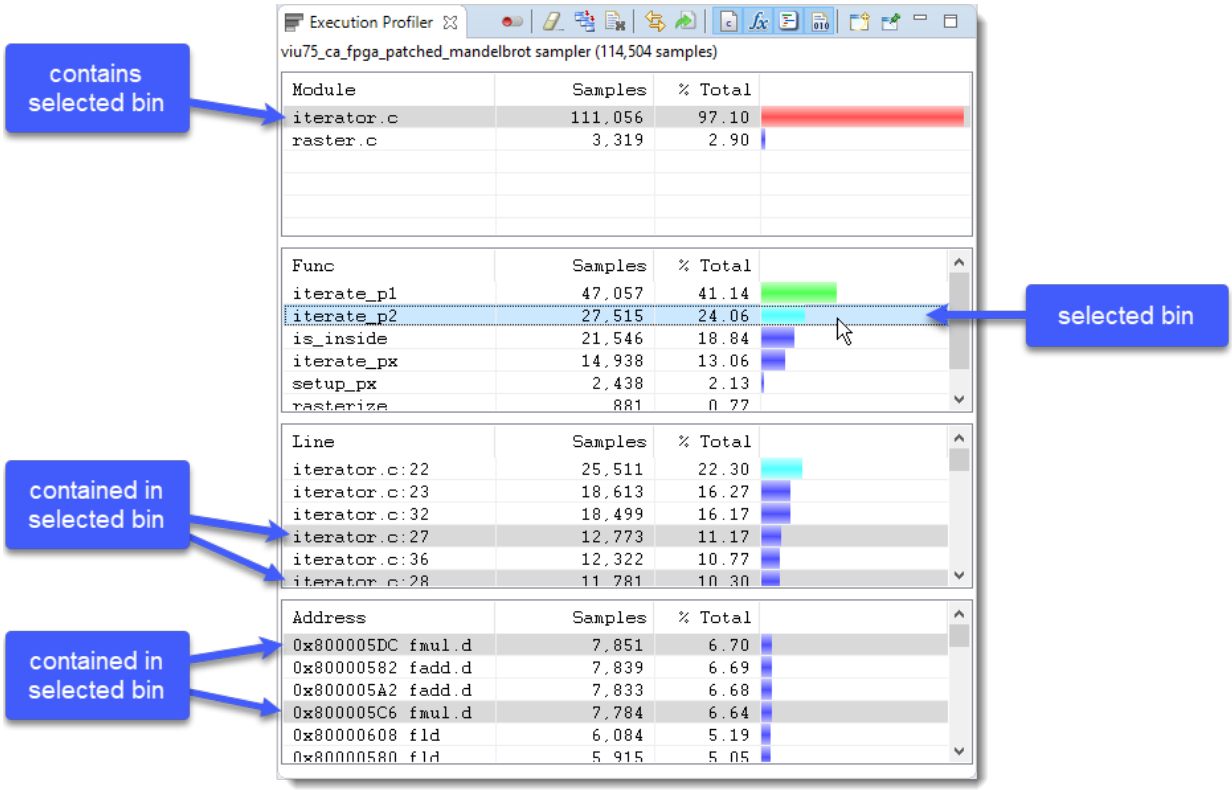
In flat mode each table shows all bins that contain any samples. In Drill-down mode each lower-level table only shows bins contained in the selected bins of a higher-level table.

The display mode is controlled by the “Toggle Drill-down Mode” switch on the view toolbar:



Flat Mode

When in Flat mode, selecting a one or more bins in any table will cause the other tables to reflect the selection by highlighting (in grey) those bins that contain (or are contained by) the selected bin(s). For example:



Drill-Down Mode

In drill-down mode each table only shows bins that are contained in the selected entries of the parent table. Drill-down mode is good for "hiding" all bins that are not related to the "scope" that you want to examine. For instance, if you just want to examine a specific function, select that function in the function table and the line and address tables will update to only show bins that are contained in the selected function.

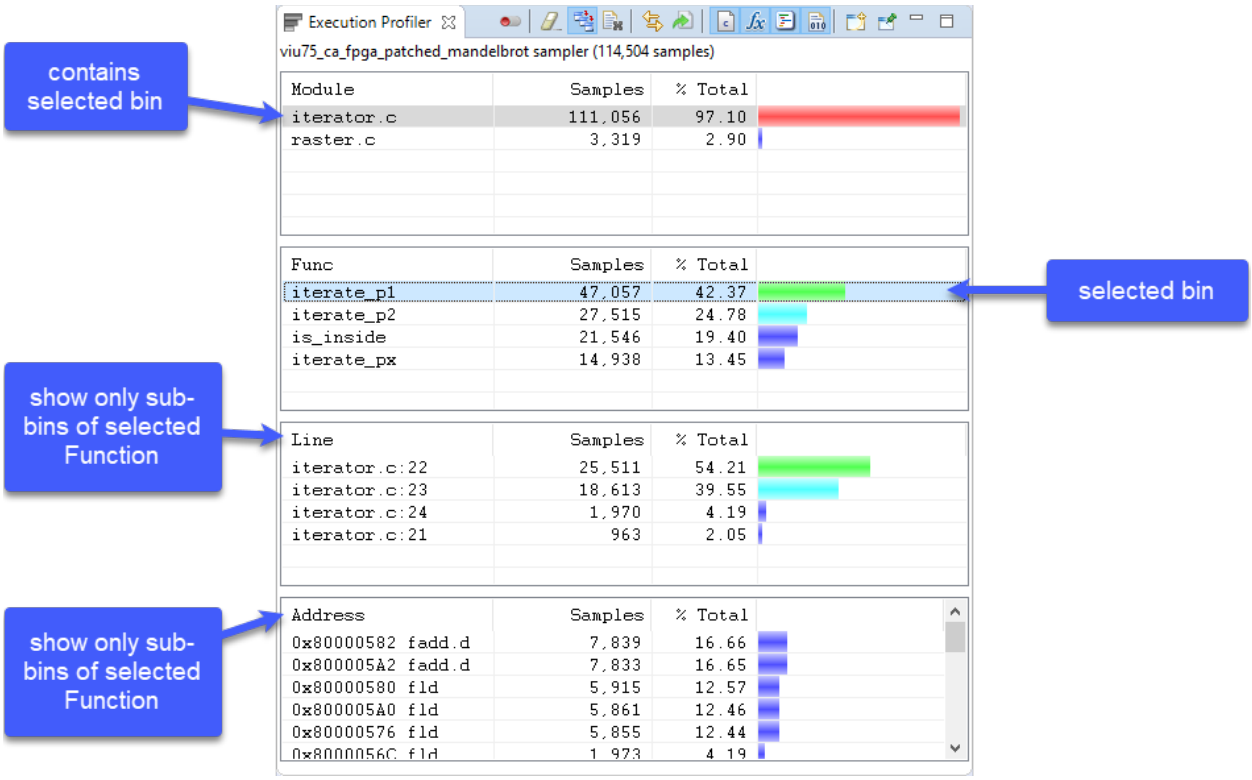
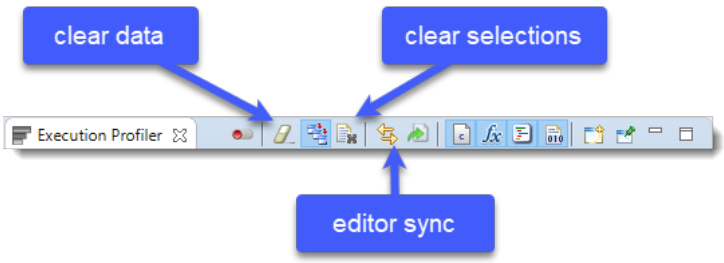


Table selections can be further controlled by selecting (or deselecting) items in each table. Downstream child tables will be updated based on the selections of the parent table.

Miscellaneous Sampling Features



Clear Data: clears all sampling data, used to reset accumulated sampling data.

Clear Selections: clears all table selections. Useful in drill-down mode to "reset" the view to show all sample bins when you may want to drill down on a different selection. Useful in flat mode to simply clear "containment" indicators in all tables.

Editor Sync: The toggle button (on the left), when enabled, will cause the source editor and disassembly view to show the source line and assembly instruction for any sample bin you select using the mouse.

The command button (on the right) can be used to cause the source editor and disassembly view to show the source line and assembly instruction for the currently selected record. This can also be done by double-clicking any bin (when the toggle sync button is turned off).

FPGA Programming

FPGA Programming Using xc3sprog/openocd

Before you continue

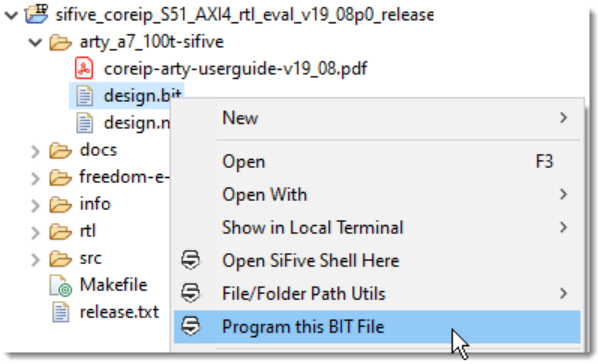
Before continuing with this section please review the Target Board Setup instructions to ensure that everything is properly configured, and all host dependencies have been installed.

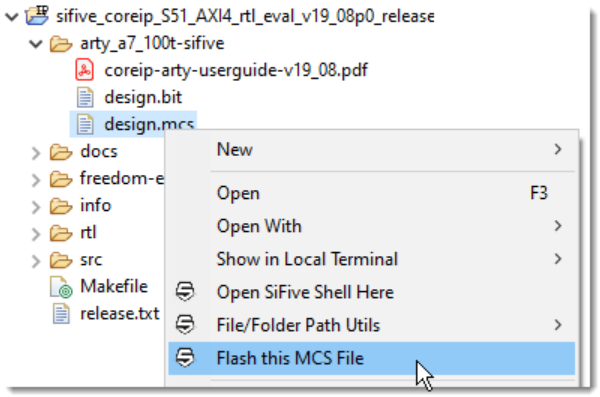
Flashing an MCS file on the FPGA requires both the Olimex probe and the Arty board USB connector be connected to the host PC. Both USB connections are used during the process. Do not simply connect the Arty USB to a power supply when flashing.

Programming a BIT file to the Arty requires only the Arty board USB connection be connected to your PC. The Olimex probe is not used in the programming process, however, having the Olimex connected will ensure that the correct device drivers for debugging with the Olimex are installed.

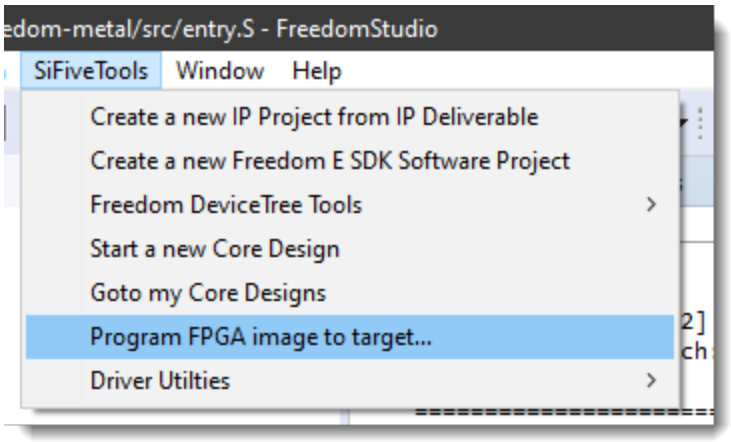
Programming an Arty Board using a JLink connection is not supported at this time.

The easiest way to program an MCS or BIT file onto the Arty board FPGA is to right-click on the file in the project explorer (you can also simply double-click the MCS or BIT file):

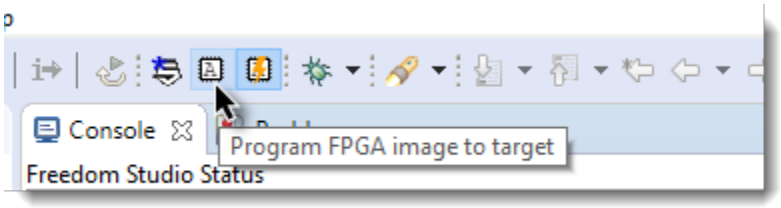




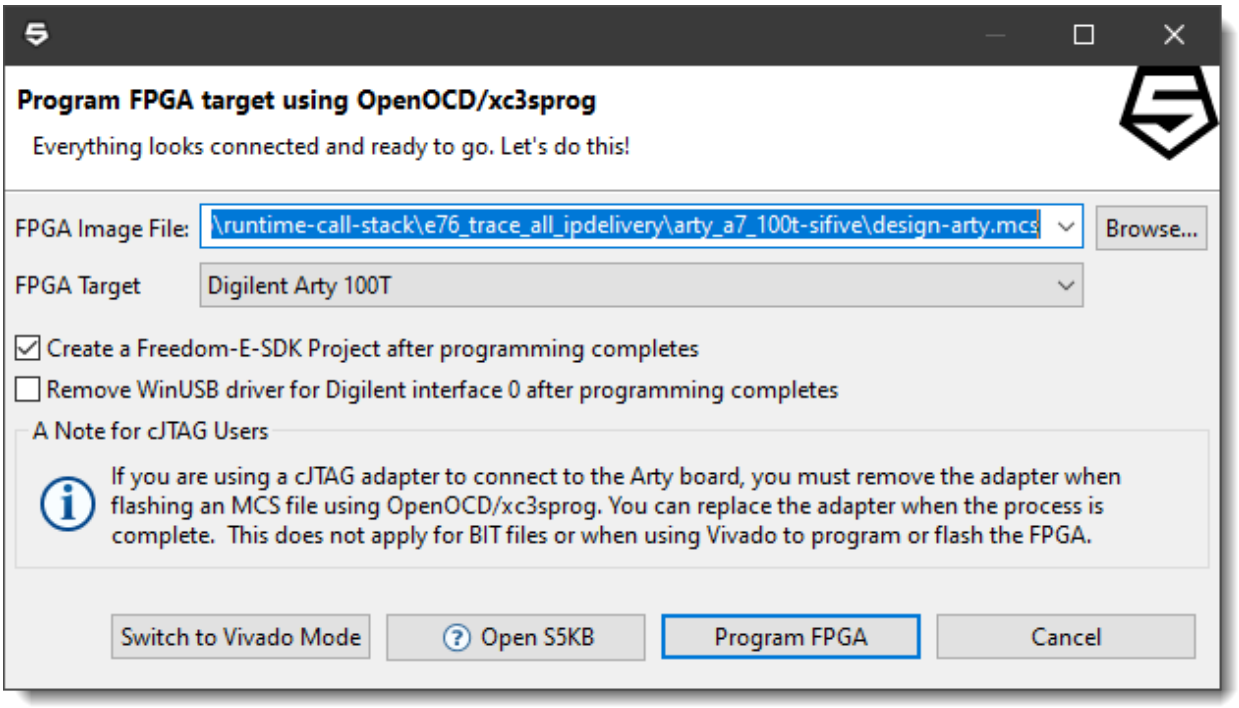
If the FPGA image file is not in a workspace project, you can open the programmer dialog from the main menu by selecting SiFiveTools → Program FPGA image to Arty…



or by clicking the FPGA Programming icon on the main toolbar:



Selecting either of these will open the FPGA Programming Dialog. This dialog will look a little different on each host platform:



1. First select the MCS or BIT file you want to program. These files are available in Core IP deliverables; in downloaded evaluation packages, or directly from Sifive, or

may be created in your flow. Use the Browse button to locate and select the desired image file.
2. Important: Make sure you select the correct FPGA configuration for your MCS file and Arty board. Freedom Studio will attempt to select the correct setting, but if it cannot be determined heuristically, no default selection is made and you will have to choose. Choose wisely. [This setting is not applicable for BIT files and the controls will be disabled when you select a BIT file to program]
3. Create a Freedom-E-SDK Project: Check this box if you want to open the New Freedom E SDK Project Wizard when the programming process is completed.
4. If you intend to use Vivado to program your FPGA bitstream you can have Freedom Studio uninstall the device driver used by xc3sprog. This will allow Vivado to connect to the target. If you do not plan to use Vivado to program images, then leave this box unchecked as it will speed up future programming operations within Freedom Studio.

Once you have made your selections, click the **[Program FPGA]** button to start the programming process. See the notes below regarding Windows hosts.

Flashing an MCS file can take several minutes to complete. When it is complete Freedom Studio will prompt you to press the PROG button on the FPGA board. You must do this to load and use the newly flashed MCS file.

Programming a BIT file is much faster (just a few seconds). Programmed BIT file are ephemeral. Power-cycling the board or pressing the PROG button will "erase" the programmed BIT file. Freedom Studio can be configured to program BIT files as part of a debug launch (See FPGA Programming at Launch)

Windows Only

On Windows host platforms Freedom Studio can monitor the connection status of the Olimex probe and the Target Digilent connection. The FPGA Programmer will report the status (as shown above) and the **[Program FPGA]** button will not be enabled unless all required devices are detected as connected.

Freedom Studio also monitors the driver status for both devices and will install required drivers as parts of the programming process. You may have to authorize the driver installation if Windows displays a UAC prompt. Programming will not succeed unless you authorize the driver installation.

Advanced Quick Programming

If you hold a <SHIFT> key down when you right-click on an image file and select the menu entry to program the file, the FPGA Programming Dialog will not be opened and the selected image file will be immediately programmed to the target board.

It is important that you know that everything is set up and working properly to ensure a successful programming operation. If you are unsure, do not use the <SHIFT> key shortcut.

This shortcut works with VCU118 images, Arty 100T MCS images, and both 35T and 100T BIT files. If you need to program a 35T MCS file you need to use the dialog to select the 35T option.

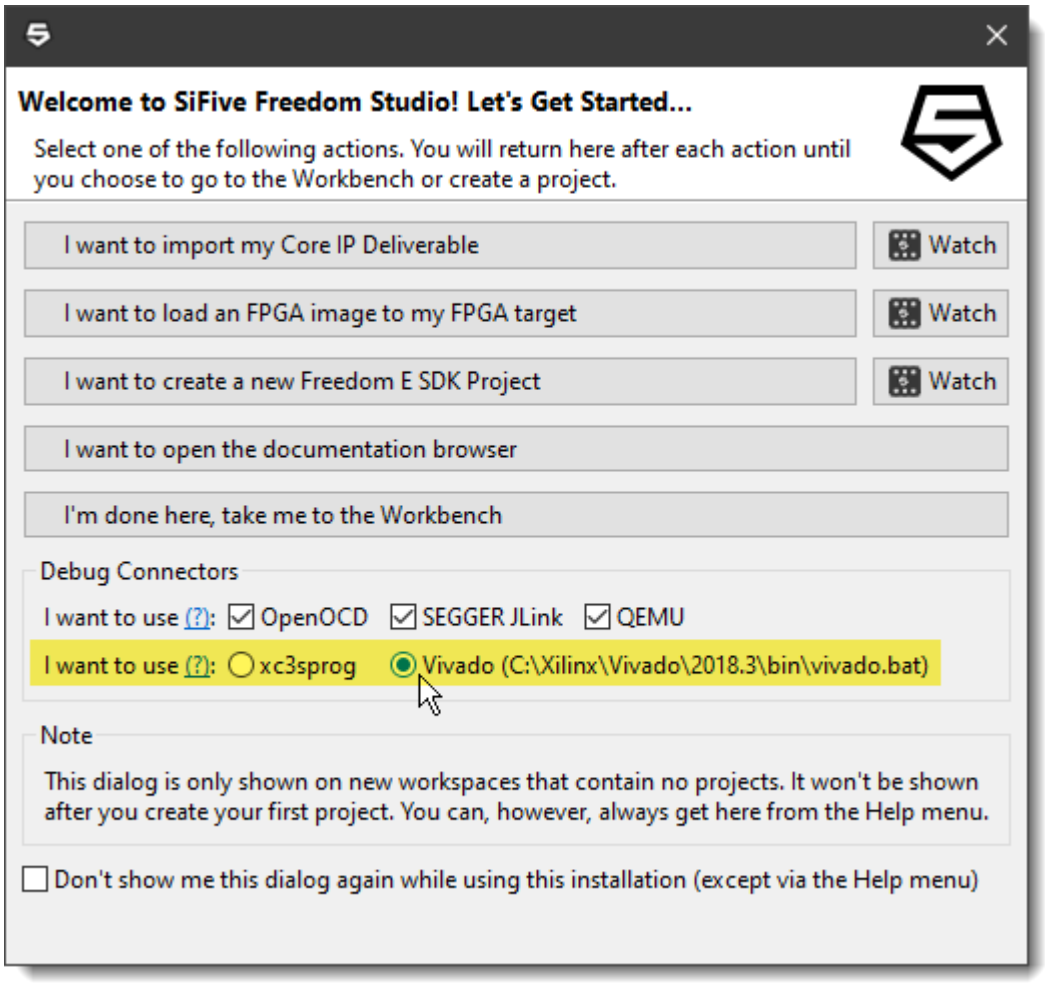
FPGA Programming Using Vivado

Windows and Linux users now have a choice of using Vivado or xc3sprog/openocd to flash MCS files or program BIT files to the FPGA. Freedom Studio has two "modes", "Vivado" mode, and "xc3sprog/OpenOCD" mode.

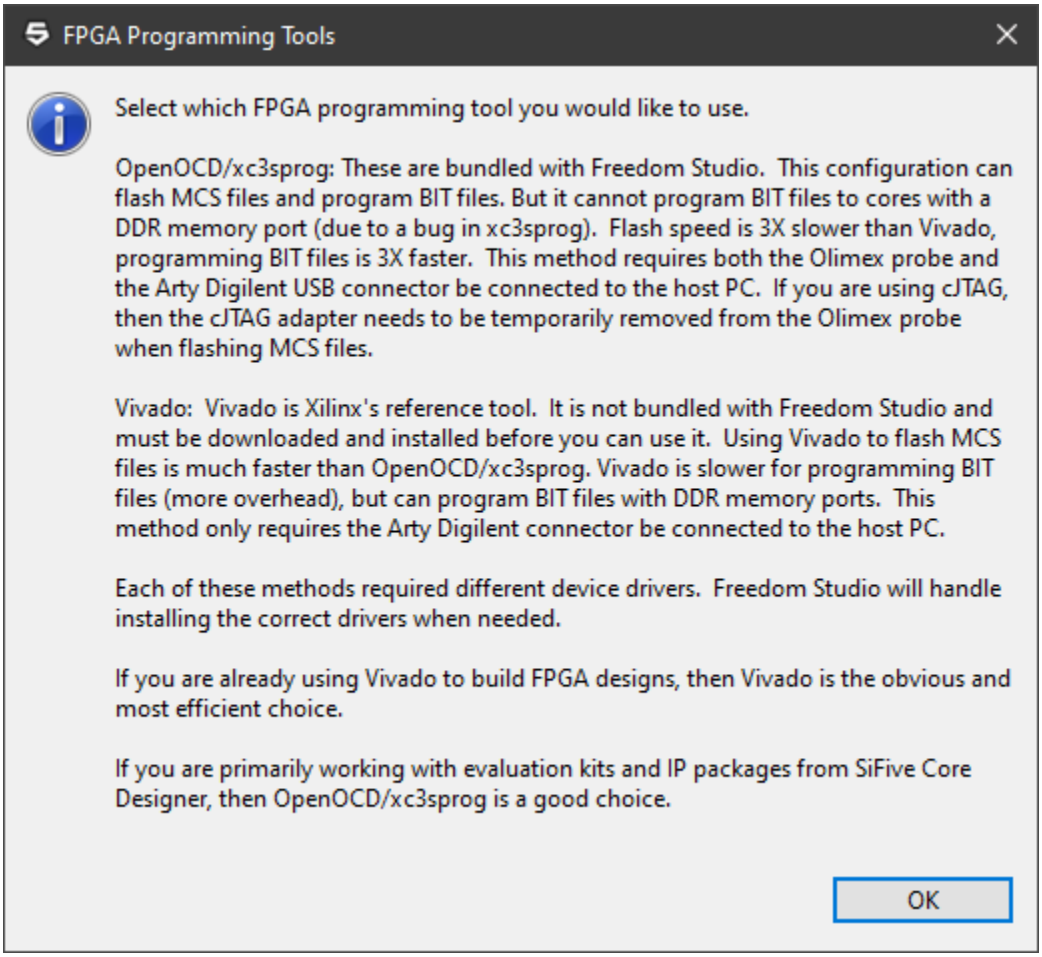
NOTE: *Vivado is not available on MacOS.*

NOTE: *Vivado is not bundled with Freedom Studio. Before you can use "Vivado" mode you must download and install Vivado or Vivado Lab on your host system. This mode is most useful for users that are actively developing FPGA images (and will probably already have Vivado installed). This mode is also recommended if your are using an FPGA target other than an Arty board (i.e. VC707, VCU118, or other Xilinx FPGA target), as xc3sprog and OpenOCD may not support programming or flashing these other targets.*

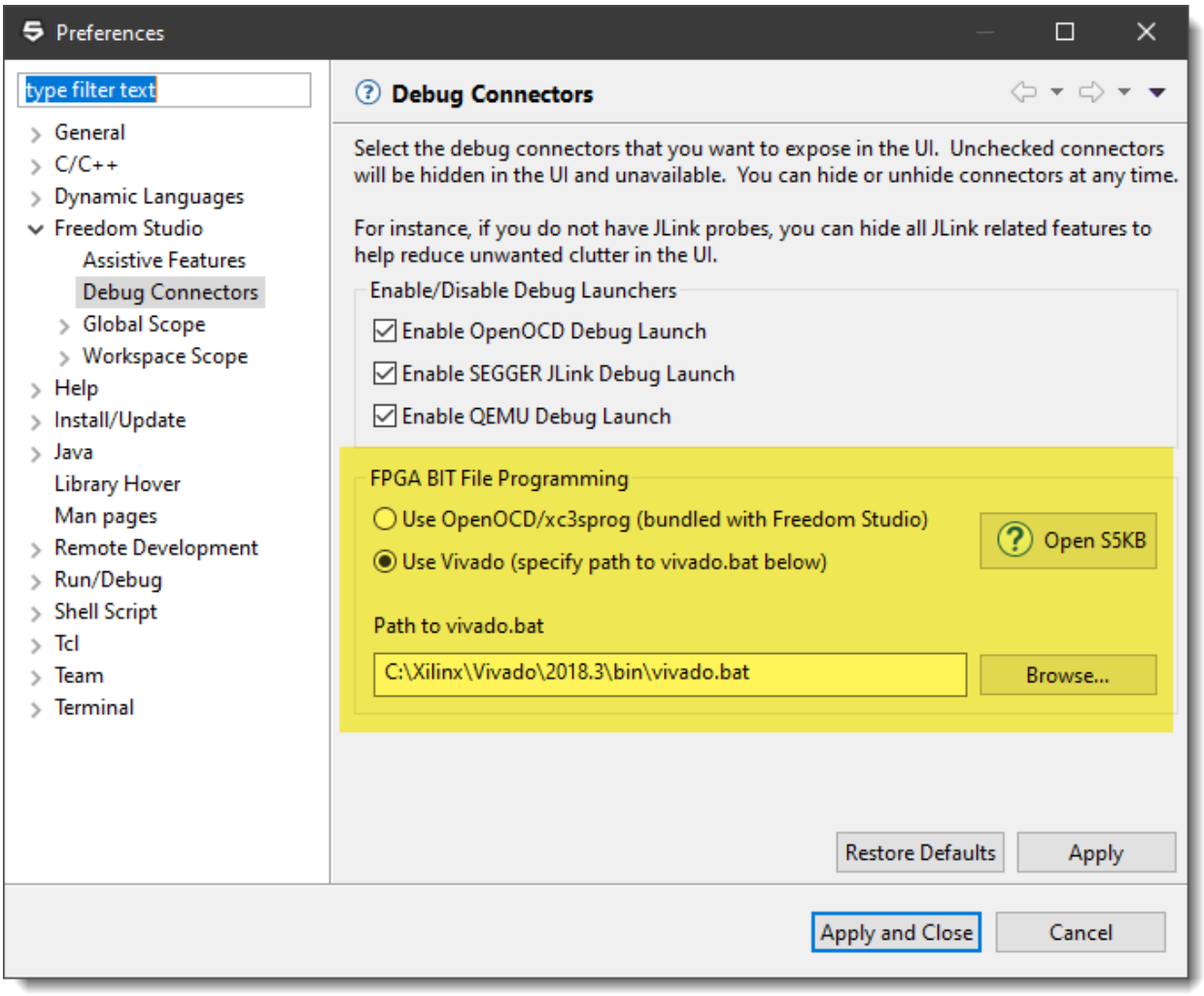
Selecting the mode to use is done using the Getting Started Dialog:



Pressing the little (?) link brings up a summary explanation of the two modes:



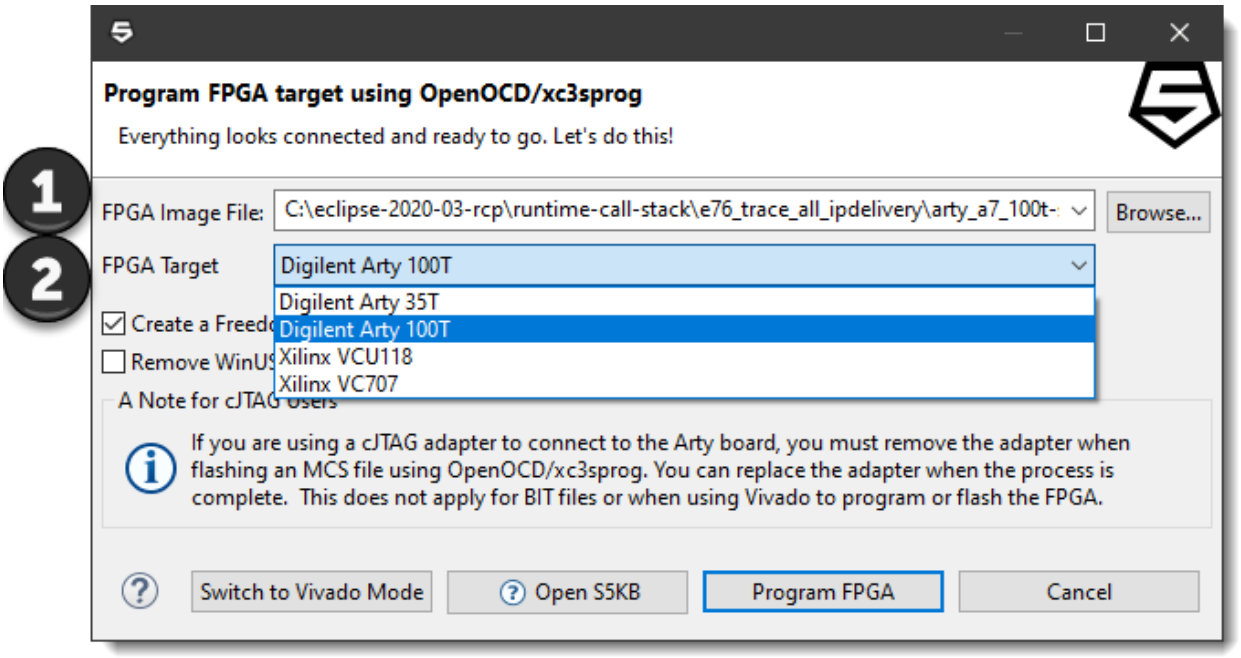
You can also select the mode via the Freedom Studio Preference Dialog:



In order to use Vivado mode you need to specify the path to the vivado.bat/vivado_lab.bat (on Windows), or vivado/vivado_lab (on Linux).

The Flash Programming Dialog is updated to support both programming modes and to support programming BIT files.

When using “Vivado” mode the Flash Dialog looks like:



1. Specify the FPGA MCS or BIT file
2. Specify the FPGA target type

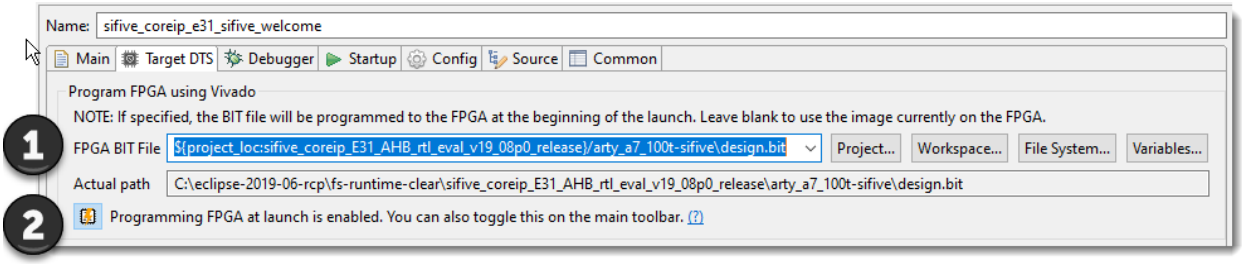
Additional FPGA targets can be added easily. Contact support for instructions on how to add more.

If required, you can switch between Vivado mode and Xc3sProg mode right from the Flash Programming dialog box.

FPGA Programming at Launch

Debug launch configurations now allow for specifying an FPGA bit file to be programmed at the start of a launch. FPGA bit files can be programmed very quickly (compared to flashing MCS files). This feature allows a developer to easily work with multiple FPGA images without having to separately flash or reprogram the FPGA image between launches.

Specifying a BIT file to program at launch is done on the “Target DTS” tab of the launch configuration dialog:



1. Specify the BIT file here. You can use any of the resource picker buttons to easily locate the correct BIT file.
2. Global Switch: There is a global switch that enables or disables programming the FPGA at launch. The switch is shown here (the small icon) and is also present in the main toolbar. You can use this switch to temporarily disable FPGA programming at launch when you know the FPGA is already programmed. This will save you time during the launch when you may need to launch debug sessions often with the same FPGA image.

NOTE: *Programming BIT files to the FPGA target is not “permanent” like flashing an MCS file. If the target is reset (via the PROG button) or power-cycled, then the FPGA image will be lost and need to be reprogrammed. If you are primarily using a single FPGA image, then flashing the MCS file may be a better and more efficient approach.*

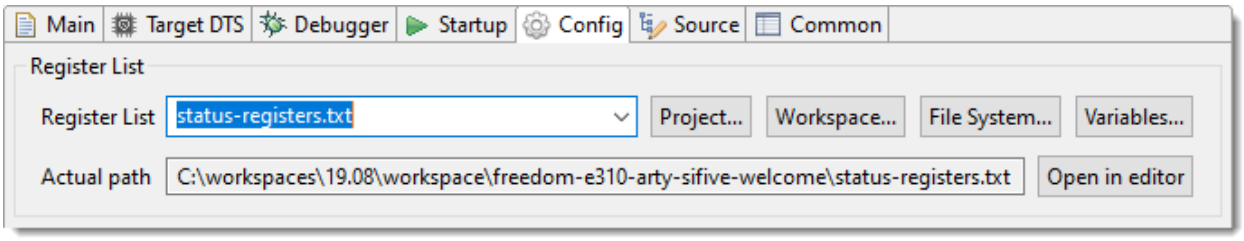
Register List Management

This document provides an overview on how to use and customize the list of registers displayed in the Freedom Studio IDE Registers View. This feature is primarily intended to give you control over what registers are displayed. You may want to use this, for example, when you do not want to see a complete list of all target registers. Or alternately, you may want to specify registers that are not included in the default list of registers.

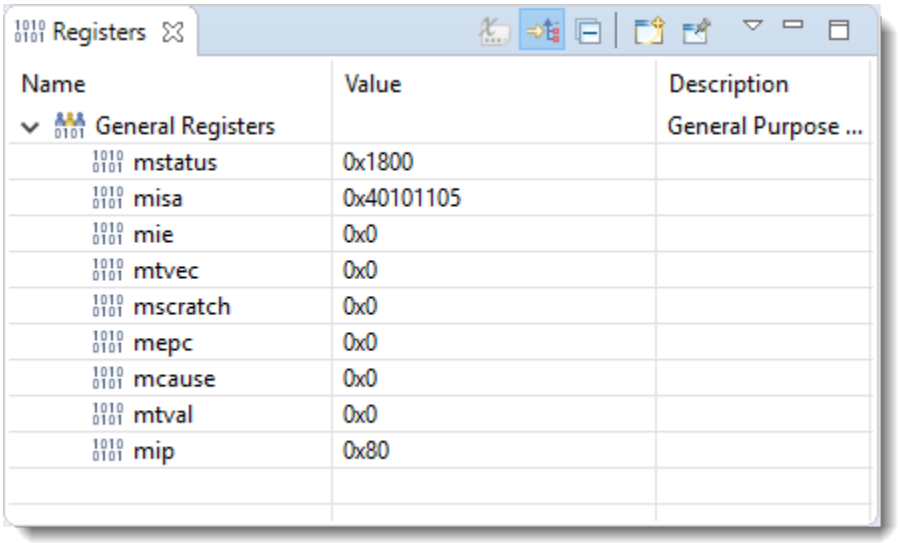
A Quick Example

Let us assume you have a register list file called 'status-registers.txt'. The content of the file looks like:

Now specify that file as a register list in the debug launch configuration:



When you launch your debug session and open the Register View you will see this list:



Creating Register List Files

The Register List File is a text file that, at its simplest form, lists a single register name on each line. Each listed register will be displayed in the Registers View in the order specified.

Commenting the Register List File

The register list file treats any line that starts with a # (hash) character as a comment line. It is ignored by the parser. The # character can be preceded by whitespace.

Only the first word of a line is treated as a register name. Any additional words are ignored.

Specifying Register Names

Single Registers

Any register can be specified by putting the name of the register as the first word on a line.

Built-in Macros

The following macros can be used to specify multiple related registers without having to list each register individually

Built-in Register List Macros

Macro Name	Description
general_registers	The 32 General Purpose Registers plus PC
machine_registers	The machine status registers
perfmon_registers	Performance Monitor Control and Data Registers
fpu_registers	Floating Point Registers
smode_registers	Supervisor Mode Registers
gdbregisters	Include all known registers

Include File

You can create several register list files, for example, building your lists of related registers, and then build a master register list by including these files in a composite register list file. To include another register list simply use:

```
#include <register-list-file>
```

The `#include` directive can be used multiple times in a single file.

Nested `#include` directives are supported. An `#include` file may `#include` additional files.

Where `register-list-file` is either an absolute or relative path. Relative paths are relative to the folder containing the current register list file being parsed. Keep this in mind if you are using nested `#include` directives and your register list files live in different folders.

Register Ordering

Registers are displayed in the Register View in the same order as they are specified in the Register List File.

Using Register List Files

Now that you have created one or more register list files you may want to use them with Freedom Studio. This section explains your options for specifying how to use your register list files.

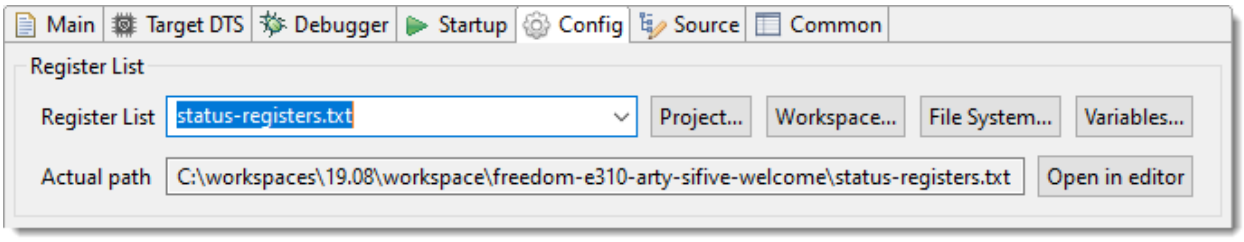
A register list file can be specified in 4 places. These four locations are prioritized such that a specification in a higher priority location will override any specification in a lower priority location. The four locations are, in descending priority order (highest priority first):

Prioritized Register List Specification Locations

Location	Description
Debug Launch Config	Specify a register list file for each individual launch configuration
Project Property	Specify a register list file for each project
Workspace Preference	Specify a register list file for each workspace
Global Preference	Specify a global register list file, for all Freedom Studio workspaces

Debug Launch Configuration

This is the highest priority option for specifying a register list file. You will find the controls to specify the debug launch register list file on the `Config` tab of the Debug Launch Configuration Dialog:

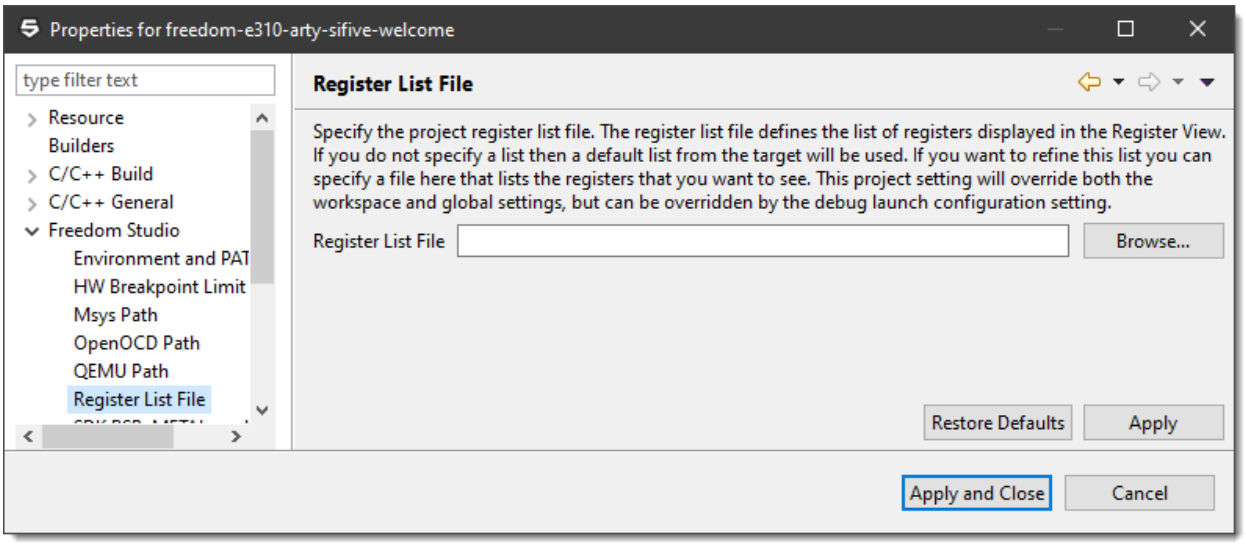


When you specify a register list file in a debug launch configuration the path displayed in the `Actual Path` box will always reflect the fully resolved path to the register list file. If you are not specifying a register list file here, then the `Actual Path` may display a path to another register list file if one has been specified using a lower priority specifier.

Project Property

Specifying a register list file as a project property will cause that register list to be used with all launch configuration created for the project, overriding any global or workspace preferences. Each launch can override the project specification by using the launch configuration option to specify a register list file.

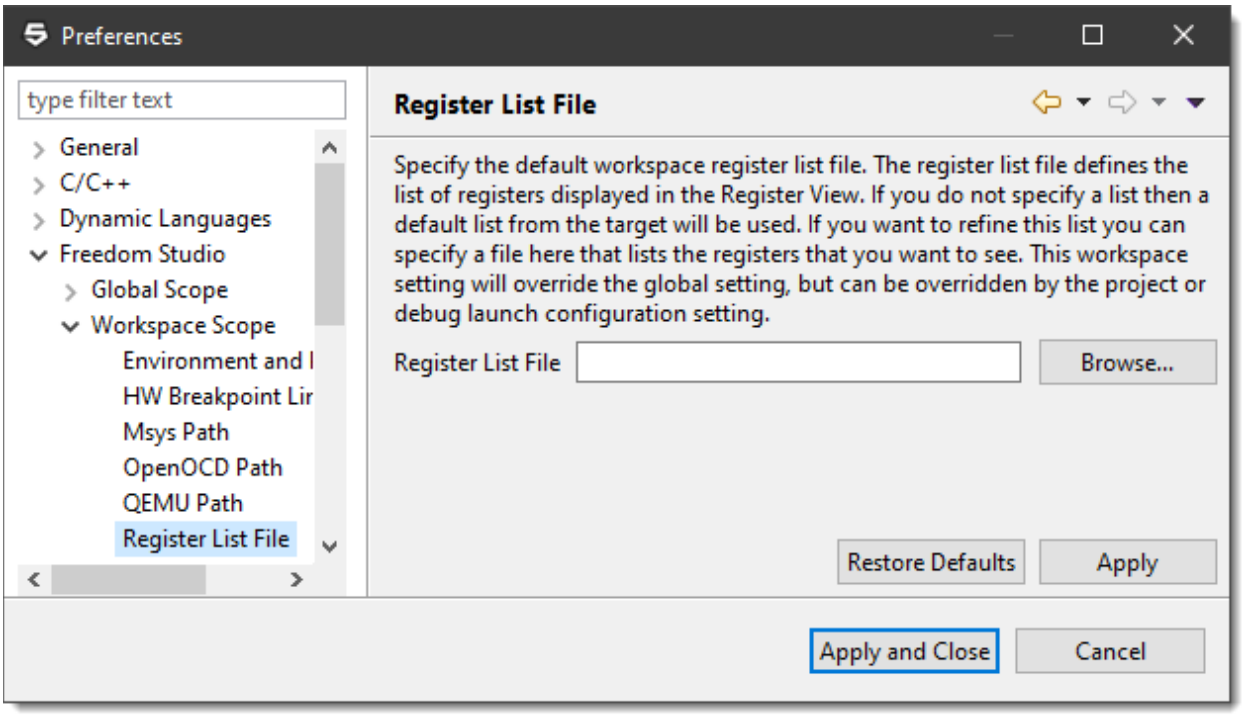
You can setup the project register list file specification by opening the Project Properties dialog and navigating to the `MCU → Register List` property page:



Workspace Preferences

Specifying a workspace register list file will cause that file to be used for all projects within the workspace unless a project overrides the setting by specifying a register list in the project properties or a debug launch configuration.

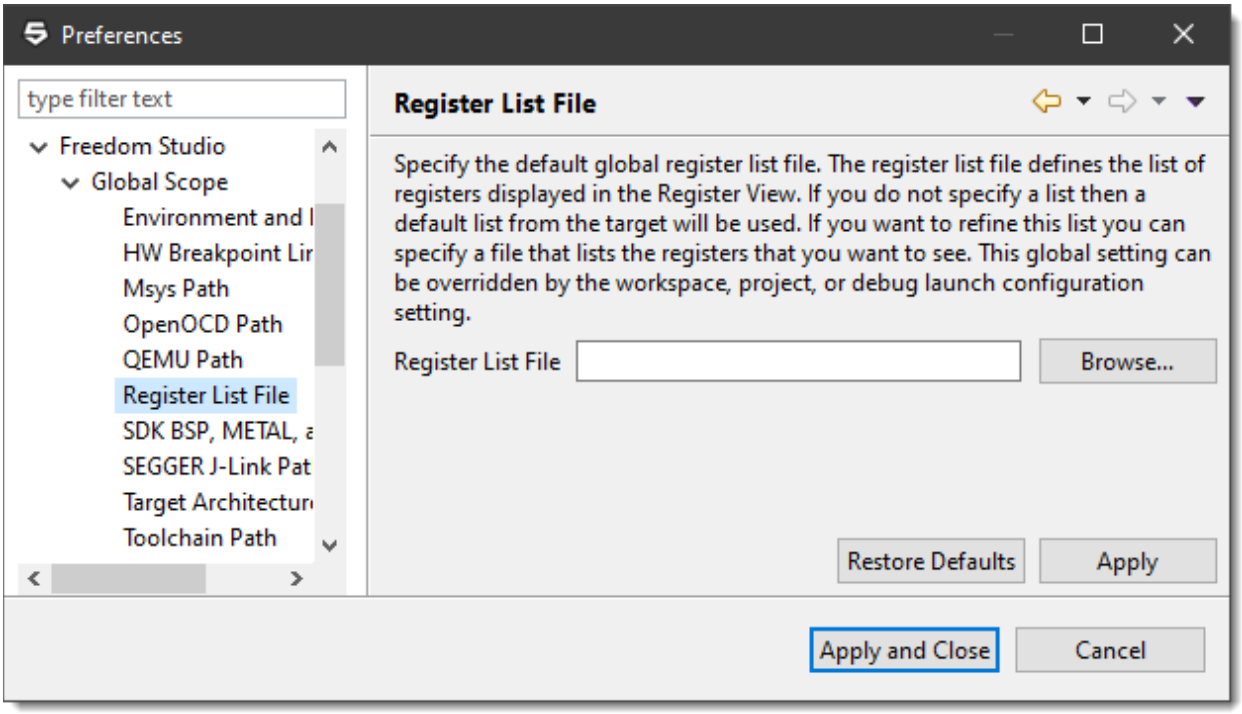
You can specify the workspace preference by opening the Freedom Studio Preference Window and navigating to the `MCU → Workspace Register List` page:



Global Preferences

Specifying a global register list file will cause that file to be used for all Freedom Studio workspaces unless a workspace, project, or debug launch overrides the setting.

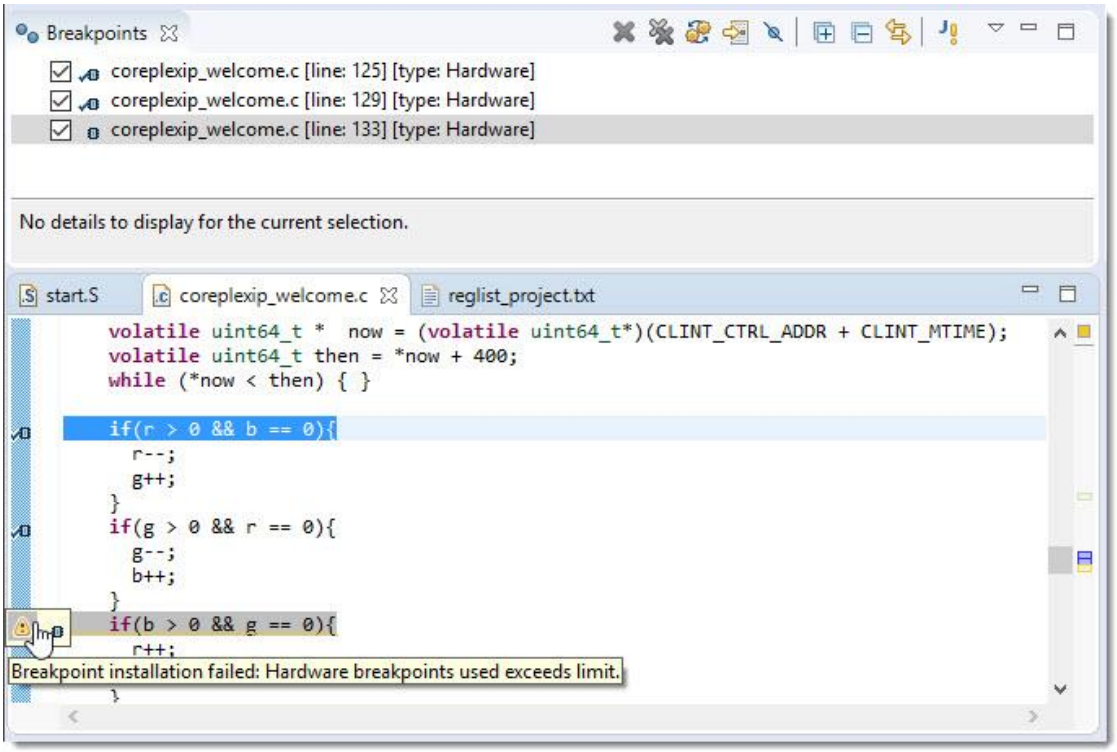
You can specify the global preference by opening the Freedom Studio Preference Window and navigating to the `MCU → Global Register List` page:



Managing Hardware Breakpoint Resources

This document summarizes how to manage hardware breakpoint resources on a target system. Different cores have different numbers of hardware breakpoints. It is important for GDB to know how many hardware breakpoints exist on a target. Attempting to use more breakpoints than exist on the target will cause unpredictable debugger problems.

When GDB knows how many hardware breakpoints exist on the target, you can create as many hardware breakpoints as you need, but only the number that exist will be enabled. Freedom Studio will indicate which breakpoints cannot be enabled due to lack of resources. You can then manage the enablement of each breakpoint to ensure that the breakpoint you need is enabled (by disabling breakpoints that you do not need). This screenshot shows how Freedom Studio indicates that too many hardware breakpoints have been enabled.



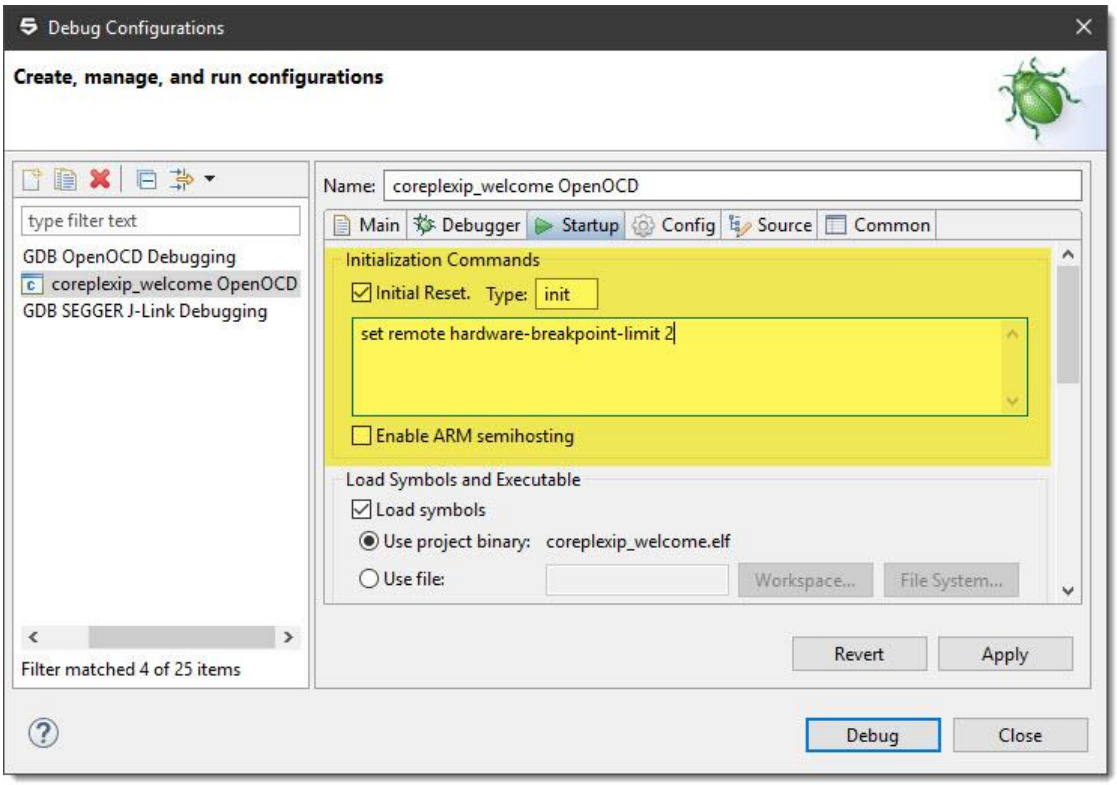
Too Many Hardware Breakpoints

Freedom Studio cannot automatically determine the number of hardware breakpoints present on the system. We plan to add this ability in a future release.

GDB needs to know the number of hardware breakpoints on the target. There are two ways to do this.

Option 1: Add a gdb initialization command

Add the 'set remote hardware-breakpoint-limit' command to the Initialization Commands section of a launch configuration. You must do this for every new launch configuration.



Note

Setting this setting using Option 1 takes precedence over Option 2 (described below). If you find that your preference setting is not being applied, check to make sure that you do not have this command specified in the Initialization Commands.

Option 2: Set a preference or project property

You can set global and workspace preferences to define the number of hardware breakpoints on your target system. You can also set this in your project properties and in a launch configuration.

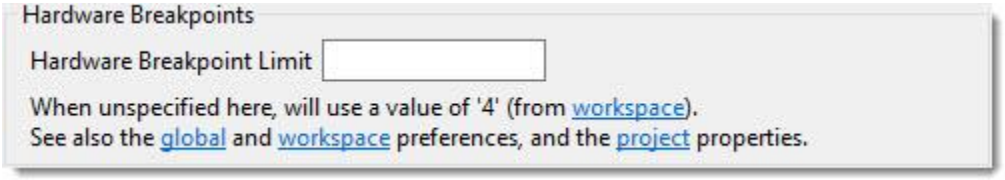
Finer-grain settings take priority over courser-grain settings. The priority, from highest to lowest is:

- Debug Launch Configuration
- Project Property
- Workspace Preference
- Global Preference

Each new launch configuration will use the highest priority setting that exists. If no setting exists, then Freedom Studio will use the hard-coded default of '2'.

The launch configuration dialog always describes the setting used and where the setting originates. For instance, the screenshot below shows the setting is 4 and originates from

the workspace preference setting. This implies that the project property setting has not been defined (it is blank). Clicking on any of the underlined setting scopes will open the corresponding settings page where you can change the setting if desired.



Setting value description

Valid settings

The following table shows the valid setting values.

Valid Setting Values

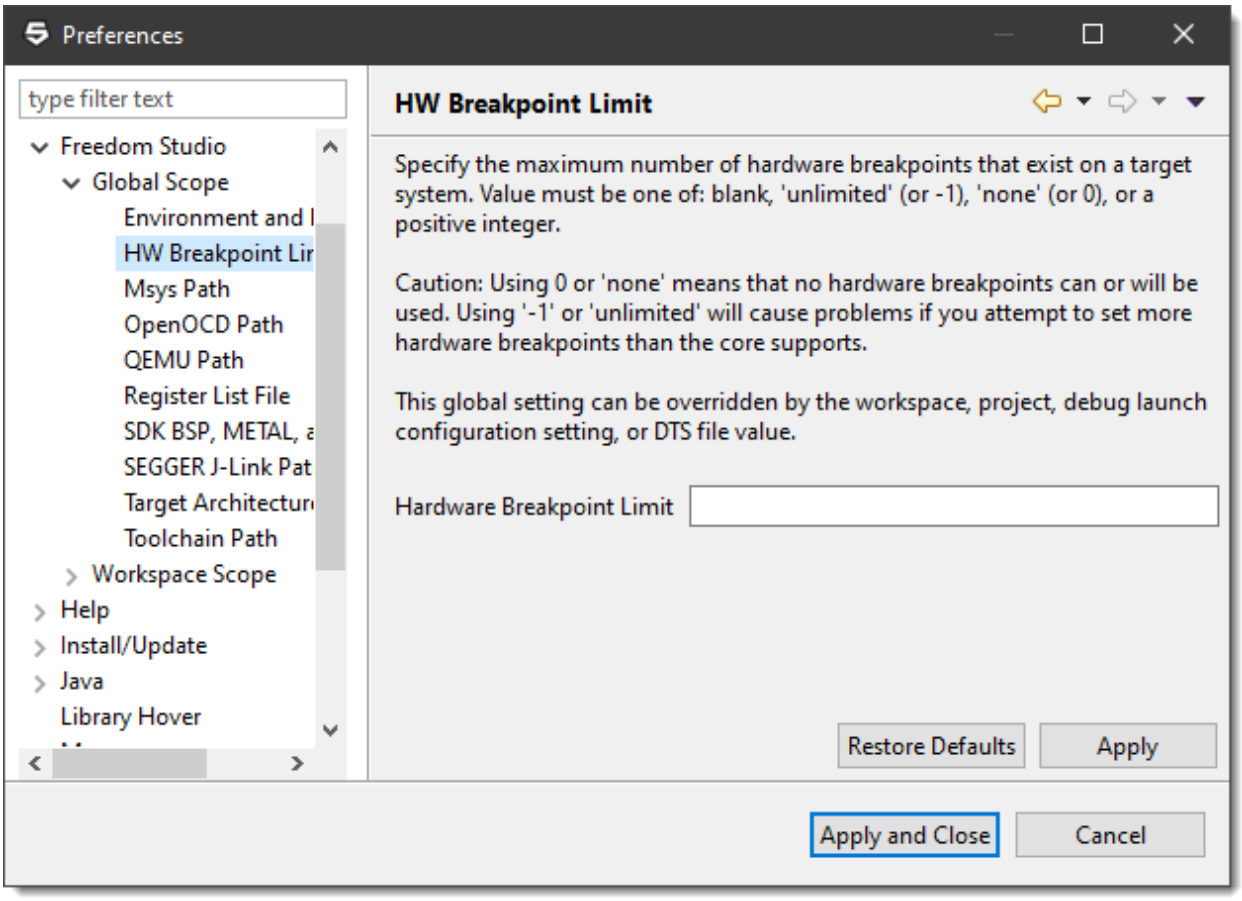
Value	Description
blank	Leave the setting blank and it will not be used.
'unlimited', or -1	Tells GDB that you have unlimited hardware breakpoints.
'none', or 0	Tells GDB that you have no hardware breakpoints.
x, a positive integer	Tells GDB that you have *x* hardware breakpoints.

GDB defaults to 'unlimited'. Freedom Studio overrides this default and uses '2'. Using 'unlimited' allows you to set more hardware breakpoints than may exist on the target. GDB will attempt to set all of them. This leads to unpredictable debugger behavior. We do not recommend using 'unlimited', but we won't stop you from doing so.

Setting the Global Preference

We recommend setting the hardware-breakpoint-limit globally when you have a single target system. This ensures that the setting applies in all workspaces, projects, and launch configurations. If you ever need to use a different target that has a different number of hardware breakpoints you can easily override the global setting using any of the higher priority settings.

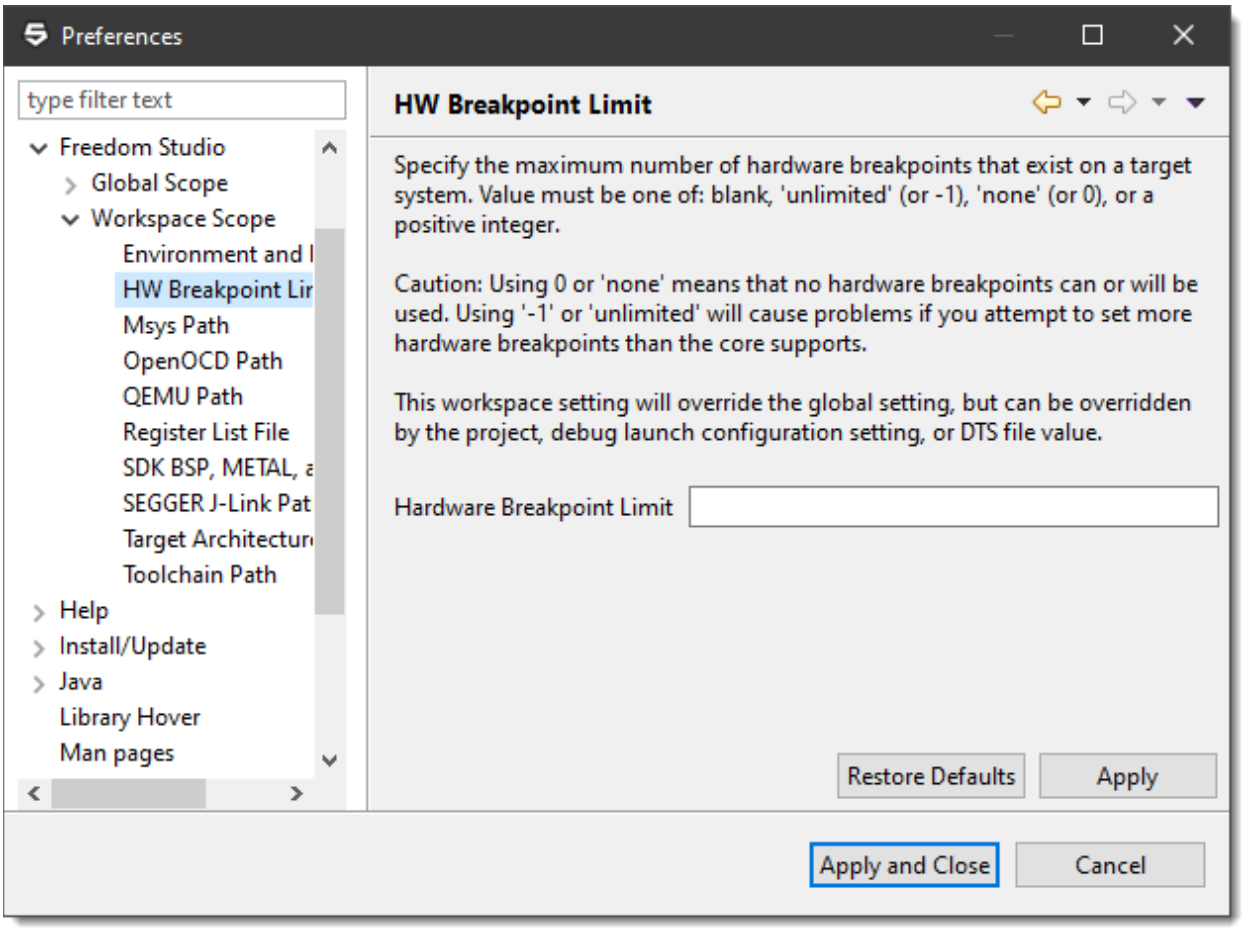
Set the global preference by opening the Preferences Dialog (`Windows -> Preferences`) and navigating to the `MCU | Global HW Breakpoint Limit` page.


Preferences
type filter text
Freedom Studio
Global Scope
Environment and I
HW Breakpoint Lir
Msys Path
OpenOCD Path
QEMU Path
Register List File
SDK BSP, METAL, a
SEGGER J-Link Pat
Target Architectur
Toolchain Path
Workspace Scope
Help
Install/Update
Java
Library Hover
HW Breakpoint Limit
Specify the maximum number of hardware breakpoints that exist on a target system. Value must be one of: blank, 'unlimited' (or -1), 'none' (or 0), or a positive integer.
Caution: Using 0 or 'none' means that no hardware breakpoints can or will be used. Using '-1' or 'unlimited' will cause problems if you attempt to set more hardware breakpoints than the core supports.
This global setting can be overridden by the workspace, project, debug launch configuration setting, or DTS file value.
Hardware Breakpoint Limit
Restore Defaults
Apply
Apply and Close
Cancel

Setting the Workspace Preference

We recommend using the Workspace Preference when you have multiple target systems and want to create a workspace for each target system. This ensures the setting is correctly applied for the target used in each workspace.

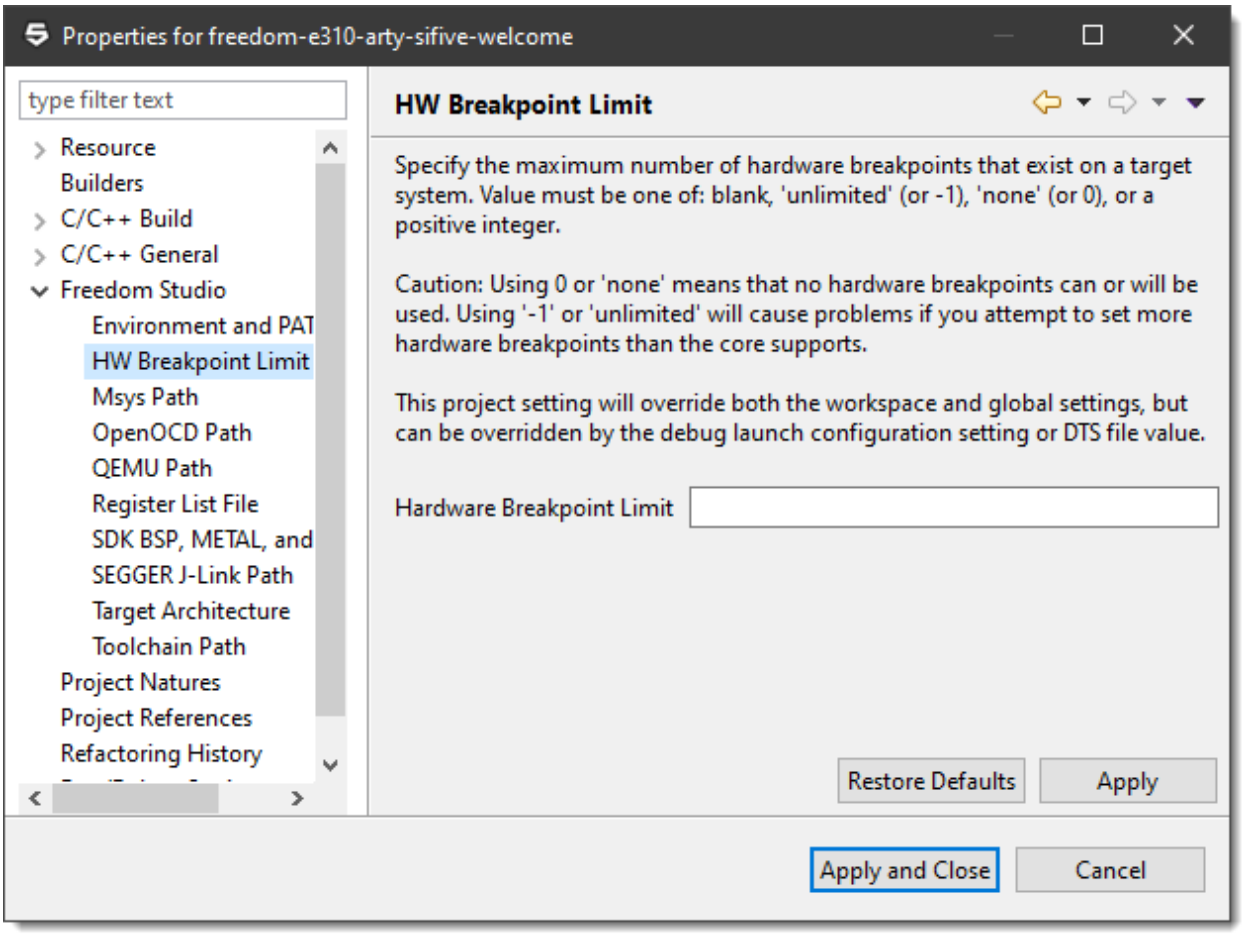
Set the workspace preference by opening the Preferences Dialog (`Windows -> Preferences`) and navigating to the `MCU | Workspace HW Breakpoint Limit` page.



Setting the Project Property

We recommend using the Project Property setting when you have multiple target system and want to work on all of them within a single Workspace. This ensures the setting is correctly applied for the target used in each project.

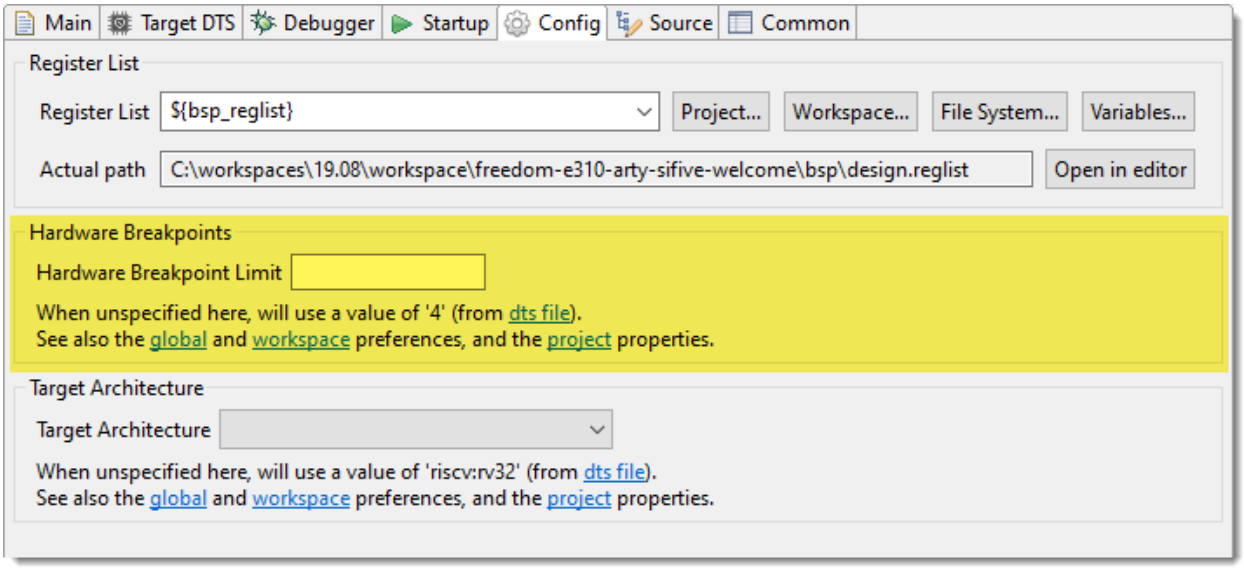
Set the project property by opening the Project Properties Dialog (`Project -> Properties`) and navigating to the `MCU | HW Breakpoint Limit` page.



Setting the Launch Configuration Attribute

We recommend using the launch configuration attribute setting for target connections that you do not use often. This ensures the setting is not applied to oft-used targets (that are better served using a more broadly applied setting from the project, workspace, or global settings).

Set the launch configuration attribute by opening the launch configuration dialog, navigating to the 'Config' tab, where the breakpoint count can be set for this single launch config.



Conditional Optimization

This section describes how to apply compiler optimization conditionally within a source file.

Debugging optimized code can be complicated because the optimizer will change the order of the code and optimize out variables. When single-stepping through the code the source line indication can jump around erratically. You will not be able to examine variable values that have been optimized away.

The normal solution is to turn off optimizations for the entire project when you need to debug something. Sometimes this is not desirable (or even possible). In these cases you can turn off optimization for just the code that needs to be debugged using compile-time `#pragma` statements.

The comments in the following source example explain how, when, and when not do use the `#pragma` statements.

Example source code.

```
// See LICENSE for license details.

#include <stdint.h>
#include <stdbool.h>
#include <stdatomic.h>
#include "encoding.h"
#include <platform.h>

#ifndef _SIFIVE_COREPLEXIP_ARTY_H
#error 'coreplexip_welcome' demo only supported for Coreplex IP Eval Kits
#endif

void pwm(uint16_t r, uint16_t g, uint16_t b);
uint16_t option0(uint16_t p1, uint16_t p2);
uint16_t option1(uint16_t p1, uint16_t p2);
uint16_t option2(uint16_t p1, uint16_t p2);

static const char sifive_msg[] = "\n\r\
\n\r\
                SIFIVE, INC.\n\r\
\n\r\
         55555555555555555555555\n\r\
        5555                   5555\n\r\
       5555                     5555\n\r\
      5555                       5555\n\r\
     5555       55555555555555555555\n\r\
    5555       5555555555555555555555\n\r\
   5555                             5555\n\r\
  5555                               5555\n\r\
 5555                                 5555\n\r\
```

```
55555555555555555555555555          55555\n\r\
 55555           555555555           55555\n\r\
   55555           55555           55555\n\r\
     55555           5           55555\n\r\
       55555                   55555\n\r\
         55555               55555\n\r\
           55555           55555\n\r\
             55555       55555\n\r\
               55555   55555\n\r\
                 555555555\n\r\
                   55555\n\r\
                     5\n\r\
\n\r\
";

#if __riscv_xlen == 32
  static const char welcome_msg[] = "\n\r\
\n\r\
Welcome to the E31 Coreplex IP FPGA Evaluation Kit!\n\r\
\n\r";
#else
static const char welcome_msg[] = "\n\r\
\n\r\
Welcome to the E51 Coreplex IP FPGA Evaluation Kit!\n\r\
\n\r";
#endif

static void _putc(char c) {
  while ((int32_t) UART0_REG(UART_REG_TXFIFO) < 0);
  UART0_REG(UART_REG_TXFIFO) = c;
}

int _getc(char * c){
  int32_t val = (int32_t) UART0_REG(UART_REG_RXFIFO);
  if (val > 0) {
    *c =  val & 0xFF;
    return 1;
  }
  return 0;
}

static void _puts(const char * s) {
  while (*s != '\0'){
    _putc(*s++);
  }
}

/*
 * Enable maximum optimization for the main() function.  The pragma directive
```

```
for
 * optimization act on a function scope. You cannot place these pragmas
around code
 * inside of a function (it will generate a compiler error).
 */
#pragma GCC push_options
#pragma GCC optimize ("3")
int main (void){

  // 115200 Baud Rate at (65 / 2) MHz
  UART0_REG(UART_REG_DIV) = 282;
  UART0_REG(UART_REG_TXCTRL) = UART_TXEN;
  UART0_REG(UART_REG_RXCTRL) = UART_RXEN;

  // Wait a bit because we were changing the GPIOs
  volatile int i=0;
  while(i < 10000){i++;}

  _puts(sifive_msg);

  _puts(welcome_msg);

/*
 * These pragma, if uncommented, will generate compiler errors because this
 * only works when used outside of functions.  Optimization is performed
 * on function blocks, not on individual code lines.
 */
//#pragma GCC push_options
//#pragma GCC optimize ("0")
  uint16_t r=0x3F;
  uint16_t g=0;
  uint16_t b=0;
//#pragma GCC push_options

  PWM0_REG(PWM_CFG)   = 0;
  PWM0_REG(PWM_CFG)   = (PWM_CFG_ENALWAYS) | (PWM_CFG_ZEROCMP) |
(PWM_CFG_DEGLITCH);
  PWM0_REG(PWM_COUNT) = 0;

  // The LEDs are intentionally left somewhat dim.
  PWM0_REG(PWM_CMP0)  = 0xFE;

  while(1){
    volatile uint64_t *  now = (volatile uint64_t*)(CLINT_CTRL_ADDR +
CLINT_MTIME);
    volatile uint64_t then = *now + 400;
    while (*now < then) { }

    if(r > 0 && b == 0){
```

```
      r--;
      g++;
    }
    if(g > 0 && r == 0){
      g--;
      b++;
    }
    if(b > 0 && g == 0){
      r++;
      b--;
    }

    pwm(r,g,b);

    g = option0(r, b);
    b = option1(r, g);
    r = option2(g, b);

  }// While (1)
}
#pragma GCC pop_options

/*
 * This function uses the project setting for optimization
 */
void pwm(uint16_t r, uint16_t g, uint16_t b)
{
    PWM0_REG(PWM_CMP1)  = 0xFF - (r >> 2);
    PWM0_REG(PWM_CMP2)  = 0xFF - (g >> 2);
    PWM0_REG(PWM_CMP3)  = 0xFF - (b >> 2);
}

/*
 * Enable maximum optimization.  The 'result' variable will be optimized out.
 */
#pragma GCC push_options
#pragma GCC optimize ("3")
uint16_t option0(uint16_t p1, uint16_t p2) {
    int result = p1 * p2;
    return result;
}
#pragma GCC pop_options

/*
 * Turn off all optimization. The 'result' variable is not optimized out.
 */
#pragma GCC push_options
#pragma GCC optimize ("0")
uint16_t option1(uint16_t p1, uint16_t p2) {
```

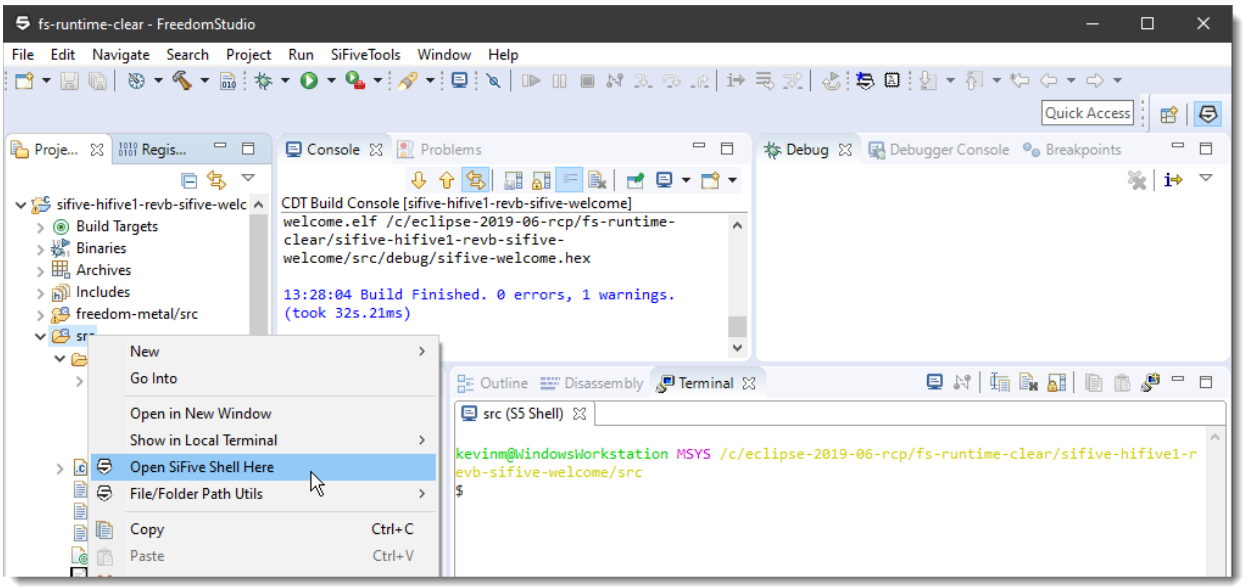
```c
    int result = p1 * p2;
    return result;
}
#pragma GCC pop_options

/*
 * Enable maximum optimization.  The 'result' variable would normally be
optimized out.
 */
#pragma GCC push_options
#pragma GCC optimize ("3")
uint16_t option2(uint16_t p1, uint16_t p2) {
    /*
     * Use 'volatile' keyword to ensure variable does not get optimized out.
     */
    volatile int result = p1 * p2;
    return result;
}
#pragma GCC pop_options
```

The SiFive Shell

The SiFive Shell refers to both the shell environment that Freedom Studio uses to build example software projects and to the interactive shell that can be opened in the terminal view.

Opening an Interactive Shell

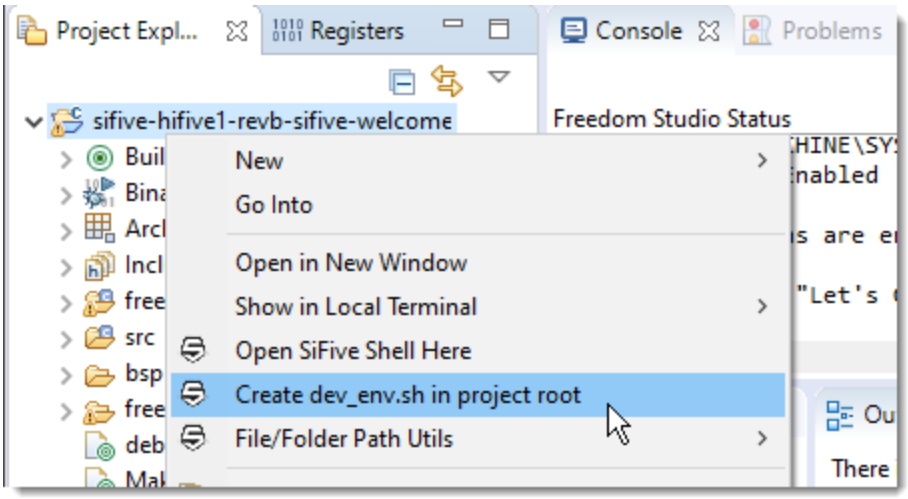
You can now open a shell in the Terminal View at any location in a project by right-clicking on the location in the Project Explorer and selecting “Open SiFive Shell Here”



By default, Freedom Studio will use bash for the shell on Window and the SHELL variable on Linux and MacOS. If you need to use a different shell, you can set the environment variable SIFIVE_SHELL=<path-to-shell-of-choice>. On Windows, you will want to ensure that your chosen shell is installed in the MSYS environment.

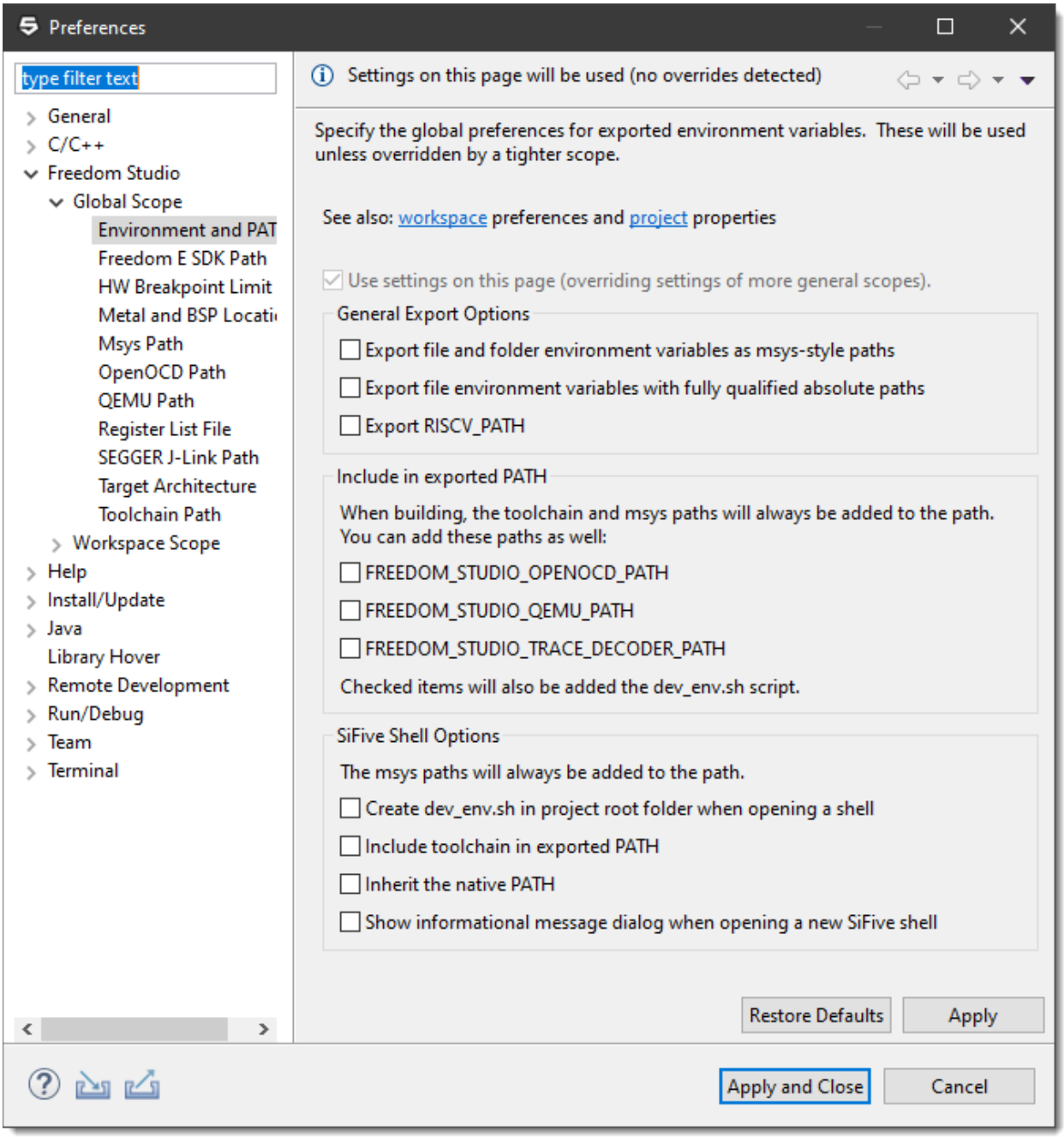
Create dev_env.sh for Your Project

The context menu for a project node in the Project Explorer has a new menu entry called “Create dev_env.sh in project root”. Selecting this menu item will create a new dev_env.sh script file in the project root. An existing dev_env.sh file will be replaced without warning. Do not hand edit this script file. Your changes will be lost.


Project Expl...
Registers
Console
Problems
sifive-hifive1-revb-sifive-welcome
Freedom Studio Status
New
Go Into
Open in New Window
Show in Local Terminal
Open SiFive Shell Here
Create dev_env.sh in project root
File/Folder Path Utils

Environment and PATH Exports

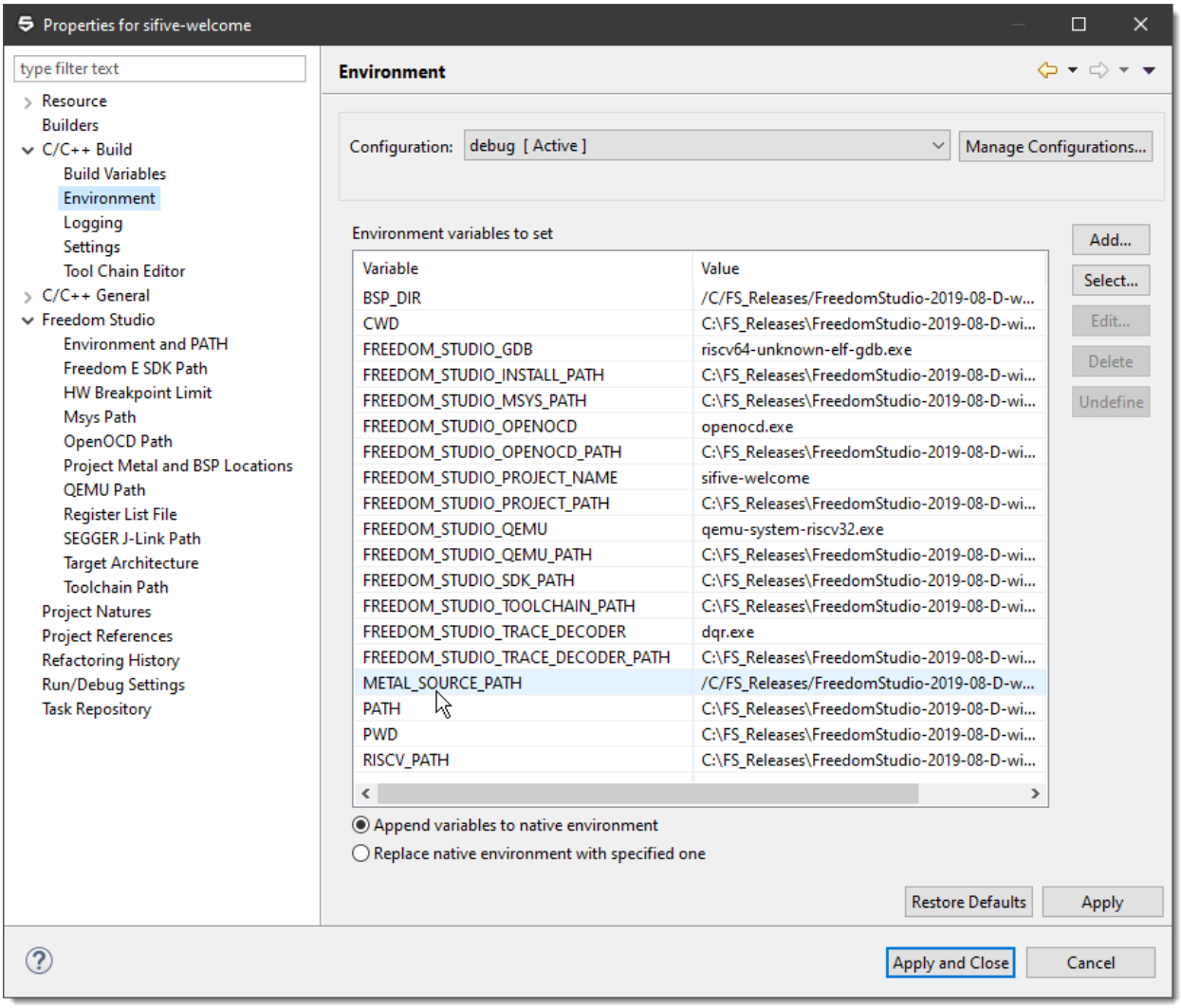
When you build a freedom-e-sdk based project several environment variables can be exported, and the PATH can be changed to include various tool locations. This is controlled in the new "Environment and PATH" node of the Global Preferences, Workspace Preferences, or Project Properties.



Notes:

- **Export RISCV_PATH**
  If checked. RISCV_PATH is exported in to the build environment and into the dev_env.sh script.
- **Create dev_env.sh in project root folder when opening a shell**
  This option, when enabled, will create a shell script file called dev_env.sh in the project root folder when you open a SiFive Shell. This script defines several environment variables and adds additional entries (if enabled) to the PATH. You can also create this file using the Project Explorer context menu on a Project node.
- **Include toolchain in exported PATH**
  If checked, the project toolchain will be added to the PATH in dev_env.sh
- **Inherit native PATH**
  [Windows Only] If checked, the native PATH will be added to the MSYS PATH
- **Show informational message dialog when opening a new SiFive Shell**
  [Global Preferences Only] If checked, Freedom Studio will display an informational dialog box when opening a new shell. This information summarizes the state of the dev_env.sh feature.

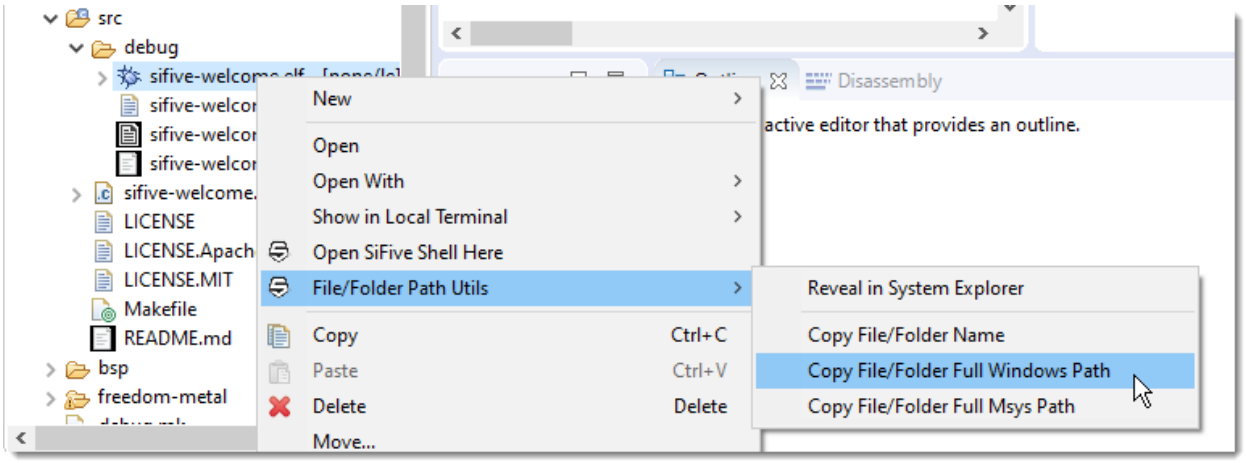
A complete list of exported environment variables can be viewed on the “C/C++ Build/Environment” project property node. For example:



All these variables are available to use in your Makefiles and scripts. But be aware that these are exported only when building from within Freedom Studio. If you want to maintain CLI builds then be sure to update your makefiles to specify appropriate defaults for any used variables, or to source the optionally generated dev_env.sh file.

File/Folder Path Utils

The context menu for project files and folders in the Project Explorer has a new sub-menu called “File/Folder Path Utils”. This submenu has some simple, but very useful items.



- **Reveal in System Explorer**
  Selecting this menu item will open the default file explorer application on you host system and take you to the folder containing the selected resource.
- **Copy File/Folder Name**
  Selecting this menu item will copy the file or folder name to the clipboard.
- **Copy File/Folder Full Windows Path**
  [Windows only] Copies the full absolute path of the selected resource to the system clipboard using Windows compatible paths.
- **Copy File/Folder Full Msys Path**
  [Windows only] Copies the full absolute path of the selected resource to the system clipboard using Msys compatible paths.
- **Copy File/Folder Full Path**
  [Linux/MacOS] Copies the full absolute path of the selected resource to the system clipboard.

Windows MSYS Environment

On Windows Freedom Studio uses a bundled MSYS environment to create a Linux-like environment required by the freedom-e-sdk.

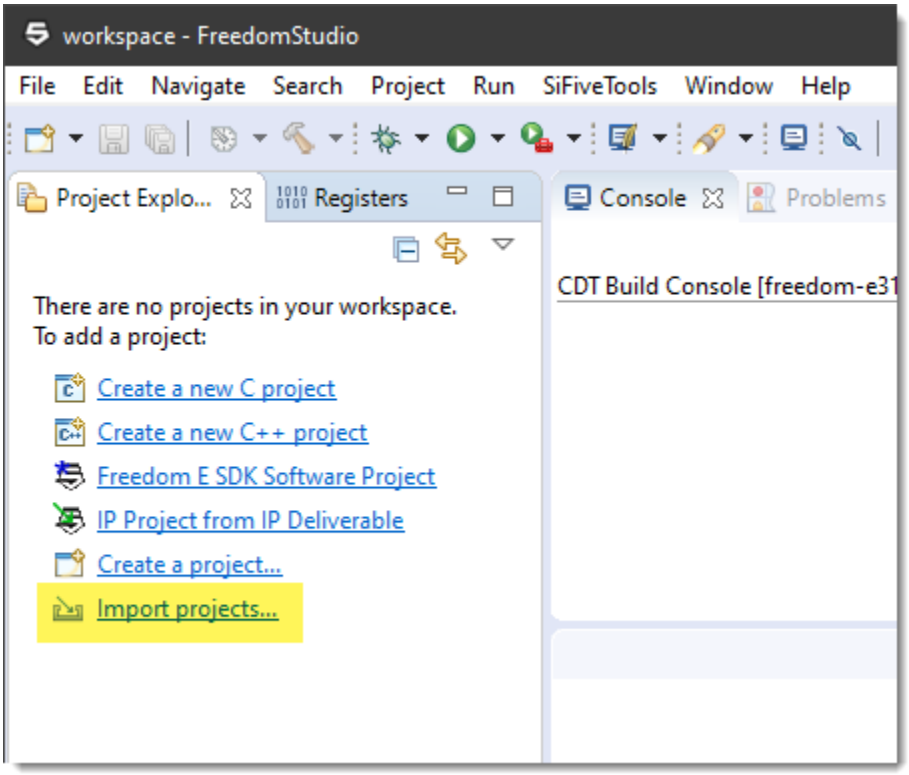
The Windows MSYS environment has been expanded to include many new tools and the ability to install additional MSYS packages using the ‘pacman’ tool. This expands the ability to write sophisticated Makefiles that can be used on all three host platforms.

You can also point Freedom Studio to a different MSYS environment (that you have installed and are responsible for managing) via the Global Preferences, Workspace Preferences, or Project Properties.

Migrating Projects from an older Freedom Studio workspace

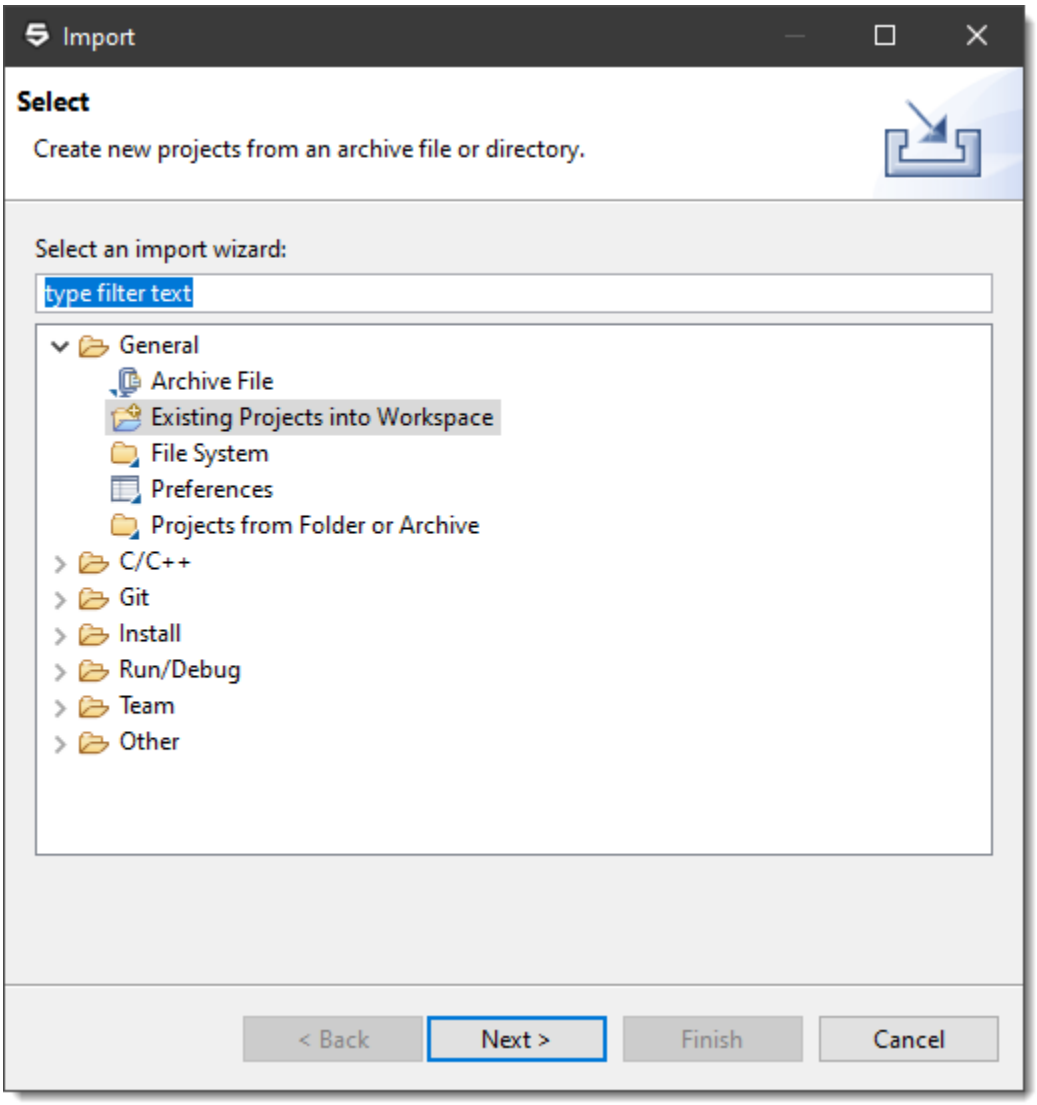
Migrating Projects

Older workspaces will be upgraded when opened in a newer version of Freedom Studio. This will render the workspace no longer compatible with the older version of Freedom Studio. A safer migration is to create a new workspace for the new version of Freedom and import the projects from the old workspace using the Import Wizard called "Existing Projects into Workspace" accessed via the Import dialog. If you have a new empty workspace open, then you can open this Dialog from the empty Project Explorer by selecting “Import projects…”



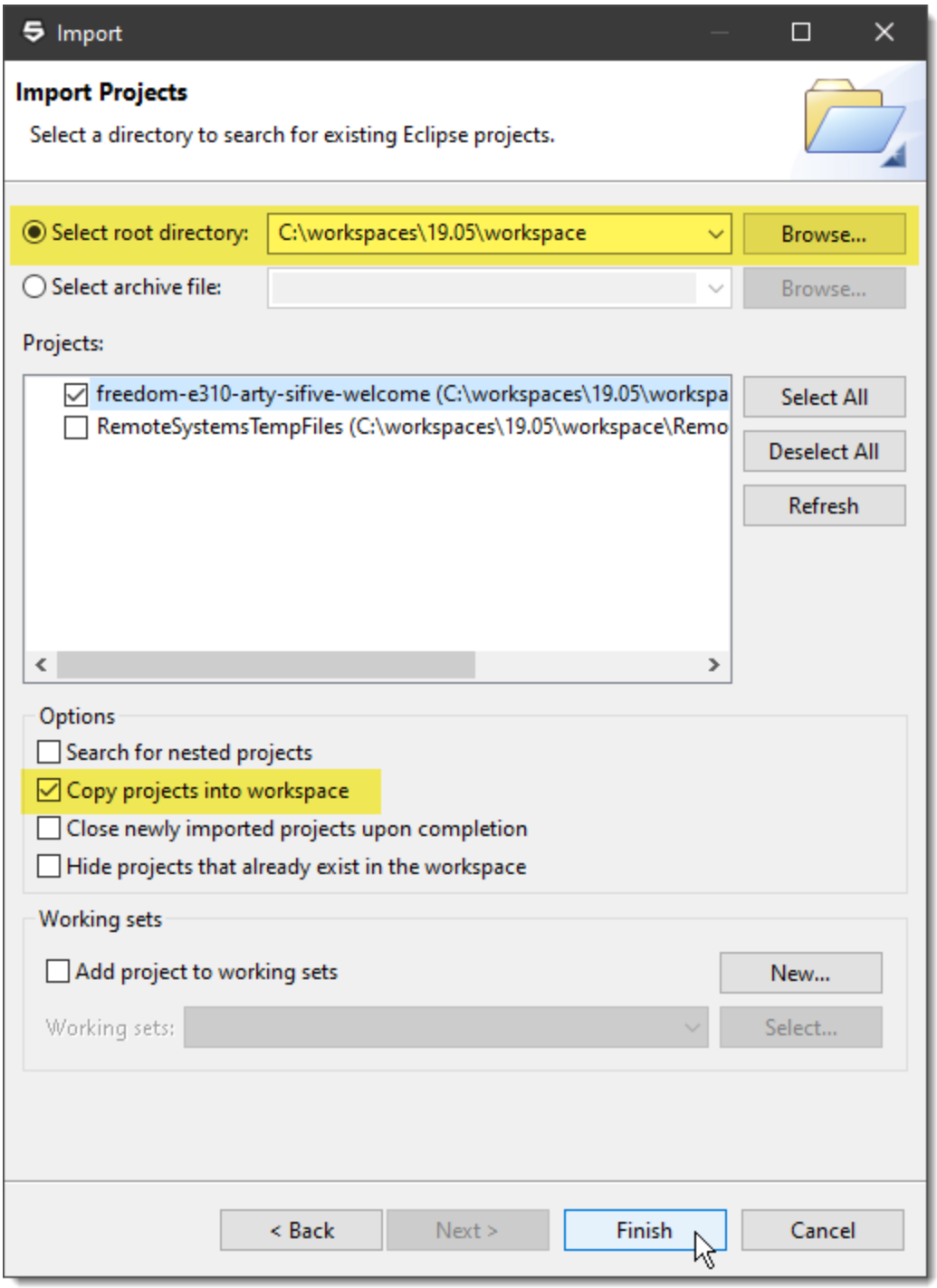
If you already have one or more projects in your workspace then open the Import dialog from the main menu: File → Import.

Once opened, expand the “General” category, and select “Existing Projects into Workspace”.



Click [Next>], then [Browse…] to select the Freedom Studio 2019.05 workspace directory. The Import wizard will show a list of all projects in the workspace. Check the ones you want to import. (If present, you should uncheck the project called “RemoteSystemTempFiles”)

Be sure to check "Copy projects into workspace" so that your original Freedom Studio workspace projects do not get updated.



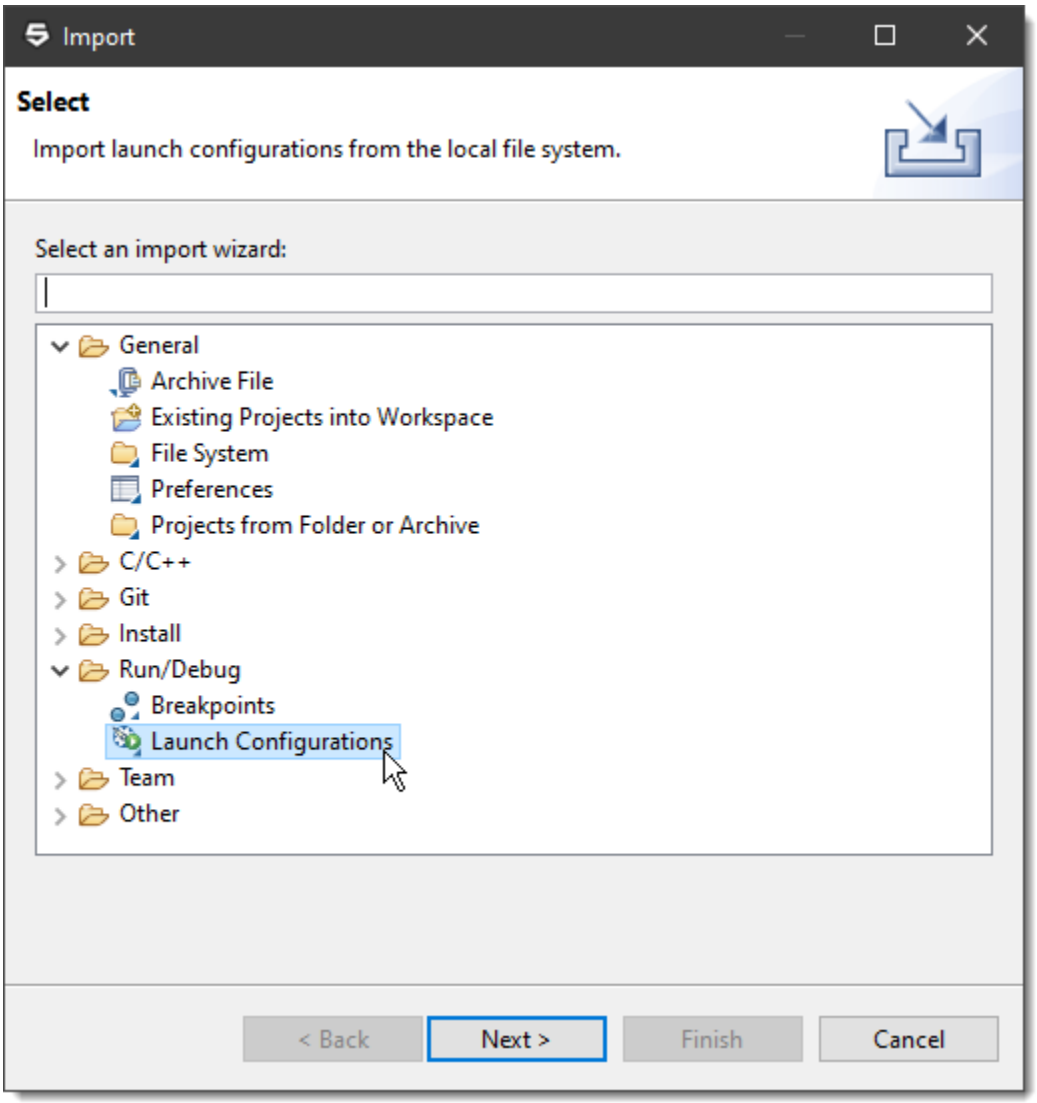
Once you have checked that all imported projects are functioning correctly you may then decide to delete the old Freedom Studio workspace and projects.

Migrating Debug Launch Configurations

Importing Debug Launch Configurations

If your debug launch configurations are "shared" configurations stored in your project directory, then they will be imported when you import your projects as described in the previous section.

If your debug launch configurations are "local" then they will not be imported when you import your projects. You will have to use the "Launch Configurations" import wizard. Open the Import dialog (File → Import), select the "Run/Debug" category, select the "Launch Configurations" wizard, then click [Next>] to open the wizard.



Now use the [Browse…] button to select the following location within your Freedom Studio workspace directory:

```
<workspace-dir>/.metadata/.plugins/org.eclipse.debug.core/.launches
```

The dialog will show the ".launches" directory in the left hand pane. Check the checkbox. The dialog will now list all "local" debug launches in the right pane. Check those that you want to import. Then click [Finish]

Updating Debug Launch Configurations

Debug launch configurations from older versions of Freedom Studio should be reviewed carefully as newer versions likely contain additional debug configuration options. Open each debug configuration and review all the tabs for new or missing configuration information.

Toolchain Management in Freedom Studio

You can configure Freedom Studio projects to use the correct toolchain needed for any project by configuring the project properties to point to the required toolchain base folder.

Freedom Studio can use any SiFive RISCV toolchain installed on your development system. It does not have to be in the Freedom Studio installation SiFive folder.

Discovered Toolchains

Freedom Studio discovers toolchains dynamically each time a list of toolchains is presented to the user. Freedom Studio discovers toolchains by:

1. Searching for any toolchain installations in the installation SiFive folder. You can copy toolchains into this folder, and they will be discovered.
2. Examining the FS_TOOLCHAINS_PATH environment variable. This variable is a "standard" path type variable that can list one or more toolchain folders, each separated by the appropriate path separator character for your host platform.
3. All workspace projects are examined to see if a specific toolchain has been configured.

A list of unique toolchains is constructed from these discovered toolchains and presented to the user.

Using the FS_TOOLCHAINS_PATH environment variable is an easy way to integrate multiple toolchains into Freedom Studio without requiring copying toolchains into each Freedom Studio installation.

Installing and Using an Older Toolchain

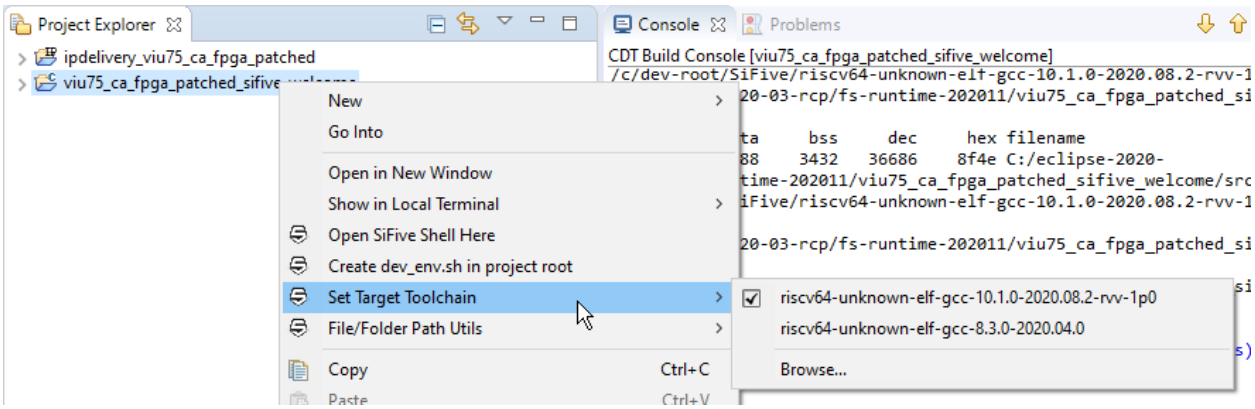
If you do not have a previous version of Freedom Studio installed, or do not have the correct toolchain installed, you can download the toolchain package from sifive.com, install it (anywhere on your system to a path that does not contain spaces), then configure your projects to use that toolchain.

Configure a Project Toolchain

There are two ways to configure which toolchain to use for a project. Both methods simply update the project properties toolchain path setting.

Method 1: Right-click on the project in the Project Explorer

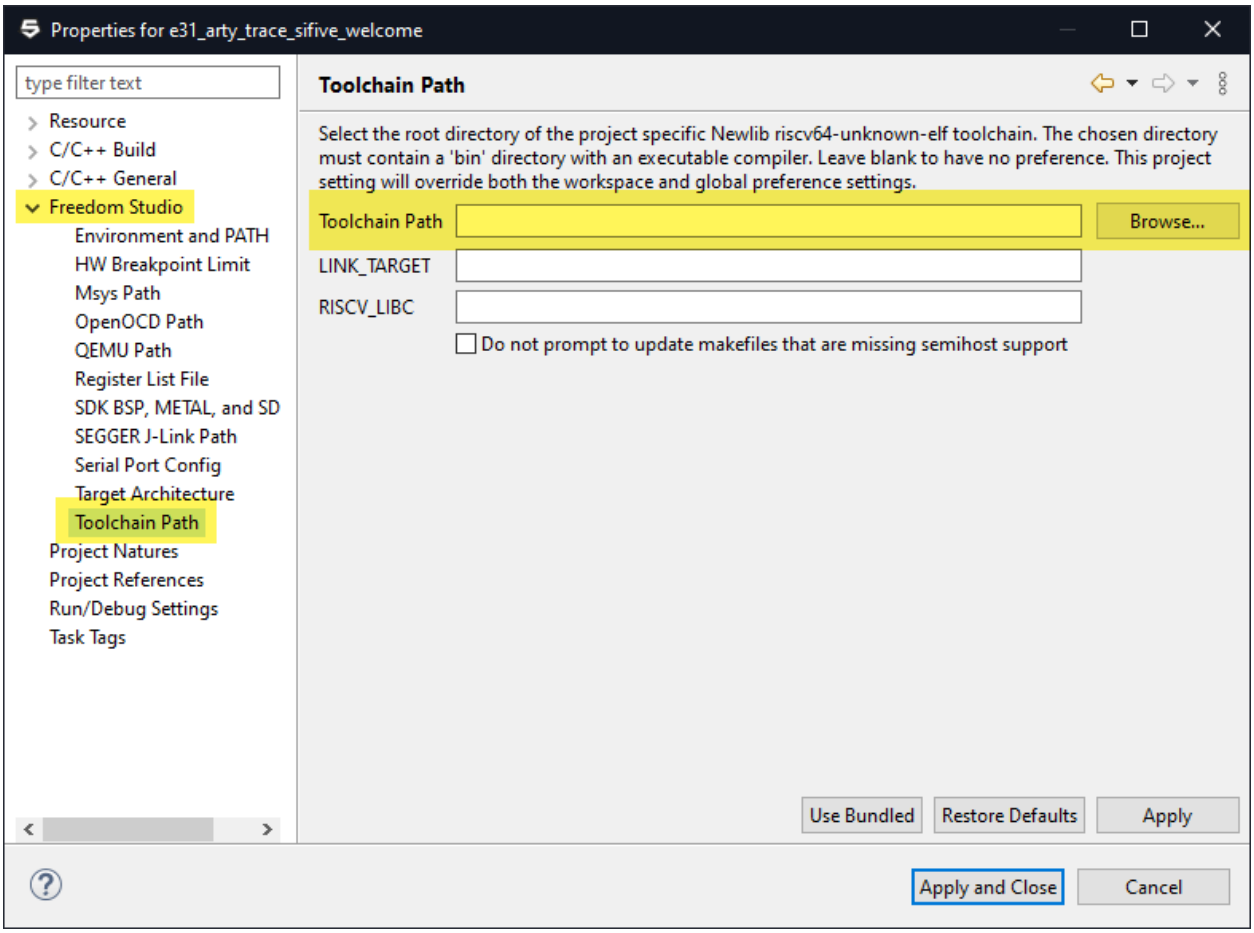
Use the project context menu and select "Set Target Toolchain". This opens a submenu with installed toolchains listed. The current selected toolchain will have a check-mark next to it. Select the toolchain you want to use.



You can also use a toolchain that is not “installed” with or “discovered” by Freedom Studio by using the “Browse…” menu item and navigating to any toolchain folder on the local filesystem.

Method 2:

If not configured otherwise, Freedom Studio will use the newest discovered toolchain. To configure a project to use a different toolchain, right-click on the project node and open the project properties dialog, select the **Freedom Studio | Toolchain Path** node, then use the Browse button to select the root folder of the required toolchain. The required toolchain may reside anywhere on the host system if the path to the toolchain does not contain any whitespace characters. Alternate toolchains do not need to reside within the Freedom Studio installation folder.



What New!

What's New in Freedom Studio 2021.04

- This PDF Manual now contains bookmarks for easy navigation.
- Simpler selection of RISCV_PATH (toolchain), RISCV_LIBC (runtime library) and LINK_TARGET (linker script) from the Project context menu
- State Browser Enhancements
    - Live Variables
    - Global Variables
- Enhanced Toolchain Configuration
- New Project Wizard allow toolchain selection
- Optionally Copy FPGA assets to new projects
- PIB SWT (Serial Wire Trace)
- Improved HW Trigger configuration
    - Trigger address expressions
    - Add triggers from the source editor
- Fixed problems with shared installations on Linux
- Several bug fixes and minor improvements.

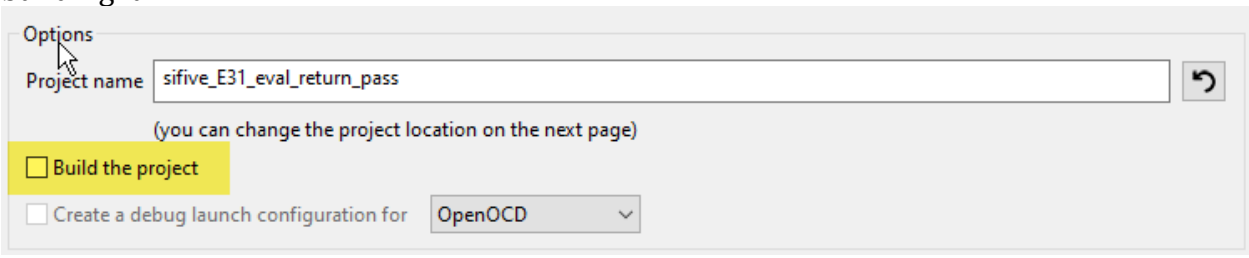
What's New in Freedom Studio 2020.11

Bundled Tools

- OpenOCD
  - Trace SRAM reads are up to 20x faster than previous releases.
  - Runtime PC Sampling
- Toolchain
  - Two toolchains are bundled:
    - gcc version 10.1.0
    - gcc version 8.3.0
- QEMU is updated to 5.1.0

Additional Features & Other Improvements

- Runtime PC Sampling with OpenOCD/Olimex.
- Performance Counter Control and Viewer.
- Process RTL simulator logs into useful information (documentation coming soon).
- Better progress dialogs at launch and during project creation.
- You can import multiple IP projects in a single operation. The only caveat is that you cannot choose a custom name for imported IP projects, only the default generated name will be used.
- You can right-click on an IP project node and create a new software project from there (previously you had to open the project and right-click on the "freedom-e-sdk" folder).
- When using the context menu to program a FPGA image, holding the <shift> key down will start the programming operation immediately, bypassing the Programming Dialog box. Only use this when you know that everything is hooked up properly and ready to go.
- The New Project Wizard has a new option to control whether the new project should be built right-away. Unchecking the box will create the project without building it.



- Several bug fixes, of course!

Known Issues

If you come across other issues not reported here, please let us know on our forum: https://forums.sifive.com/.

When the debugger first connects, I receive a message saying "No source available for address"

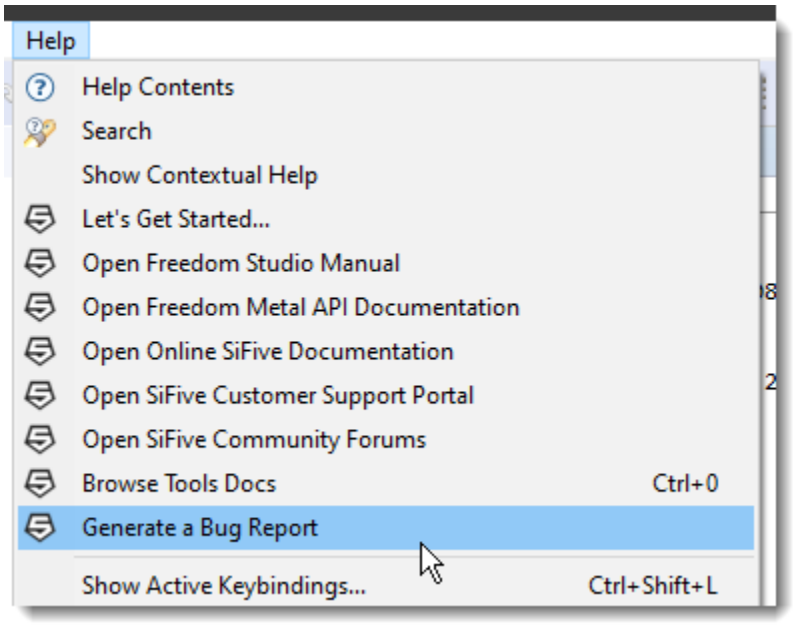
This occurs when instructing the debugger to halt immediately after connecting to the target. It is safe to ignore this message. Stepping/Running the target will work as expected from this point.

Upon starting a debug connection, the Console prints out a lot of text in red colored font

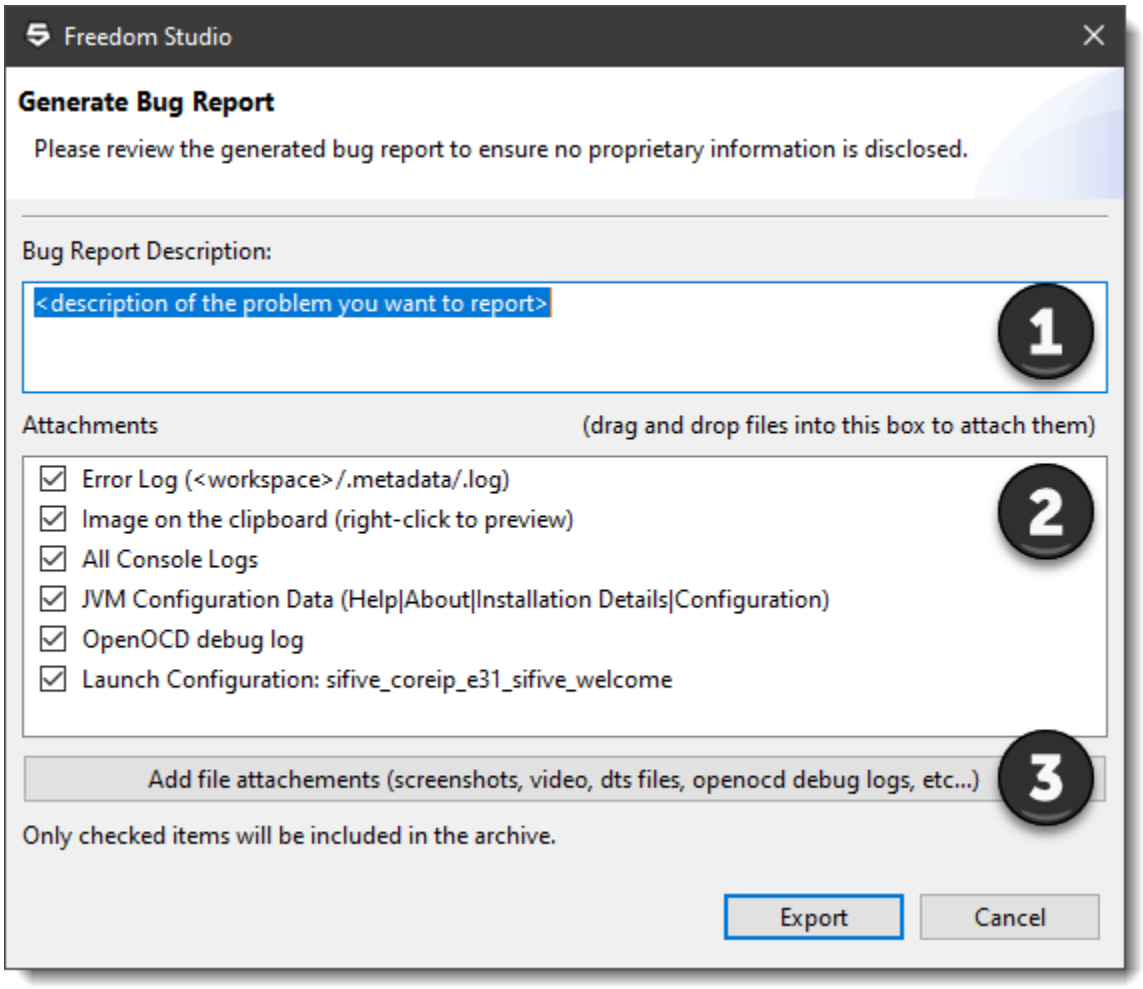
While red font can be scary, it is generally benign debugging output. This happens because OpenOCD output status message through stderr and Freedom Studio renders stderr in red. You can change this color in the Freedom Studio Preference but be aware that this will affect all consoles that accept and display stderr text.

Freedom Studio Bug Report Generator

Freedom Studio gains a Bug Report Generator dialog that will help gather and bundle information for reporting bugs to SiFive support. This dialog is accessible from the Help menu:



Selecting this will open the Bug Report Generator Dialog:



The Bug Report Generator collects information from the Freedom Studio environment that may be useful in diagnosing a problem. You have full control over what information is collected. Please ensure that you are not sending any proprietary information.

The sections of the dialog are described:

1. This text box allows you to provide a description of the problem you are reporting.
2. The Attachments List: This list is prepopulated with several entries gathered from the Freedom Studio Environment:
    a. Error Log: This is the main Freedom Studio application error log. It contains stack traces for any exceptions that occur.
    b. Image on the clipboard: If your system clipboard contains an image it will be detected and added. The intent is to easily include a screenshot showing the problem you are reporting. If you have multiple screenshots, or a video, you can easily attach them as well using the button below the table.
    c. All Console Logs: This will include the text from all console logs (build logs, debug logs, trace console, etc.)

    d. JVM Configuration Data: This log contains a complete snapshot of the process environment Freedom Studio is running. This includes all environment variables, JVM properties, and general host system information.
    e. OpenOCD debug log: If the root of your project contains a file named "openocd-debug.log" then include this file. This is the default log file name used when you use the "Launch OpenOCD Externally" feature to capture an openocd debug log. If your captured log is named differently or located elsewhere you can drag the file into the list or use the "attach" button to attach it.
    f. Launch Configurations: All launch configurations are listed individually for inclusion. These are XML files that fully describe a launch configuration.
3. Add Attachments Button
    a. Press this button to open a file browser from which you can add any additional files to the bug report.
    b. You can also simply drag and drop files from your file explorer application into the attachment list to attach them.

Press the "Export" button to choose a destination folder and export the bug report as a ZIP file. The ZIP file name will be automatically generated using the form:

`freedomstudio_bug_report_<YYYY>_<MM>_<DD>_<HH>_<MM>`

After exporting Freedom Studio will ask if you'd like to visit the Support Portal to submit the bug report. Before submitting, please review the contents of the bug report before submitting it to ensure that you are not providing any proprietary information.

Troubleshooting

Launch fails with "can't add breakpoint"

This can happen if a "bad" breakpoint exists in the breakpoint view prior to the launch. Freedom Studio will try to install the breakpoint and if it is at an address that does not map to the current target, you will get this error. Simply delete this bad breakpoint then relaunch.

Linux USB Permission Issues

By default, some Linux distributions do not give users permissions to access USB devices. The HiFive1 and FPGA getting started guides describe the process to grant your user the correct permissions. For your convenience the *99-openocd.rules* file is included with in the *FreedomStudio/SiFive/Misc* directory.

Correcting Terminal Output

Freedom Studio attempts to ensure terminal output looks correct on all supported platforms. But...

When using the Terminal View in Freedom Studio you may see terminal output from a target UART that does not properly handle “carriage returns”. You may see output that looks like :

To resolve this, open a command window and issue the following command:

```
stty -F <tty-device-name> onlcr inlcr
```

You can do this while connected to the terminal in Freedom Studio. You should see immediate results. You may need to adjust other stty settings depending on your environment.

You may need to experiment with other stty settings to get correct output.

Target Board Setup

Windows Board Setup

This section will describe how to connect SiFive development boards to your Windows computer.

Digilent (on Arty boards) and Olimex devices require specific device drivers to function properly with Freedom Studio. Starting with Freedom Studio 2019.05 these device drivers are automatically installed when needed. There is no need to manually install any device drivers. When a driver is installed you may be prompted by Windows UAC to authorize the installation of the driver.

The device driver for the Digilent USB connection is only installed when you use the Arty Programming utility within Freedom Studio.

If you have used Freedom Studio to update or install an FPGA image and then decide to use Vivado, you will need to uninstall the device driver installed by Freedom Studio before Vivado will recognize the target again. You can choose to have this driver uninstalled automatically at the end of the FPGA programming process, or you can manually uninstall the driver any time from the SiFiveTools menu.

Windows JLink USB Driver

Note: If you have installed JLink software independently of Freedom Studio then the USB driver is already installed.

If you are using a HiFive1-revB board (which has a JLink interface built-in), or if you intend to use a JLink Probe you need to ensure that the JLink USB device driver is installed. Freedom Studio does not install this driver automatically. The driver installation file is located at:

```
<install-folder>/SiFive/jlink/jlink<version-
info>/USBDriver/x64/dpinst_x64.exe
```

Run the installer and accept the default choices.

Alternatively, you can download the installer from Segger directly and run it.

We recommend using the latest JLink software package version. The version bundled in Freedom Studio will work but may not be the latest version.

macOS Board Setup

Cataline & Big Sur

Freedom Studio and OpenOCD do not require any additional setup

Mojave and earlier

By default, macOS has the standard FTDI driver installed while OpenOCD expects to communicate over USB using libusb. To allow OpenOCD to communicate with the SiFive development boards, it is necessary to unload the FTDI driver from macOS.

The procedure to unload the driver is available through the *SiFiveTools -> Setup OpenOCD FTDI Access* menu entry or by typing it manually at the command prompt:

- Open *Applications/Utilities/Terminal*
- Paste in the following command:
  `sudo kextunload -p -b com.apple.driver.AppleUSBFTDI`
- Paste in the following command:
  `sudo kextutil -b com.apple.driver.AppleUSBFTDI -p AppleUSBEFTDI-6010-1`

Note: This is not a permanent solution and after logging out of your computer it is necessary to issue the above commands above.

To avoid having to issue these commands on every log-in, it is possible to add the above commands to your user's *.17ex/.bash_profile*. By doing so, the above commands will be issued automatically every time your user logs in.

To switch back to standard Apple FTDI Access the *SiFiveTools -> Restore Apple FTDI Access* menu entry can be used or again it can be typed manually at the command prompt:

- Open *Applications/Utilities/Terminal*
- Paste in the following command:
  `sudo kextunload -p -b com.apple.driver.AppleUSBFTDI`
- Paste in the following command:
  `sudo kextutil -b com.apple.driver.AppleUSBFTDI`

Linux OS Board Setup

Required Libraries

Important Note Starting with Freedom Studio 2019.08 all dependencies are included or statically linked. This section is only applicable to earlier releases of Freedom Studio.

For Arty board and Olimex support The following libraries need to be installed on the host system:

- libftdi1
- libusb

These can be installed on Ubuntu with the following command:

```
>sudo apt-get install libftdi1-2 libusb-0.1-4 libusb-1.0
```

And on CentOS 7 with the following command:

```
>sudo yum install libftdi libusb
```

And on Fedora 29 with the following command:

```
>sudo yum install libftdi-1.3-12.fc29.x86_64 libusb-1:0.1.5-13.fc29.x86_64
```

Let us Check Our Dependencies

The two programs that require these libraries are `OpenOCD` and `xc3sprog`. You can check that all dependencies are satisfied using the `ldd` utility.

For instance, on Ubuntu:

```
$ cd ~/FreedomStudio/SiFive/xc3sprog/xc3sprog-0.1.2-2019.04.1
$ ldd xc3sprog
    linux-vdso.so.1 =>  (0x00007ffed35f8000)
    libftdi1.so.2 => not found
    libusb-0.1.so.4 => not found
    libstdc++.so.6 => /usr/lib/x86_64-linux-gnu/libstdc++.so.6
(0x00007f395565f000)
    libgcc_s.so.1 => /lib/x86_64-linux-gnu/libgcc_s.so.1 (0x00007f3955447000)
    libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f395507d000)
    libm.so.6 => /lib/x86_64-linux-gnu/libm.so.6 (0x00007f3954d74000)
    /lib64/ld-linux-x86-64.so.2 (0x00007f3955a42000)
$ cd ~/FreedomStudio/SiFive/riscv-openocd/riscv-openocd-0.10.0-2019.05.0-
RC1/bin
$ ldd openocd
    linux-vdso.so.1 =>  (0x00007ffe3cadd000)
    libusb-1.0.so.0 => /lib/x86_64-linux-gnu/libusb-1.0.so.0
(0x00007fe58b0b1000)
    libm.so.6 => /lib/x86_64-linux-gnu/libm.so.6 (0x00007fe58ada8000)
    librt.so.1 => /lib/x86_64-linux-gnu/librt.so.1 (0x00007fe58aba0000)
    libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007fe58a99c000)
    libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0
(0x00007fe58a77f000)
    libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007fe58a3b5000)
    libudev.so.1 => /lib/x86_64-linux-gnu/libudev.so.1 (0x00007fe58b4b2000)
    /lib64/ld-linux-x86-64.so.2 (0x00007fe58b2c9000)
```

While OpenOCD looks good, we can see the we need to install libusb (version 0.1) and libftdi in order to satisfy dependencies for xc3sprog, so let's do that:

```
$ sudo apt-get install libftdi1-2 libusb-0.1-4
  <not showing all the output here>
$ cd ~/FreedomStudio/SiFive/xc3sprog/xc3sprog-0.1.2-2019.04.1
$ ldd xc3sprog
    linux-vdso.so.1 =>  (0x00007ffc051b5000)
    libftdi1.so.2 => /usr/lib/x86_64-linux-gnu/libftdi1.so.2
(0x00007fbded75d000)
    libusb-0.1.so.4 => /lib/x86_64-linux-gnu/libusb-0.1.so.4
(0x00007fbded554000)
    libstdc++.so.6 => /usr/lib/x86_64-linux-gnu/libstdc++.so.6
(0x00007fbded171000)
    libgcc_s.so.1 => /lib/x86_64-linux-gnu/libgcc_s.so.1 (0x00007fbdecf59000)
```

```
    libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007fbdecb8f000)
    libusb-1.0.so.0 => /lib/x86_64-linux-gnu/libusb-1.0.so.0
(0x00007fbdec977000)
    libm.so.6 => /lib/x86_64-linux-gnu/libm.so.6 (0x00007fbdec66e000)
    /lib64/ld-linux-x86-64.so.2 (0x00007fbded96b000)
    libudev.so.1 => /lib/x86_64-linux-gnu/libudev.so.1 (0x00007fbdedb54000)
    libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0
(0x00007fbdec451000)
    librt.so.1 => /lib/x86_64-linux-gnu/librt.so.1 (0x00007fbdec249000)
```

That looks good! Now both openocd and xc3sprog are ready to go.

Enable Access to USB Devices

By default, most Linux distributions do not give users permissions to access USB devices. One either needs root access or to be given the appropriate permissions.

Below are steps you can follow to access your development kit without sudo permissions (although sudo permissions are required for the initial setup):

Step 1: With your board's debug interface connected, make sure your device shows up with the lsusb command:

```
+

$ lsusb
  .
  .
  .
```

With your devices connected, check the output of the `lsusb` command to see that your devices are visible to the system. Use the table below to determine which entry you should see for your devices.

lsusb identifiers

Device	**USB Identifier**
Arty USB	Bus XXX Device XXX: ID 0403:6010 Future Technology Devices International, Ltd FT2232C Dual USB-UART/FIFO IC
HiFive2	Bus 001 Device 019: ID 0403:6011 Future Technology Devices International, Ltd FT4232H Quad HS USB-UART/FIFO IC
Olimex	Bus XXX Device XXX: ID 15ba:002a Olimex Ltd. ARM-USB-TINY-H JTAG interface.
HiFive1 RevB	Bus XXX Device XXX: ID 1366:1051 SEGGER
JLink Probe	Bus XXX Device XXX: ID 1366:0101 SEGGER J-Link PLUS

Step 2: Set the udev rules to allow the device to be accessed by the plugdev group:

Note: For your convenience, a *99-freedomstudio.rules* file is included with Freedom Studio in the FreedomStudio/SiFive/Misc directory. You can install this file with this command:

```
$ sudo cp 99-freedomstudio.rules /etc/udev/rules.d/
```

The 99-freedomstudio.rules files installs rules that recognize the following USB devices and adds them to the plugdev group:

- Olimex ARM_USB_TINY_H
- HiFive2
- Arty Digilent USB

Step 3: See if your board shows up as a serial device belonging to the plugdev group. For instance, with the Arty Board USB connector connected and an Olimex probe connected you should see something like

```
$ ls -l /dev/ttyUSB*
.
.
crw-rw-r-- 1 root plugdev 188, 0 Jun  7 11:01 /dev/ttyUSB0
crw-rw-r-- 1 root plugdev 188, 1 Jun  7 11:01 /dev/ttyUSB1
crw-rw-r-- 1 root plugdev 188, 2 Jun  7 11:07 /dev/ttyUSB2
.
.
```

But how do you know which serial port belongs to which device? You cannot tell from the output above. In fact, there is no simple way to do it, so we have provided a handy shell script called *listusb.sh* located in the *FreedomStudio/SiFive/Misc* directory. Freedom Studio can use this script during a debug launch to automatically select the correct serial port to open.

Running that script yields much enlightenment:

```
$ ./listusb.sh
/dev/ttyUSB1 - Digilent_Digilent_USB_Device_210319A92CC9
/dev/ttyUSB0 - Digilent_Digilent_USB_Device_210319A92CC9
/dev/ttyUSB2 - 15ba_Olimex_OpenOCD_JTAG_ARM-USB-TINY-H_OL150D61
```

Note: If you have other serial devices or multiple boards attached, you may have more devices listed.

The ID (ttyUSB *X*) is assigned dynamically and is dependent on the order in which you connect your devices. Their assignment will change if you disconnect and reconnect in a different order. (But as long as you do not disconnect a device, its assigned ID will not change.)

Note: If your device presents more than a single UART you will always want to select the higher number of the pair. In the example above you would want to use /dev/ttyUSB1

Note: The tty/USB device provided by the Olimex probe cannot be used as a UART. You can ignore this device.

Step 4: Add yourself to the `plugdev` and `dialout` groups. You can use the whoami command to determine your username.

```
> sudo usermod -a -G plugdev `whoami`
```

```
> sudo usermod -a -G dialout `whoami`
```

Log out and log back in, then check that you are now a member of the two groups:

```
$ groups
... plugdev … dialout
```

Now you should be able to access the serial (UART) (dialout) and debug interface (plugdev) without sudo permissions.

SiFive Copyright Notice

Software Licenses

Portions of Freedom Studio are governed by the following licenses

Eclipse Public License - v 2.0

```
    THE ACCOMPANYING PROGRAM IS PROVIDED UNDER THE TERMS OF THIS ECLIPSE
    PUBLIC LICENSE ("AGREEMENT"). ANY USE, REPRODUCTION OR DISTRIBUTION
    OF THE PROGRAM CONSTITUTES RECIPIENT'S ACCEPTANCE OF THIS AGREEMENT.

1. DEFINITIONS

"Contribution" means:

  a) in the case of the initial Contributor, the initial content
     Distributed under this Agreement, and

  b) in the case of each subsequent Contributor:
     i) changes to the Program, and
     ii) additions to the Program;
  where such changes and/or additions to the Program originate from
  and are Distributed by that particular Contributor. A Contribution
  "originates" from a Contributor if it was added to the Program by
  such Contributor itself or anyone acting on such Contributor's behalf.
  Contributions do not include changes or additions to the Program that
  are not Modified Works.

"Contributor" means any person or entity that Distributes the Program.

"Licensed Patents" mean patent claims licensable by a Contributor which
are necessarily infringed by the use or sale of its Contribution alone
or when combined with the Program.

"Program" means the Contributions Distributed in accordance with this
Agreement.

"Recipient" means anyone who receives the Program under this Agreement
or any Secondary License (as applicable), including Contributors.

"Derivative Works" shall mean any work, whether in Source Code or other
form, that is based on (or derived from) the Program and for which the
editorial revisions, annotations, elaborations, or other modifications
represent, as a whole, an original work of authorship.

"Modified Works" shall mean any work in Source Code or other form that
results from an addition to, deletion from, or modification of the
contents of the Program, including, for purposes of clarity any new file
in Source Code form that contains any contents of the Program. Modified
Works shall not include works that contain only declarations,
```

interfaces, types, classes, structures, or files of the Program solely in each case in order to link to, bind by name, or subclass the Program or Modified Works thereof.

"Distribute" means the acts of a) distributing or b) making available in any manner that enables the transfer of a copy.

"Source Code" means the form of a Program preferred for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Secondary License" means either the GNU General Public License, Version 2.0, or any later versions of that license, including any exceptions or additional permissions as identified by the initial Contributor.

2. GRANT OF RIGHTS

a) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, Distribute and sublicense the Contribution of such Contributor, if any, and such Derivative Works.

b) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free patent license under Licensed Patents to make, use, sell, offer to sell, import and otherwise transfer the Contribution of such Contributor, if any, in Source Code or other form. This patent license shall apply to the combination of the Contribution and the Program if, at the time the Contribution is added by the Contributor, such addition of the Contribution causes such combination to be covered by the Licensed Patents. The patent license shall not apply to any other combinations which include the Contribution. No hardware per se is licensed hereunder.

c) Recipient understands that although each Contributor grants the licenses to its Contributions set forth herein, no assurances are provided by any Contributor that the Program does not infringe the patent or other intellectual property rights of any other entity. Each Contributor disclaims any liability to Recipient for claims brought by any other entity based on infringement of intellectual property rights or otherwise. As a condition to exercising the rights and licenses granted hereunder, each Recipient hereby assumes sole responsibility to secure any other intellectual property rights needed, if any. For example, if a third party patent license is required to allow Recipient to Distribute the Program, it is Recipient's responsibility to acquire that license before distributing the Program.

d) Each Contributor represents that to its knowledge it has sufficient copyright rights in its Contribution, if any, to grant the copyright license set forth in this Agreement.

e) Notwithstanding the terms of any Secondary License, no Contributor makes additional grants to any Recipient (other than those set forth in this Agreement) as a result of such Recipient's receipt of the Program under the terms of a Secondary License (if permitted under the terms of Section 3).

3. REQUIREMENTS

3.1 If a Contributor Distributes the Program in any form, then:

a) the Program must also be made available as Source Code, in accordance with section 3.2, and the Contributor must accompany the Program with a statement that the Source Code for the Program is available under this Agreement, and informs Recipients how to obtain it in a reasonable manner on or through a medium customarily used for software exchange; and

b) the Contributor may Distribute the Program under a license different than this Agreement, provided that such license:

i) effectively disclaims on behalf of all other Contributors all warranties and conditions, express and implied, including warranties or conditions of title and non-infringement, and implied warranties or conditions of merchantability and fitness for a particular purpose;

ii) effectively excludes on behalf of all other Contributors all liability for damages, including direct, indirect, special, incidental and consequential damages, such as lost profits;

iii) does not attempt to limit or alter the recipients' rights in the Source Code under section 3.2; and

iv) requires any subsequent distribution of the Program by any party to be under a license that satisfies the requirements of this section 3.

3.2 When the Program is Distributed as Source Code:

a) it must be made available under this Agreement, or if the Program (i) is combined with other material in a separate file or files made available under a Secondary License, and (ii) the initial Contributor attached to the Source Code the notice described in Exhibit A of this Agreement, then the Program may be made available under the terms of such Secondary Licenses, and

b) a copy of this Agreement must be included with each copy of the Program.

3.3 Contributors may not remove or alter any copyright, patent, trademark, attribution notices, disclaimers of warranty, or limitations of liability ("notices") contained within the Program from any copy of the Program which they Distribute, provided that Contributors may add their own appropriate notices.

4. COMMERCIAL DISTRIBUTION

Commercial distributors of software may accept certain responsibilities with respect to end users, business partners and the like. While this license is intended to facilitate the commercial use of the Program, the Contributor who includes the Program in a commercial product offering should do so in a manner which does not create potential liability for other Contributors. Therefore, if a Contributor includes the Program in a commercial product offering, such Contributor ("Commercial Contributor") hereby agrees to defend and indemnify every other Contributor ("Indemnified Contributor") against any losses, damages and costs (collectively "Losses") arising from claims, lawsuits and other legal actions brought by a third party against the Indemnified Contributor to the extent caused by the acts or omissions of such Commercial Contributor in connection with its distribution of the Program in a commercial product offering. The obligations in this section do not apply to any claims or Losses relating to any actual or alleged intellectual property infringement. In order to qualify, an Indemnified Contributor must: a) promptly notify the Commercial Contributor in writing of such claim, and b) allow the Commercial Contributor to control, and cooperate with the Commercial Contributor in, the defense and any related settlement negotiations. The Indemnified Contributor may participate in any such claim at its own expense.

For example, a Contributor might include the Program in a commercial product offering, Product X. That Contributor is then a Commercial Contributor. If that Commercial Contributor then makes performance claims, or offers warranties related to Product X, those performance claims and warranties are such Commercial Contributor's responsibility alone. Under this section, the Commercial Contributor would have to defend claims against the other Contributors related to those performance claims and warranties, and if a court requires any other Contributor to pay any damages as a result, the Commercial Contributor must pay those damages.

5. NO WARRANTY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, AND TO THE EXTENT PERMITTED BY APPLICABLE LAW, THE PROGRAM IS PROVIDED ON AN "AS IS"

BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Each Recipient is solely responsible for determining the appropriateness of using and distributing the Program and assumes all risks associated with its exercise of rights under this Agreement, including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and unavailability or interruption of operations.

6. DISCLAIMER OF LIABILITY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, AND TO THE EXTENT PERMITTED BY APPLICABLE LAW, NEITHER RECIPIENT NOR ANY CONTRIBUTORS SHALL HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OR DISTRIBUTION OF THE PROGRAM OR THE EXERCISE OF ANY RIGHTS GRANTED HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. GENERAL

If any provision of this Agreement is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this Agreement, and without further action by the parties hereto, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

If Recipient institutes patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Program itself (excluding combinations of the Program with other software or hardware) infringes such Recipient's patent(s), then such Recipient's rights granted under Section 2(b) shall terminate as of the date such litigation is filed.

All Recipient's rights under this Agreement shall terminate if it fails to comply with any of the material terms or conditions of this Agreement and does not cure such failure in a reasonable period of time after becoming aware of such noncompliance. If all Recipient's rights under this Agreement terminate, Recipient agrees to cease use and distribution of the Program as soon as reasonably practicable. However, Recipient's obligations under this Agreement and any licenses granted by Recipient relating to the Program shall continue and survive.

Everyone is permitted to copy and distribute copies of this Agreement, but in order to avoid inconsistency the Agreement is copyrighted and may only be modified in the following manner. The Agreement Steward

reserves the right to publish new versions (including revisions) of this Agreement from time to time. No one other than the Agreement Steward has the right to modify this Agreement. The Eclipse Foundation is the initial Agreement Steward. The Eclipse Foundation may assign the responsibility to serve as the Agreement Steward to a suitable separate entity. Each new version of the Agreement will be given a distinguishing version number. The Program (including Contributions) may always be Distributed subject to the version of the Agreement under which it was received. In addition, after a new version of the Agreement is published, Contributor may elect to Distribute the Program (including its Contributions) under the new version.

Except as expressly stated in Sections 2(a) and 2(b) above, Recipient receives no rights or licenses to the intellectual property of any Contributor under this Agreement, whether expressly, by implication, estoppel or otherwise. All rights in the Program not expressly granted under this Agreement are reserved. Nothing in this Agreement is intended to be enforceable by any entity that is not a Contributor or Recipient. No third-party beneficiary rights are created under this Agreement.

Exhibit A - Form of Secondary Licenses Notice

"This Source Code may also be made available under the following Secondary Licenses when the conditions for such availability set forth in the Eclipse Public License, v. 2.0 are satisfied: {name license(s), version(s), and exceptions or additional permissions here}."

Simply including a copy of this Agreement, including this Exhibit A is not sufficient to license the Source Code under Secondary Licenses.

If it is not possible or desirable to put the notice in a particular file, then You may include the notice in a location (such as a LICENSE file in a relevant directory) where a recipient would be likely to look for such a notice.

You may add additional accurate notices of copyright ownership.

GNU GENERAL PUBLIC LICENSE, V2

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Copyright (C) 1989, 1991 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.

Preamble

The licenses for most software are designed to take away your

freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this

License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to

control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program 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.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to

refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public 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.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

```
    You should have received a copy of the GNU General Public License along
    with this program; if not, write to the Free Software Foundation, Inc.,
    51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this
when it starts in an interactive mode:

    Gnomovision version 69, Copyright (C) year name of author
    Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License.  Of course, the commands you use may
be called something other than `show w' and `show c'; they could even be
mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the program, if
necessary.  Here is a sample; alter the names:

  Yoyodyne, Inc., hereby disclaims all copyright interest in the program
  `Gnomovision' (which makes passes at compilers) written by James Hacker.

  <signature of Ty Coon>, 1 April 1989
  Ty Coon, President of Vice

This General Public License does not permit incorporating your program into
proprietary programs.  If your program is a subroutine library, you may
consider it more useful to permit linking proprietary applications with the
library.  If this is what you want to do, use the GNU Lesser General
Public License instead of this License.
```

GNU GENERAL PUBLIC LICENSE, V3

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 3, 29 June 2007

 Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

  The GNU General Public License is a free, copyleft license for
software and other kinds of works.
```

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents.

States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

  The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

  0. Definitions.

  "This License" refers to version 3 of the GNU General Public License.

  "Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

  "The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

  To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

  A "covered work" means either the unmodified Program or a work based on the Program.

  To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

  To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

  An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a

covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product

(including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial

commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of

it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

  Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

    a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

    b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

    c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

    d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

    e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

    f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

  All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

  If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

  Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions;

the above requirements apply either way.

  8. Termination.

  You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

  However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

  Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

  Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

  9. Acceptance Not Required for Having Copies.

  You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

  10. Automatic Licensing of Downstream Recipients.

  Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

  An "entity transaction" is a transaction transferring control of an

organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means,

then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public 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.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> <one line to give the program's name and a brief idea of what it does.>
> Copyright (C) <year> <name of author>
>
> This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

  If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
    <program>  Copyright (C) <year>  <name of author>
    This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.
```

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

  You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

  The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.