



Fast Models Tools

Version 11.30

User Guide

Non-Confidential

Copyright © 2024–2025 Arm Limited (or its affiliates).
All rights reserved.

Issue 00

109415_1130_00_en



Fast Models Tools User Guide

This document is Non-Confidential.

Copyright © 2024–2025 Arm Limited (or its affiliates). All rights reserved.

This document is protected by copyright and other intellectual property rights.

Arm only permits use of this document if you have reviewed and accepted [Arm's Proprietary Notice](#) found at the end of this document.

This document (109415_1130_00_en) was issued on 2025-11-19. There might be a later issue at <https://developer.arm.com/documentation/109415>

The product version is 11.30.

See also: [Proprietary Notice](#) | [Product and document information](#) | [Useful resources](#)

Start reading

If you prefer, you can skip to [the start of the content](#).

Intended audience

This document is written for software developers using the Fast Models tools to create, build, and debug custom Fast Models systems.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

This document includes language that can be offensive. We will replace this language in a future issue of this document.

To report offensive language in this document, email terms@arm.com.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on <https://support.developer.arm.com>.

To provide feedback on the document, fill the following survey: <https://developer.arm.com/documentation-feedback-survey>.

Contents

1. Fast Models tools.....	5
1.1 Fast Models design flow.....	5
1.2 Project files.....	7
1.3 Repository files.....	9
1.4 File processing order.....	10
1.5 Hierarchical systems.....	11
2. System Generator.....	14
2.1 SimGen command-line options.....	14
2.2 SimGen project (sgproj) file format.....	17
2.3 SimGen detection of duplicate connections.....	22
3. System Canvas.....	23
3.1 System Canvas tutorial.....	23
3.1.1 Starting System Canvas.....	23
3.1.2 Creating a new project.....	24
3.1.3 Adding the Arm® processor.....	27
3.1.4 Naming components.....	28
3.1.5 Resizing components.....	28
3.1.6 Hiding ports.....	28
3.1.7 Moving ports.....	29
3.1.8 Adding components.....	29
3.1.9 Using port arrays.....	30
3.1.10 Connecting components.....	31
3.1.11 View project properties and settings.....	31
3.1.12 Changing the address mapping.....	34
3.1.13 Building the system.....	36
3.1.14 Building a SystemC ISIM target.....	37
3.2 System Canvas GUI.....	38
3.2.1 Menu bar.....	39
3.2.2 Toolbar.....	46
3.2.3 Workspace window.....	48
3.2.4 Component window.....	51

3.2.5 Output window.....	52
3.2.6 System Canvas dialogs.....	53
4. Iris Monitor.....	82
4.1 Key features of Iris Monitor.....	82
4.2 Retargetable debugger.....	82
4.3 Get started with Iris Monitor.....	83
4.4 Iris Monitor GUI.....	85
4.4.1 Simulation status and control.....	86
4.4.2 Disassembly view.....	87
4.4.3 Breakpoints view.....	89
4.4.4 Registers view.....	90
4.4.5 Memory view.....	90
4.4.6 Instances view.....	91
4.4.7 Overview view.....	92
4.4.8 Parameters view.....	93
4.5 Iris Monitor command-line options.....	94
4.6 Iris Monitor tutorial.....	94
Proprietary Notice.....	99
Product and document information.....	101
Product status.....	101
Revision history.....	101
Conventions.....	102
Useful resources.....	104

1. Fast Models tools

Fast Models tools enable you to create custom system models from the Fast Models portfolio of component models, and debug them.

System Generator or `simgen`

A backend tool that handles system model generation. System Generator can either be invoked from the System Canvas GUI, or by using the `simgen` command-line utility. System models that are created using System Generator can be used with other Arm® development tools, for example Arm® Development Studio or Iris Monitor, or can be exported to SystemC for integration with proprietary models.

System Canvas or `sgcanvas`

A GUI design tool for developing new model components written in LISA+ and for creating and building system models. To launch System Canvas from the command line, type `sgcanvas`. The GUI displays the model as either LISA+ source code, or graphically, in a block diagram editor.

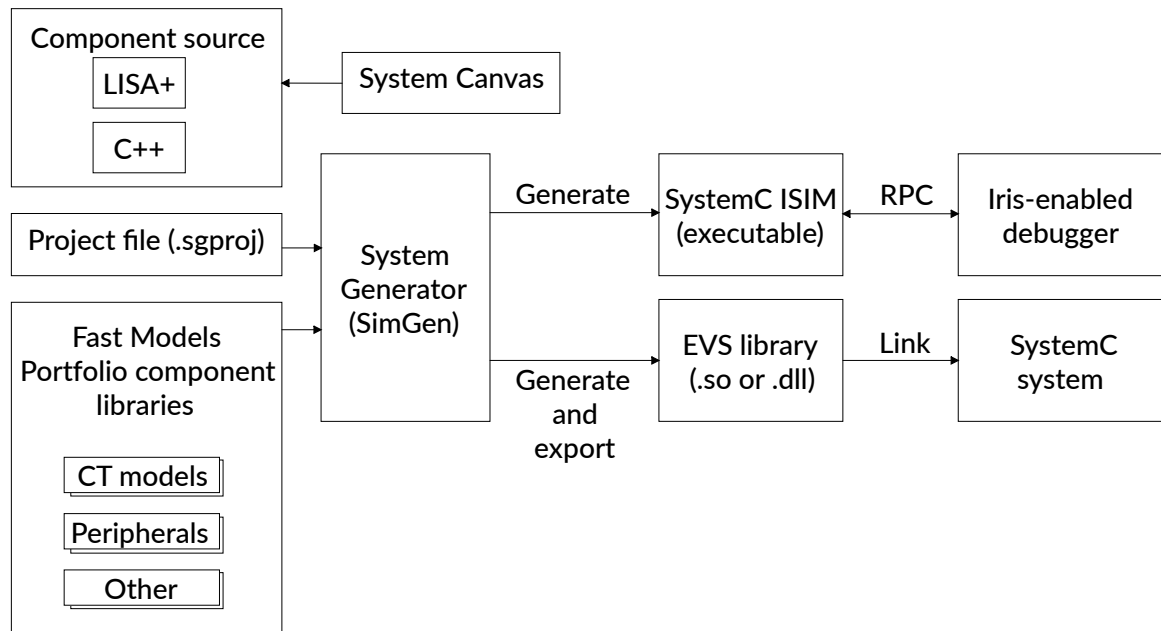
Iris Monitor

A browser-based debugger for Fast Models that lets you debug and view the state of the Fast Models components in a system.

1.1 Fast Models design flow

This section describes the process of designing a Fast Models system.

1. Create or license standard component models.
2. Use System Canvas to connect components and set parameters in the LISA+ source code.
3. Generate a new system model using System Generator either from the command line, using `simGen`, or using the System Canvas GUI.
4. Use the new model as input to a more complex system or distribute it as a standalone simulation environment.

Figure 1-1: Fast Models design flow

The inputs to System Generator are:

C++ library objects

Typically these are models of processors or standard peripherals.

LISA+ source code

The source code files define custom peripheral components. They can be existing files in the Fast Models portfolio or new LISA+ files that were created in System Canvas. The LISA+ descriptions can be located in any directory. One LISA+ file can contain one or more component descriptions.

Project file

System Generator requires a `.sgproj` project file to configure the build.

After the required components have been added and connected, System Generator uses gcc or the Visual Studio C++ compiler to produce the output object as one of the following:

- An ISIM executable, for instance an FVP. You could run this standalone, or you could connect an Iris-enabled debugger to it, such as Arm® Development Studio Debugger or Iris Monitor.
- An EVS, which can be used as a building block for a SystemC system. It is generated using the Fast Models SystemC Export feature.



To build ISIM executables or EVSs, you must have installed a SystemC environment and set the `SYSTEMC_HOME` environment variable to it.

Related information

[SimGen project \(.sgproj\) file format](#) on page 17

1.2 Project files

A single project file (`.sgproj`) describes to System Generator (SimGen) the build configuration to use for each host platform and the files that are required to build the model.

The build configuration includes:

- The compiler version to use.
- Whether to build release or debug binaries.
- Linker and compiler flags.
- SimGen flags.
- Build target, for example EVS library or ISIM.

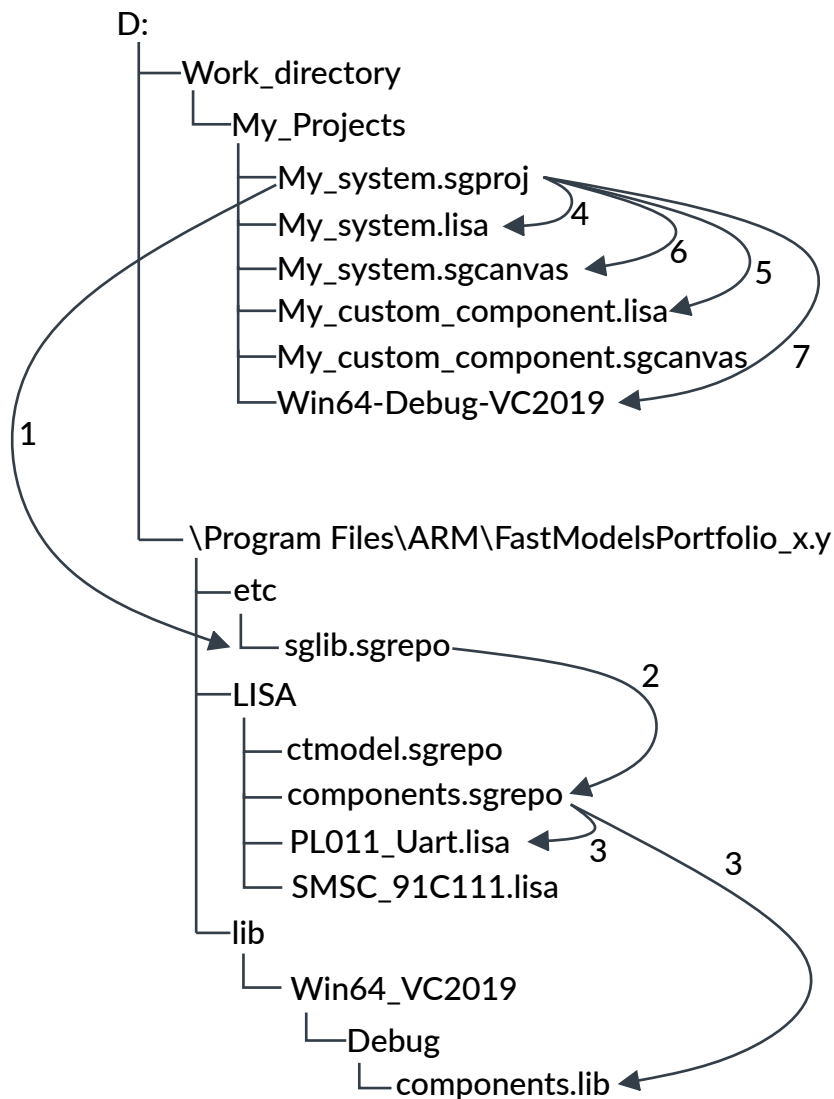
It also specifies the location of the LISA source files for the project.

There is no requirement to provide a makefile and a set of configuration files for each new project.

Each project file references all files that System Canvas or SimGen needs to build and run a simulation, including LISA, C, and C++ sources, libraries, files to deploy to the simulation directory, and nested repository files.

Repository files (`.sgrepo`) have the same format as project files.

You can add single files or a complete repository, such as the Fast Models Portfolio, to the project file.

Figure 1-2: Example organization of project directories and files on Microsoft Windows

The `My_Projects` directory contains the `My_System.sgproj` project file:

1. `My_system.sgproj` points to the standard Fast Models Portfolio repository file `sglib.sgrepo`.
2. The `sglib.sgrepo` repository file contains a list of repository locations such as `components.sgrepo`.
3. `components.sgrepo` lists the locations of the LISA files for the components and the location and type of libraries that are available for the components.
4. The project file lists `My_system.lisa` as the top-level LISA file for the system. The top-level LISA file lists the components in the system and shows how they interconnect.
5. This project uses a custom component in addition to the standard Fast Models Portfolio components. Custom components can exist anywhere in the directory structure. In this case,

only the `My_system` component uses the custom component, so the `My_custom_component.lisa` file is in the same directory.

6. System Canvas generates the `My_system.sgcanvas` and `My_custom_component.sgcanvas` files to save changes made to the display settings in the **Workspace** window. The display settings include:
 - Component location and size.
 - Label text, position and formatting.
 - Text font and size.
 - The moving of or hiding of ports.
 - Grid spacing.

The build process does not use `.sgcanvas` files. System Canvas uses them for its **Block Diagram** view.

7. `My_system.sgproj` defines `win64-Debug-vc2019` as the build directory for the selected platform. Other build options in the project file include:
 - The host platform, for instance "win64".
 - The compiler, for example "vc2019" and compiler options.
 - Additional linker options.
 - Additional options to be passed to `simGen`.
 - The type of target to build, for example an ISIM executable or a SystemC component.

Related information

[Project Settings dialog](#) on page 73

[SimGen project \(sgproj\) file format](#) on page 17

1.3 Repository files

Repository files group together references to commonly used files, eliminating the need to specify the path and library for each component in a project.

Repository files contain:

- A list of components.
- The paths to the LISA sources for the components.
- A list of library objects for the components.
- Optionally, lists of paths to other repository files. This enables a hierarchical structure.

System Canvas adds the default model repositories to a project when creating it. Changing these repository settings does not affect existing projects. The `project_name.sgproj` files contain the paths to the repositories as hard code. To change the repositories for an existing project, open the file and edit the paths.

Default repositories can also preset required configuration parameters for projects that rely on the default model library. These parameters are:

- Additional Include Directories.
- Additional Compiler Settings.
- Additional Linker Settings.

1.4 File processing order

The processing order enables a custom implementation of a Fast Models component.

An example of a project file

```
/// project file
sgproject "MyProject.sgproj"
{
  files
  {
    path = "./MyTopComponent.lisa";
    path = "./MySubComponent1.lisa";
    path = "./repository.sgrepo";
    path = "./MySubComponent2.lisa";
  }
}
```

An example of a repository file

```
/// subrepository file
sgproject "repository.sgrepo"
{
  files
  {
    path = "../LISA/ASubComponent1.lisa";
    path = "../LISA/ASubComponent2.lisa";
  }
}
```

System Canvas processes the files in sequence, expanding sub-repositories as it encounters them:

1. ./MyTopComponent.lisa
2. ./MySubComponent1.lisa
3. ./repository.sgrepo
 - a. ../LISA/ASubComponent1.lisa
 - b. ../LISA/ASubComponent2.lisa
4. ./MySubComponent2.lisa

Changing the processing order allows customization. If `MySubComponent1.lisa` and `../LISA/ASubComponent1.lisa` both list a component with the same name, the application uses only the first definition.

The **File List** view of System Canvas shows the order of components in the project file. Use the application controls to re-order the files and repositories:

- The **Up** and **Down** context menu entries in the **File List** view of the **Component** window. The commands have keyboard shortcuts of **Alt+Arrow Up** and **Alt+Arrow Down**.

You can also drag-and-drop files inside a repository or between repositories.

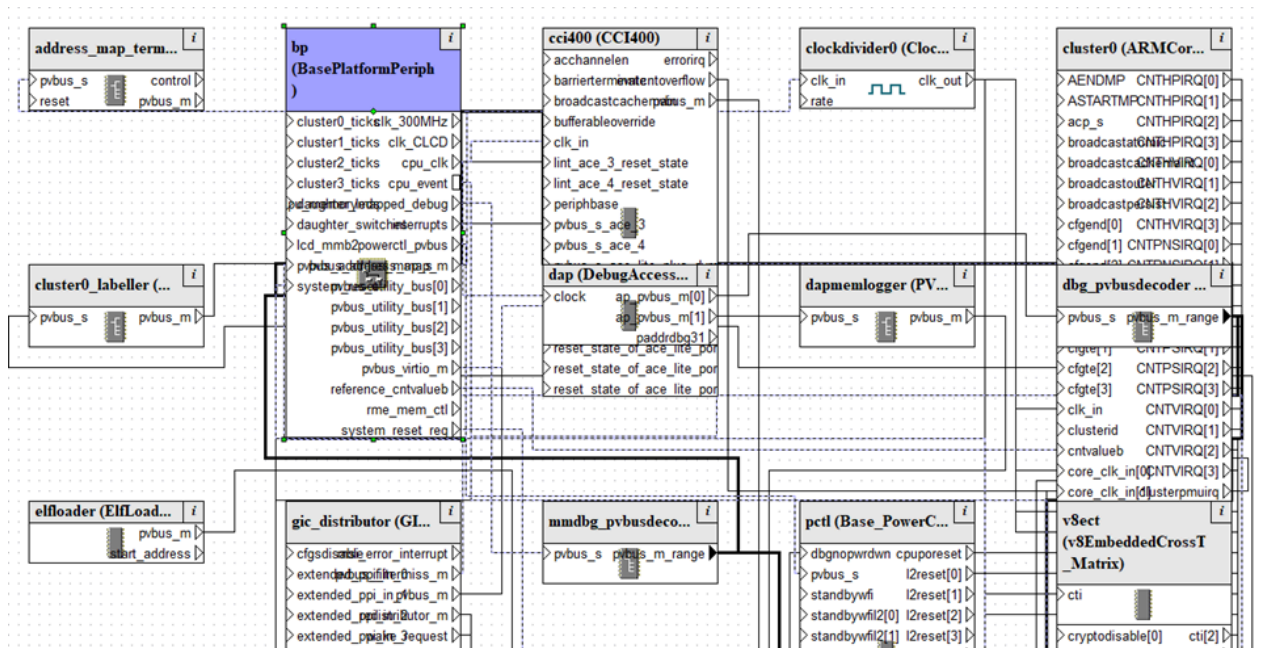
- The **Up** and **Down** buttons on the **Default Model Repository** tab in the **Properties** dialog, for repositories in new projects.

1.5 Hierarchical systems

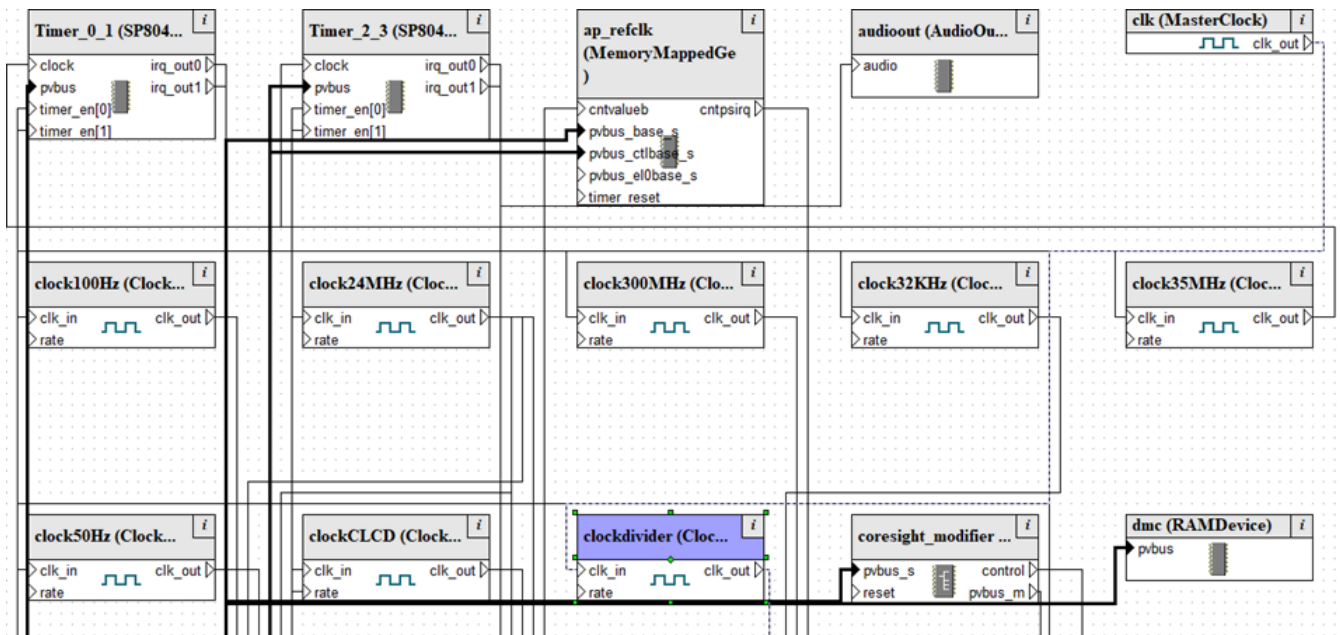
A Fast Models system is composed of a hierarchy of components and subcomponents. The following block diagrams show the advantage of using a hierarchical system with a complex model, in this example FVP Base Cortex X1.

In System Canvas, select the **File > Load project ...** menu to load one of the example projects, for example `$PVLIB_HOME/examples/LISA/FVP_Base/Build_Cortex-X1/FVP_Base_Cortex-X1.sgproj`. By default, the model is displayed in the **Block Diagram** view:

Figure 1-3: Block diagram of top-level Base Platform model

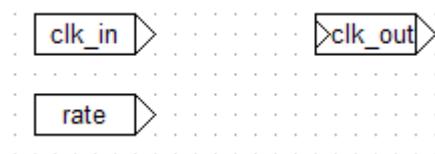


To open the BasePlatformPeripherals component, double click it. It is a complex object with many subcomponents:

Figure 1-4: Contents of BasePlatformPeripherals component

Hiding the complexity of the BasePlatformPeripherals in a component simplifies the diagram and enables the BasePlatformPeripherals component to be shared between different FVPs.

Double click the `clockDivider` subcomponent highlighted in [Figure 1-4: Contents of BasePlatformPeripherals component](#) on page 12. The **Block Diagram** window displays the external ports for this subcomponent:

Figure 1-5: Clock divider component external ports

The `clockDivider` contains only external ports, and it has no subcomponents. The behavior for this component is determined by the LISA+ code, which you can view by clicking the **Source** tab at the bottom of the window

A component communicates with components in the higher-level system through its self ports. Self ports refer to ports in a system that are not part of a subcomponent, and are represented by a hollow rectangle with triangles to indicate data flow, and a text label in the rectangle.

Self ports can be internal or external.

Internal ports

These ports communicate with subcomponents and are not visible if the component is used in a higher-level system. Unlike hidden external ports, you cannot expose internal ports

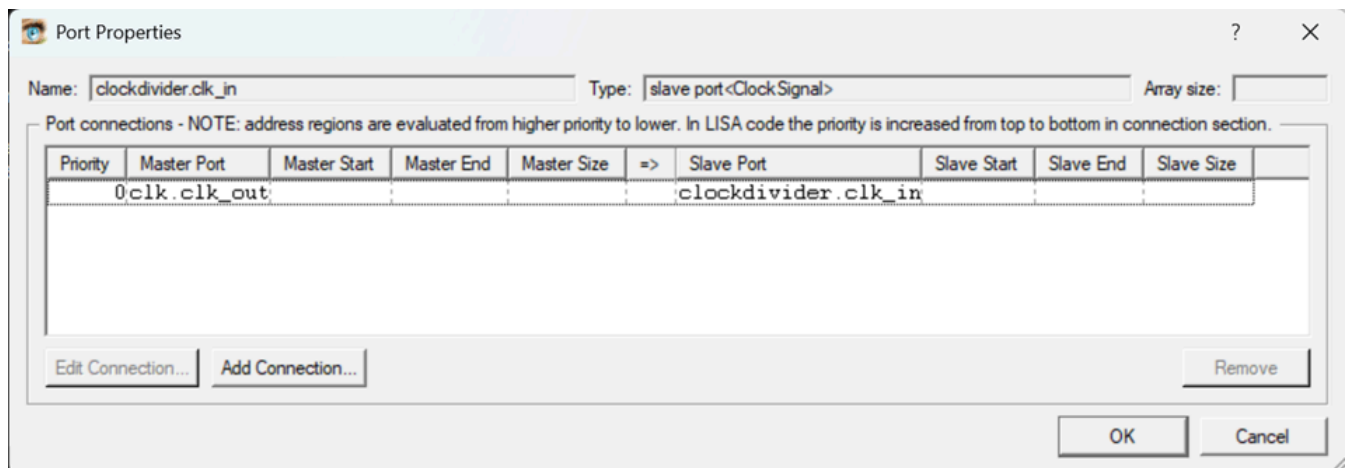
outside the subcomponent. Right-click on a port and select **Object Properties...** to identify or create internal ports. Set the port attributes to **Internal** for an internal self port.

External ports

These ports communicate with components in a higher-level system, and by default are external.

For example, the `clockdivider.clk_in` port is connected to an external port called `clk.clk_out`. To see details about the connection, in the **Block Diagram** window for the `BasePlatformPeripherals` component, double click the `clk_in` port for `clockdivider`. This displays the **Port Properties** dialog:

Figure 1-6: Port Properties dialog



If you use the **Block Diagram** editor to make a connection between an external port and a subcomponent, the LISA code uses the keyword `self` to indicate the standalone port:

```
self.clk_in_master => clkdiv_ref25.clk_in;
```

2. System Generator

Platform models are built using a tool called System Generator, also called SimGen, and a Fast Models `.sgproj` project file.

You can use SimGen in the following ways:

- By building the project in System Canvas, which invokes SimGen indirectly.
- By directly invoking `simgen` on the command line.



To use SimGen, you must have installed a supported version of GCC or Visual Studio C++ compiler. On Windows, if SimGen cannot find `devenv.exe` for Visual Studio, the build fails. You can specify the path to `devenv.exe` in the System Canvas **Preferences** dialog or using the `--devenv-path` command-line option.

A SimGen command to build a Fast Models project might use the following options:

```
simgen --build --project-file <proj_file>.sgproj --configuration <config_name>
```

where `config_name` is the build configuration, for example `Linux64-Release-GCC-9.3`, `Win64-Debug-VC2019`, OR `Linux64_armv81-Release-GCC-9.3`.



A build configuration name beginning `Linux64_armv81` indicates an Arm® AArch64 Linux host.

2.1 SimGen command-line options

This table documents all System Generator (SimGen) options.

The command for invoking SimGen is:

```
simgen <options> [<additional-file-list>]
```

where:

<options>

SimGen command-line options.

<additional-file-list>

Optional, space-separated list of `.lisa` and `.sgrepo` files appended to the SimGen command line, in any order. `additional-file-list` enables you to build different variants of a platform using a single, common `.sgproj` file. List the files that are common to all variants in the `files`

section of the `.sgproj` file, and use `additional-file-list` to list the files that are specific to the platform variant.

Table 2-1: SimGen command-line options

Option	Short form	Description
<code>--allow-deprecated</code>	-	Allow the use of components marked as deprecated in the LISA+ component properties section.
<code>--allow-prerelease</code>	-	Suppress warning about pre-release model quality. If not specified, SimGen outputs a warning when it builds a model containing one or more pre-release <code>modelquality</code> components. Pre-release means the quality level is either preliminary support or alpha support. For more information about quality levels, see Quality level definitions in the Fast Models Reference Guide.
<code>--bridge-conf-file <FILENAME></code>	-	Set auto-bridging JSON configuration file <i>FILENAME</i> .
<code>--build</code>	<code>-b</code>	Build the targets.
<code>--build-directory <DIR></code>	-	Set build directory <i>DIR</i> .
<code>--clean</code>	<code>-C</code>	Clean the targets.
<code>--config <FILENAME></code>	-	Set SimGen configuration file <i>FILENAME</i> . By default, <code>simgen.conf</code> .
<code>--configuration <NAME></code>	-	The name of the build configuration, for example <code>Linux64-Release-GCC-9.3</code> .
<code>--cpp-flags-start</code>	-	Ignore all parameters between this and <code>--cpp-flags-end</code> , except <code>-D</code> and <code>-I</code> .
<code>--cpp-flags-end</code>	-	See <code>--cpp-flags-start</code> .
<code>--cxx-flags-start</code>	-	Ignore all parameters between this and <code>--cxx-flags-end</code> , except <code>-D</code> .
<code>--cxx-flags-end</code>	-	See <code>--cxx-flags-start</code> .
<code>--debug</code>	<code>-d</code>	Enable debug mode.
<code>--define <SYMBOL></code>	<code>-D</code>	Define preprocessor <i>SYMBOL</i> . You can also use <code>SYMBOL=DEF</code> .
<code>--devenv-path <ARG></code>	-	Windows only. Path to Visual Studio development environment, <code>devenv</code> .
<code>--disable-warning <NUM></code>	-	Disable warning number <i>NUM</i> . This overrides the <code>--warning-level</code> option.
<code>--dumb-term</code>	-	The terminal in which SimGen is running is dumb, so instead of fancy progress indicators, use simpler ones.
<code>--enable-warning <NUM></code>	-	Enable warning number <i>NUM</i> . This option overrides the <code>--warning-level</code> option.
<code>--gcc-path <PATH></code>	-	Linux only. Full path of the GCC C++ compiler to build the model. Ensure the compiler version matches the GCC version in the model configuration. By default, SimGen uses the g++ in the search path.
<code>--gen-sysgen-lib</code>	-	Generate system library.

Option	Short form	Description
--help	-h	Print help message with a list of command-line options then exit.
--ignore-compiler-version	-	Windows only. Do not stop on a compiler version mismatch. Try to build anyway.
--include <INC_PATH>	-I	Add include path <i>INC_PATH</i> .
--indir_tpl <DIR>	-	Set directory <i>DIR</i> where SimGen finds its template data files.
--link-against <LIBS>	-	Whether the final executable will be linked against debug or release libraries. <i>LIBS</i> can be either debug or release. This option performs some consistency checks.
--MSVC-debuginfo-type <ARG>	-	Set the type of debug information for MSVC projects. <i>ARG</i> can be one of: <ul style="list-style-type: none"> • none: No debug information. • /zi: Program database. • /Zd: Line numbers only.
--no-deploy	-	Prevent SimGen from copying deployed files from their original location to the location of the model. For example, when this option is used, SimGen does not copy <i>armctmodel.dll</i> or <i>libarmctmodel.so</i> from the model library to the location of the generated model. This option is for advanced users who are building models in a batch system, or as part of another tool where you are responsible for making sure all the required libraries are present.
--no-lineinfo	-c	Do not generate line number redirection in generated source and header files.
--num-build-cpus <NUM>	-	The number of host CPUs to use for the build.
--num-comps-file <NUM>	-	The number of components SimGen places into each C++ file. The default is one, which means SimGen places each component in a separate C++ file, along with generic code. A higher value reduces the total amount of generic code, which can reduce the compilation time when building a whole project. A lower value can reduce the compilation time if only some components have changed and need recompilation.
--outdir_arch <DIR>	-	Set output directory <i>DIR</i> for file with variable filenames.
--outdir_fixed <DIR>	-	Set output directory <i>DIR</i> for file with constant filenames.
--override-config-parameter <PARAM>=<VALUE>	-P	Override a configuration parameter defined in the <i>.sgproj</i> file. To override multiple parameters, specify this option multiple times. This option also supports appending a string to a parameter value defined in the <i>.sgproj</i> file, by using the syntax <i><PARAM>+=<STRING></i> . For example: <i>-P ADDITIONAL_COMPILER_SETTINGS+="-Wnew-warning"</i>
--print-config	-	Print the configuration parameters to file <i>.ConfigurationParameters.txt</i> .
--print-preprocessor-output	-E	Print preprocessor output, then exit.

Option	Short form	Description
<code>--print-resource-mapping</code>	-	Print flat resource mapping when generating a simulator.
<code>--project-file <FILENAME></code>	<code>-p</code>	Set SimGen project file <i>FILENAME</i> .
<code>--replace-strings</code>	-	Replace one or more strings in one or more files, then exit. Ignore binary files. Usage: <code>simgen --replace-strings FROM1 TO1 [FROM2 TO2 ...] -- FILE1 [FILE2 ...]</code>
<code>--replace-strings-bin</code>	-	Replace one or more strings in one or more files, then exit. Do not ignore binary files. Usage: <code>simgen --replace-strings-bin FROM1 TO1 [FROM2 TO2 ...] -- FILE1 [FILE2 ...]</code>
<code>--top-component <COMP></code>	-	Set the top-level component.
<code>--user-MSVC-libs-start</code>	-	Set additional libraries for MSVC projects. The list is terminated by <code>--user-MSVC-libs-end</code> .
<code>--user-MSVC-libs-end</code>	-	See <code>--user-MSVC-libs-start</code> .
<code>--user-sourcefiles-start</code>	-	Add source files listed between this option and <code>--user-sourcefiles-end</code> to the executable.
<code>--user-sourcefiles-end</code>	-	See <code>--user-sourcefiles-start</code> .
<code>--verbose <ARG></code>	<code>-v</code>	Set SimGen verbosity to either <i>on</i> , <i>sparse</i> (default), or <i>off</i> .
<code>--version</code>	<code>-V</code>	Print the version and exit.
<code>--warning-level <LEVEL></code>	<code>-w</code>	Set the warning level. Possible values are 0-3, where 0 means no warnings and 3 means all warnings. The default is 2, for all useful warnings.
<code>--warnings-as-errors</code>	-	Treat LISA parsing and compiler warnings as errors.

2.2 SimGen project (sgproj) file format

A SimGen project file, or `sgproj` file for short, is used to configure a Fast Models build. The filename has a `.sgproj` extension.

Specify the project file in one of the following ways:

- When using SimGen, use the `--project-file` option, or the short version `-p`.
- When using System Canvas, select **File > Load Project**.

Syntax

The `sgproj` file defines:

- A build configuration for each supported build target.
- The default configuration to use for Linux or Windows.
- A list of files that are used to build the platform.

In the following code block, items in angle brackets (<>) are variables and ellipses (...) indicate elements that can be repeated. The variables are described after the code block:

```
# comment
sgproject "<filename>.sgproj"
{
  TOP_LEVEL_COMPONENT = "<top_level_component>";
  ACTIVE_CONFIG_LINUX = "<active_x86_64_linux_config>";
  ACTIVE_CONFIG_LINUX_ARMV8L = "<active_arm_aarch64_linux_config>";
  ACTIVE_CONFIG_WINDOWS = "<active_windows_config>";

  config "<config_name>"
  {
    <parameter_ID>="<value>";
    ...
  }
  ...
  files
  {
    path = "<file_or_directory>" [,
    platform="<platform>", compiler="<compiler>", action="<action>"];
    ...
  }
}
```

Parameters

filename

The sgproj filename.

top_level_component

The top-level component in the system. Can be overridden using the SimGen option `--top-component`.

active_x86_64_linux_config, active_arm_aarch64_linux_config, and active_windows_config

The name of the active build configuration for each supported host platform, unless overridden using the SimGen option `--configuration`.

config_name

Build configuration name. The format is `<host_OS>-[Debug|Release]-<compiler>`. For example `Linux64-Release-GCC-9.3`.

parameter_ID and value

Project parameter ID and value. These IDs also are exposed in the System Canvas **Project Settings** dialog. For a list of parameter IDs and their default values, see [3.2.6.17.3 Project parameter IDs](#) on page 76. To print all parameter IDs and values for a project, use the SimGen option `--print-config`. Parameter values set in the sgproj file can be overridden using the SimGen option `--override-config`.

file_or_directory

LISA source file, component repository file, or a directory to add to the include or library search path. Repository files have a `.sgrepo` filename extension. Directories have a trailing

/ character. File and directory names can be either absolute or relative to the project file location and paths can contain environment variables.

Any `path` statement can optionally include values for `platform`, `compiler`, and `action`. For example:

```
path = "../lib/Linux64_GCC-9.3/my_file.a", platform="Linux64", compiler="gcc-9.3", action="link|deploy";
```

If any of `platform`, `compiler`, or `action` is omitted, its default value is used.

platform

Optional. Host platform for which this configuration is valid. The default is any platform. Possible values are:

Linux64

Linux x86-64.

Linux64_armv81

Arm® AArch64 Linux.

Win64

64-bit Microsoft Windows, linked against release runtime library.

Win64D

64-bit Microsoft Windows, linked against debug runtime library.

compiler

Optional. On Linux, to select any compiler, omit this option. The default of `gcc` is then used. Possible values are:

vc2019

Microsoft Visual Studio 2019.

gcc

The first GCC version in the search path. To enable SimGen to automatically select the libraries that match the current GCC compiler, use this value.

gcc-9.3

GCC 9.3.

gcc-10.3

GCC 10.3.

gcc-12.3

GCC 12.3.

action

Optional. The default action depends on the file type, see [3.2.6.9 File/Path Properties dialog](#) on page 62 for details. Multiple actions, separated by `|` can be used.

If you apply a directory-specific action to a file, SimGen applies the action to the directory containing the file. In the following example, SimGen treats `MyFile.lisa` as LISA source and adds the parent directory of `MyFile.lisa` to the include and library search paths:

```
path = "MyFile.lisa", actions="lisa|incpath|libpath";
```

Possible values are:

lisa

Process the file as a LISA file. This action is not applicable to directories.

compile

Process the file as a C++ file. If acting on a directory, the compiler compiles all `*.c`, `*.cpp`, and `*.cxx` files in the directory.

ignore

Exclude the file or directory from the build and deploy process.

link

Link the file with existing files. If acting on a directory on Microsoft Windows, System Generator adds all `*.lib` and `*.obj` files in the directory to the linker input. On Linux, it adds all `*.a` and `*.o` files.

deploy

Produce a deployable file. The file is copied to the output directory of the active build configuration. If acting on a directory, SimGen copies the entire directory and its subdirectories to the destination. This action is the only action that acts recursively on subdirectories.

incpath

Add the directory to the list of additional include directories for the compiler. This value can also be set using the `INCLUDE_DIRS` project parameter. This is the default action for directories.

libpath

Add the directory to the list of directories that the linker searches for library files.

Example

This example project file builds an ISIM with configurations for Windows and Linux x86-64 and Linux Arm® AArch64 hosts:

```
sgproject "exampleSystem.sgproj"
{
    TOP_LEVEL_COMPONENT = "exampleSystem";
    ACTIVE_CONFIG_LINUX = "Linux64-Release-GCC-9.3";
    ACTIVE_CONFIG_LINUX_ARMV8L = "Linux64_armv8l-Release-GCC-9.3";
    ACTIVE_CONFIG_WINDOWS = "Win64-Release-VC2019";

    config "Linux64-Debug-GCC-9.3"
    {
        ADDITIONAL_COMPILER_SETTINGS = "-march=core2 -ggdb3 -Wall -std=c++14 -Wno-deprecated -Wno-unused-function";
        ADDITIONAL_LINKER_SETTINGS = "-Wl,--no-undefined";
        BUILD_DIR = "./Linux64-Debug-GCC-9.3";
        COMPILER = "gcc-9.3";
        CONFIG_DESCRIPTION = "Default x86_64 Linux configuration for GCC 9.3 with debug information";
    }
}
```

```

    CONFIG_NAME = "Linux64-Debug-GCC-9.3";
    ENABLE_DEBUG_SUPPORT = "1";
    PLATFORM = "Linux64";
    SIMGEN_COMMAND_LINE = "--num-comps-file 10";
    TARGET_SYSTEMC_ISIM = "1";
}
config "Linux64-Release-GCC-9.3"
{
    ADDITIONAL_COMPILER_SETTINGS = "-march=core2 -O3 -Wall -std=c++14 -Wno-deprecated -Wno-
unused-function";
    ADDITIONAL_LINKER_SETTINGS = "-Wl,--no-undefined";
    BUILD_DIR = "./Linux64-Release-GCC-9.3";
    COMPILER = "gcc-9.3";
    CONFIG_DESCRIPTION = "Default x86_64 Linux configuration for GCC 9.3, optimized for
speed";
    CONFIG_NAME = "Linux64-Release-GCC-9.3";
    PLATFORM = "Linux64";
    PREPROCESSOR_DEFINES = "NDEBUG";
    SIMGEN_COMMAND_LINE = "--num-comps-file 50";
    TARGET_SYSTEMC_ISIM = "1";
}
config "Linux64_armv8l-Debug-GCC-9.3"
{
    ADDITIONAL_COMPILER_SETTINGS = "-march=armv8-a -ggdb3 -Wall -std=c++14 -Wno-deprecated -
Wno-unused-function";
    ADDITIONAL_LINKER_SETTINGS = "-Wl,--no-undefined";
    BUILD_DIR = "./Linux64_armv8l-Debug-GCC-9.3";
    COMPILER = "gcc-9.3";
    CONFIG_DESCRIPTION = "Default armv8l Linux configuration for GCC 9.3 with debug
information";
    CONFIG_NAME = "Linux64_armv8l-Debug-GCC-9.3";
    ENABLE_DEBUG_SUPPORT = "1";
    PLATFORM = "Linux64_armv8l";
    SIMGEN_COMMAND_LINE = "--num-comps-file 10";
    TARGET_SYSTEMC_ISIM = "1";
}
config "Linux64_armv8l-Release-GCC-9.3"
{
    ADDITIONAL_COMPILER_SETTINGS = "-march=armv8-a -O3 -fomit-frame-pointer -Wall -std=c++14
-Wno-deprecated -Wno-unused-function";
    ADDITIONAL_LINKER_SETTINGS = "-Wl,--no-undefined";
    BUILD_DIR = "./Linux64_armv8l-Release-GCC-9.3";
    COMPILER = "gcc-9.3";
    CONFIG_DESCRIPTION = "Default armv8l Linux configuration for GCC 9.3, optimized for
speed";
    CONFIG_NAME = "Linux64_armv8l-Release-GCC-9.3";
    PLATFORM = "Linux64_armv8l";
    PREPROCESSOR_DEFINES = "NDEBUG";
    SIMGEN_COMMAND_LINE = "--num-comps-file 50";
    TARGET_SYSTEMC_ISIM = "1";
}
config "Win64-Debug-VC2019"
{
    ADDITIONAL_COMPILER_SETTINGS = "/Od /RTCsu /Zi";
    ADDITIONAL_LINKER_SETTINGS = "/DEBUG";
    BUILD_DIR = "./Win64-Debug-VC2019";
    COMPILER = "VC2019";
    CONFIG_DESCRIPTION = "Default x86_64 Windows configuration for Visual Studio 2019 with
debug information";
    CONFIG_NAME = "Win64-Debug-VC2019";
    ENABLE_DEBUG_SUPPORT = "1";
    PLATFORM = "Win64D";
    SIMGEN_COMMAND_LINE = "--num-comps-file 10";
    TARGET_SYSTEMC_ISIM = "1";
}
config "Win64-Release-VC2019"
{
    ADDITIONAL_COMPILER_SETTINGS = "/O2";
    BUILD_DIR = "./Win64-Release-VC2019";
    COMPILER = "VC2019";

```

```
CONFIG_DESCRIPTION = "Default x86_64 Windows configuration for Visual Studio 2019,
optimized for speed";
CONFIG_NAME = "Win64-Release-VC2019";
PLATFORM = "Win64";
PREPROCESSOR_DEFINES = "NDEBUG";
SIMGEN_COMMAND_LINE = "--num-comps-file 50";
TARGET_SYSTEMC_ISIM = "1";
}
files
{
    path = "$(PVLIB_HOME)/etc/sglib.sgrepo";
    path = "../LISA/exampleSystem.lisa";
    path = "../LISA/exampleComponent.lisa";
}
}
```

Related information

[1.2 Project files](#) on page 7

[1.3 Repository files](#) on page 9

[1.4 File processing order](#) on page 10

[2.1 SimGen command-line options](#) on page 14

2.3 SimGen detection of duplicate connections

SimGen outputs warnings if it detects duplicate connections between components that are present at compile time.

To prevent these warnings, either:

- Set the environment variable `FASTMODELS_SIMGEN_DISABLE_CONNECTION_DUPLICATION_CHECK` to 1. This disables the duplicate connection checking.
- Use the SimGen command line option `--disable-warning 8225`. This disables the warnings.



Note

SimGen cannot warn about duplicate connections that are created at runtime.

In Fast Models 11.31, these warnings will change to errors.

3. System Canvas

System Canvas is a GUI design tool for visualizing, configuring, and connecting the components of a platform model.

To launch System Canvas from the command line, type `sgcanvas`. It displays the platform as either LISA+ source code, or graphically, in a block diagram editor. System Canvas also allows you to build and launch platform models.

3.1 System Canvas tutorial

This tutorial describes how to perform some basic operations in System Canvas to build a standalone system model that can run an application image.

It demonstrates how to:

- Create a System Canvas project.
- Add, connect, and modify components in the project. You can use the Block Diagram view in System Canvas to do this. You do not need to edit LISA source code directly.
- Build the project.

3.1.1 Starting System Canvas

Start System Canvas from a Linux terminal or from the Microsoft Windows Start menu.

About this task

To start System Canvas:

- On Linux, enter `sgcanvas` in a terminal window. The command has these options:

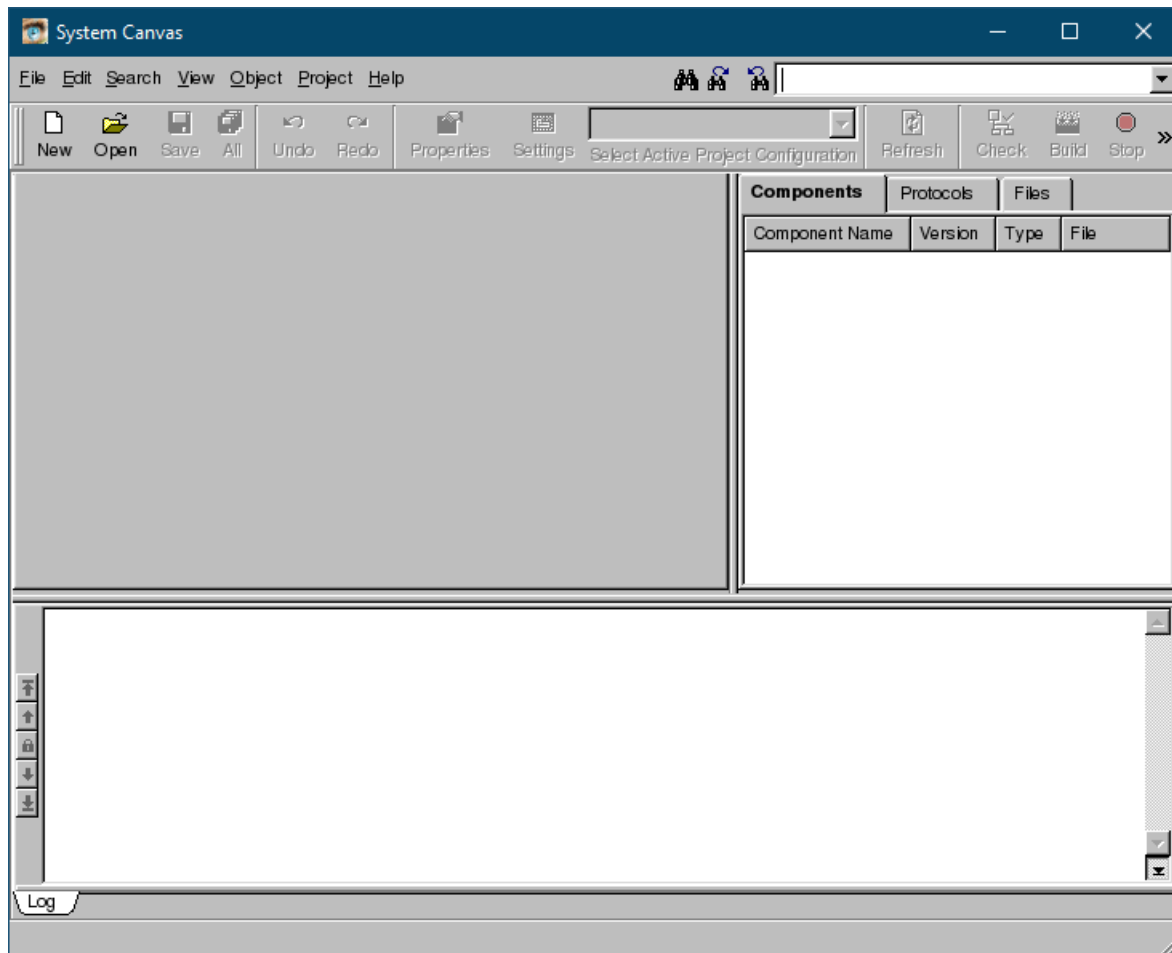
Table 3-1: System Canvas command-line options

Short form	Long form	Description
-h	--help	Print help text and exit.
-v	--version	Print version and exit.

- On Microsoft Windows, launch the **System Canvas** application from the **Start** menu.

The application contains the following subwindows:

- A blank diagram window on the left-hand side of the application window.
- A component window at the right-hand side.
- An output window across the bottom.

Figure 3-1: System Canvas at startup

Related information

[Preferences - Applications group](#) on page 70

[System Canvas GUI](#) on page 38

3.1.2 Creating a new project

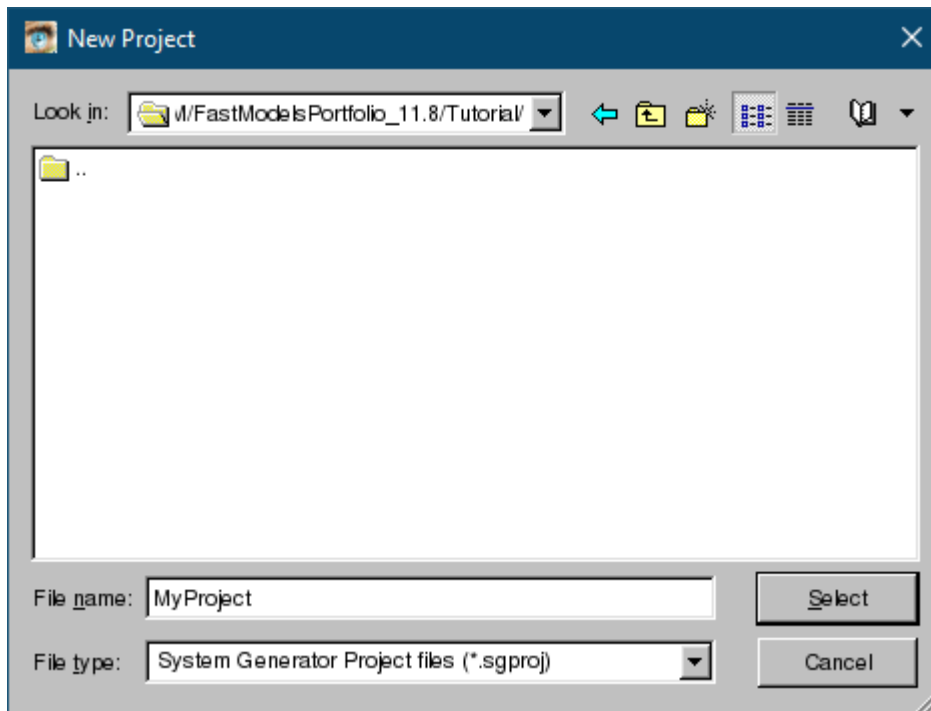
This section describes how to create a new project. The project will be used to create a new system model.

Before you begin

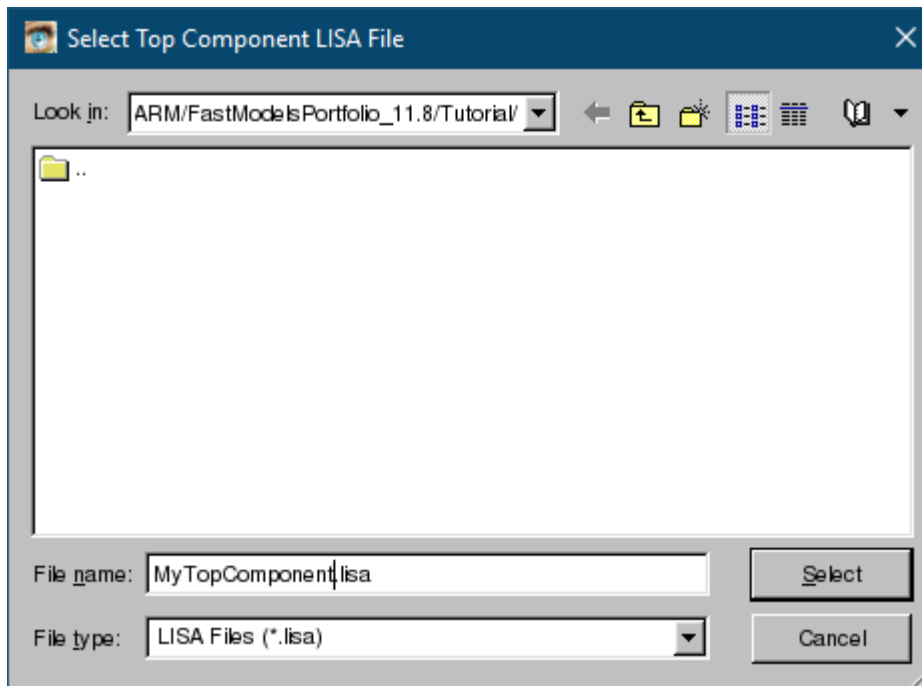
Make sure you have write permission for the directory in which you will create the project.

Procedure

1. Select **New Project** from the **File** menu. Alternatively, click the **New** button on the toolbar. The **New Project** dialog appears.

Figure 3-2: New Project dialog

2. Navigate to the directory to use for your project. Enter `MyProject` in the filename box and click the **Select** button.
A dialog appears for you to enter the name and location of the LISA+ file that represents your new system.

Figure 3-3: Select Top Component LISA File dialog

3. Enter `MyTopComponent.lisa` in the filename box and click the **Select** button.
The component name for the top component is, by default, set to the name of the LISA+ file.

The Workspace area contains a blank block diagram with scroll bars. The Component window, to the right of the Workspace area, lists the components in the default repositories.

Results

These steps create a project file, `MyProject.sgproj` and a LISA+ source file, `MyTopComponent.lisa`. The project file contains:

- System components.
- Connections between system components.
- References to the component repositories.
- Settings for model generation and compilation.

Do not edit the project file. System Canvas modifies it if you change project settings.

The block diagram view of your system is a graphical representation of the LISA+ source. To display the contents of `MyTopComponent.lisa`, click the **Source** tab. This file is automatically updated if you add or rename components in the block diagram.

You can view the LISA+ source for many of the supplied components. To do so, double-click on a component in the Block Diagram. Alternatively, right click on a component in the Components window and select **Open Component**.

3.1.3 Adding the Arm® processor

This section describes how to add an Arm® processor component to the system model.

Procedure

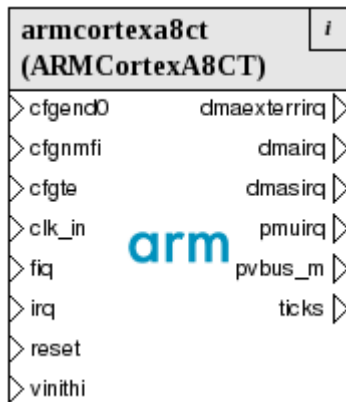
1. Click the **Block Diagram** tab in the Workspace window, unless the block diagram window is already visible.
A blank window with grid points appears.
2. Select the **Components** tab in the Components window to display the Fast Models Repository components.
3. Move the mouse pointer over the `ARMCortexA8CT` processor component in the Component window and press and hold the left mouse button.
4. Drag the component to the middle of the Workspace window.



If you move the component within the Workspace window, the component automatically snaps to the grid points.

5. Release the left mouse button when the component is in the required location.
The system receives the component.

Figure 3-4: ARMCortexA8CT processor component in the Block Diagram window



6. Save the file by selecting **File > Save File** or using **Ctrl+S**.
The asterisk (*) at the end of the system name, in the title bar, shows unsaved changes.

Results

These steps create a System Canvas file, `MyTopComponent.sgcanvas`, in the same location as the project and LISA+ files. It contains the block diagram layout information for your system. Do not edit this file.

3.1.4 Naming components

This section describes how to change the name of a component, for example the processor.

About this task



Component names cannot have spaces in them, and must be valid C identifiers.

Procedure

1. Select the component and click the **Properties** button on the toolbar to display the **Component Instance Properties** dialog.
You can also display the dialog by either:
 - Right-clicking on the component and select **Object Properties** from the context menu.
 - Selecting the component and then selecting **Object Properties** from the **Object** menu.
2. Click the **General** tab on the **Component Instance Properties** dialog.
3. Enter `Arm` in the **Instance name** field.
4. Click **OK** to accept the change. The instance name of the component, that is the name displayed in the processor component title, is now `Arm`.

3.1.5 Resizing components

This section describes how to resize components.

Procedure

1. Select the processor component and move the mouse pointer over one of the green resize control boxes on the edges of the component.
2. Hold the left mouse button down and drag the pointer to resize the component.
3. Release the mouse button to end the resize operation.
To vertically resize the component title bar to avoid truncating text, click the component and drag the lower handle of the shaded title bar.

3.1.6 Hiding ports

This section describes how to hide ports, for instance because they are not connected to anything.

About this task

If there are only a few ports to hide, use the port context menu. Right click on the port and select **Hide Port**. To hide multiple ports:

Procedure

1. Select the component and then select **Object Properties** from the **Object** menu.

2. Click the **Ports** tab on the dialog.
3. Click **Select All** to select all of the ports.
4. Click **Hide selected ports**.
5. Select the boxes next to `clk_in` and `pbus_m`.
6. Click **OK** to accept the change, so that all ports except `clk_in` and `pbus_m` are hidden in the Block Diagram view.

Results

Figure 3-5: Processor component after changes



Related information

[Using port arrays](#) on page 30

3.1.7 Moving ports

This section describes how to move ports, for example to improve readability.

Procedure

1. Place the mouse pointer over the port. The mouse pointer changes shape to a hand with a pointing finger. This is the move-port mouse pointer.
2. Press and hold the left mouse button down over the port, and drag the port to the new location.
This can be anywhere along the inner border of the component that is not on top of an existing port. If you select an invalid position, the port returns to its original location.
3. When the port is in position, release the mouse button.
Arrange any other ports as needed. The `clk_in` port must be on the left side.

3.1.8 Adding components

This section describes how to add components to a project.

Procedure

1. Drag and drop the following components onto the Block Diagram window:
 - `ClockDivider`.
 - `MasterClock`.
 - `PL340_DMC`.
 - `PVBusDecoder`.
 - `RAMDevice`.

The `PL340_DMC` component is included to demonstrate some features of System Canvas and is not part of the final example system.

2. Select the new components individually and use the **General** tab of the **Component Instance Properties** dialog to rename them to:
 - Divider.
 - Clock.
 - PL340.
 - BusDecoder.
 - Memory.

3.1.9 Using port arrays

This section describes how to expand, collapse, and hide port arrays.

Procedure

1. Right click on one of the `axi_if_in` ports in the `PL340` component to open a context menu. Select **Collapse Port** to reduce the port array to a single visible item in the component.
2. Select the `PL340` component and then select **Object Properties** from the **Object** menu.
3. Select the **Ports** tab in the **Component Instance Properties** dialog. The `axi_if_in` port is a port array as indicated by the + beside the port name. Click the + to expand the port tree view.
4. Deselect the checkboxes beside `axi_if_in[2]` and `axi_if_in[3]` to hide the chosen array ports so that expanding the port array still does not display them. Click **OK** to close the dialog. You can also hide a port by using the port context menu and selecting **Hide Port**.
5. To expand the `axi_if_in` port in the `PL340` component, you can:
 - Right click on the port and select **Expand Port** from the port context menu.
 - a. Display the **Component Instance Properties** dialog.
 - b. Select the **Ports** tab.
 - c. Click the + next to the port array to expand the port tree view.
 - d. Select the **Show as Expanded** radio button.

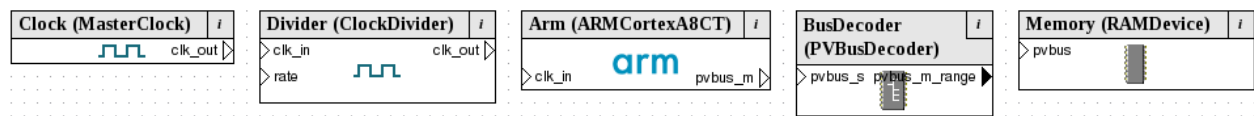
Only the `axi_if_in[0]` and `axi_if_in[1]` ports are shown.

6. To redisplay the `axi_if_in[2]` and `axi_if_in[3]` ports, you can:
 - Use the port context menu and select **Show All Ports**.
 - Reverse the deselection step, selecting the checkboxes next to the hidden ports, in the **Component Instance Properties** dialog.

Ports with more than eight items are shown collapsed by default.

Next steps

The rest of this tutorial does not require the `PL340` component, so you can delete it.

Figure 3-6: Example system with added components

3.1.10 Connecting components

This section describes how to connect components.

Procedure

1. Select connection mode, by doing either of the following:

- Click the **Connect** button.
- Select **Connect Ports Mode** from the **Edit** menu.

2. Move the mouse pointer around in the Block Diagram window:

Option	Description
Not over an object	The pointer changes to the invalid pointer, a circle with a diagonal line through it.
Over an object	The pointer changes to the start connection pointer and the closest valid port is highlighted.

3. Move the cursor so that it is over the `clock` component and close to the `clk_out` port.
4. Highlight the `clk_out` port, then press and hold the left mouse button down.
5. Move the cursor over the `clk_in` port of the `Divider` component.
6. Release the mouse button to connect the two ports.
The application remains in connect mode after the connection is made.
7. Make the remaining connections.

Figure 3-7: Connected components

Connections between the addressable bus ports have bold lines.

3.1.11 View project properties and settings

Before building the model, verify the toolchain configuration and top component using the **Project Settings** dialog.

3.1.11.1 Viewing the project settings

Use the **Project Settings** dialog to view and edit the project configuration. Although no changes are required for this tutorial, this section demonstrates the steps to use if changes were necessary.

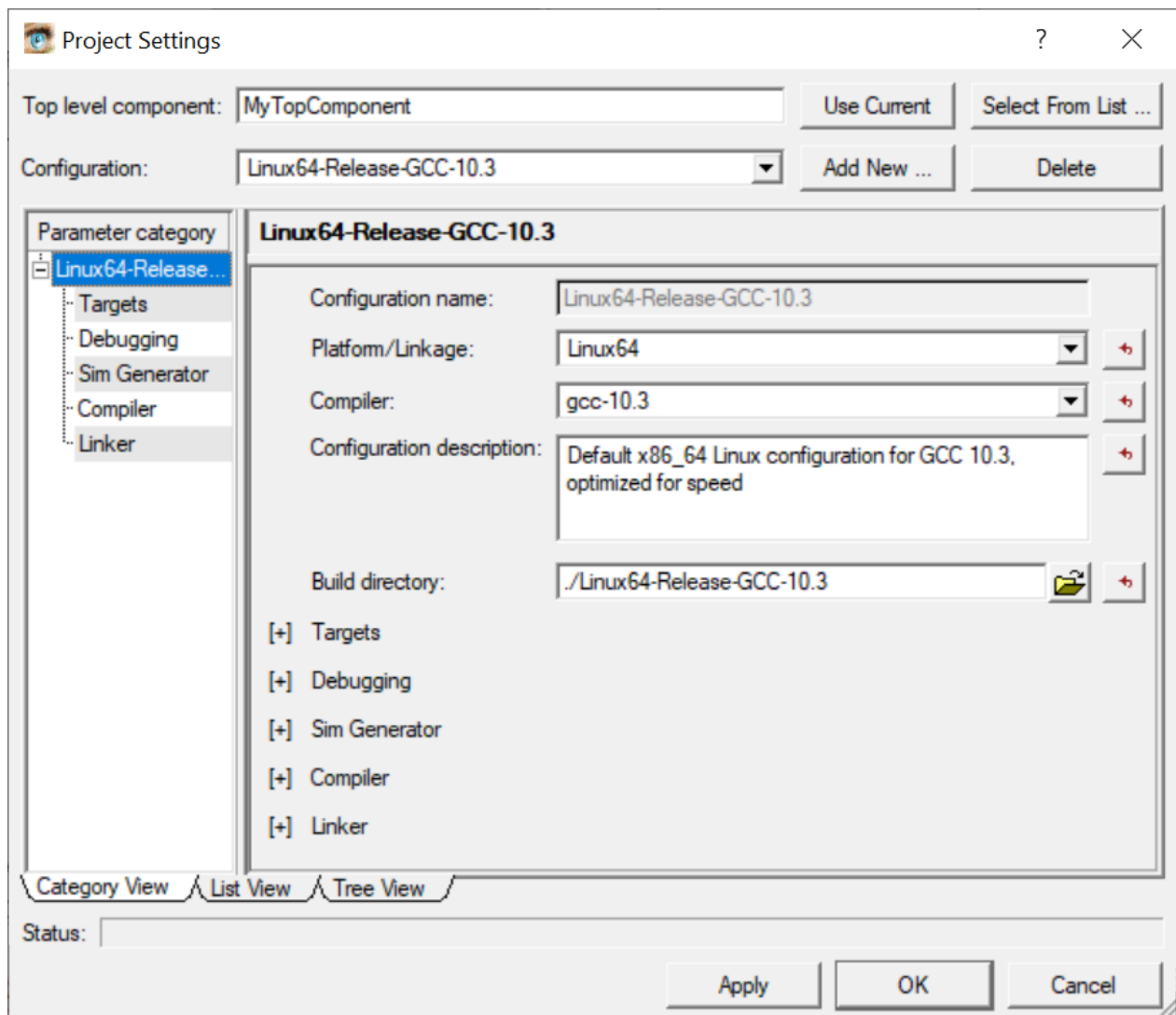
Procedure

Open the **Project Settings** dialog to inspect the project settings for the system, by doing either of the following:

- Click the **Settings** button.
- Select **Project Settings** from the **Project** menu.

The **Project Settings** dialog appears:

Figure 3-8: Project settings for the example



The **Category View**, **List View**, and **Tree View** tabs present different views of the project parameters.

3.1.11.2 Specifying the Active Project Configuration

Use the **Select Active Project Configuration** drop-down menu on the main toolbar to display the configuration options that control how the target model is generated.

About this task

You can choose to build:

- Models with debug support.
- Release models that are optimized for speed.

Display and edit the full list of project settings by selecting **Project Settings** from the **Project** menu. Inspect and modify a configuration for your operating system by selecting it from the **Configuration** drop-down list and clicking the different list elements to view the settings.



Note

- The configuration options available, including compilers and platforms, depend on the operating system.
 - Projects that were created with earlier versions of System Generator might not have the compiler version specified in the **Project Settings** dialog, but are updateable.
-

3.1.11.3 Selecting the top component

The top component defines the root component of the system. Any component can be set as the top component. This flexibility enables building models from subsystems.

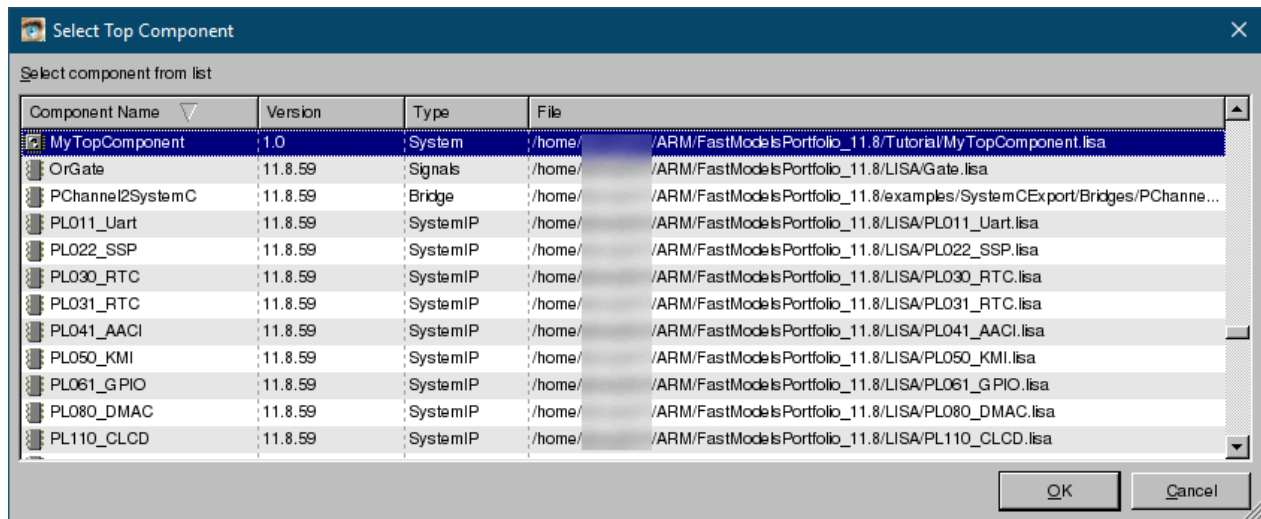
About this task

In the **Project Settings** dialog, click the **Select From List...** button. The **Select Top Component** dialog opens and lists all the components in the system.



Note

If the value in the **Type** column is `system`, the component has subcomponents.

Figure 3-9: Select Top Component dialog showing available components

3.1.12 Changing the address mapping

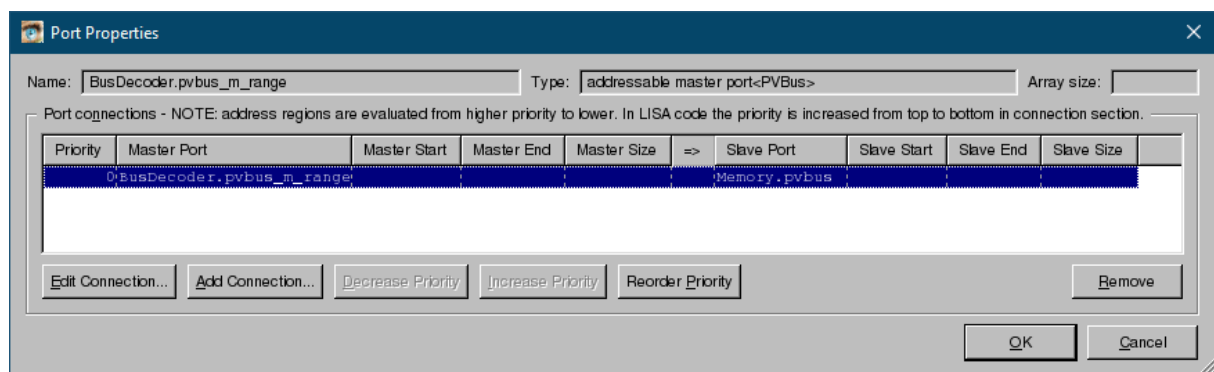
Addressable bus mappings, connections that have bold lines, have editable address maps.

About this task

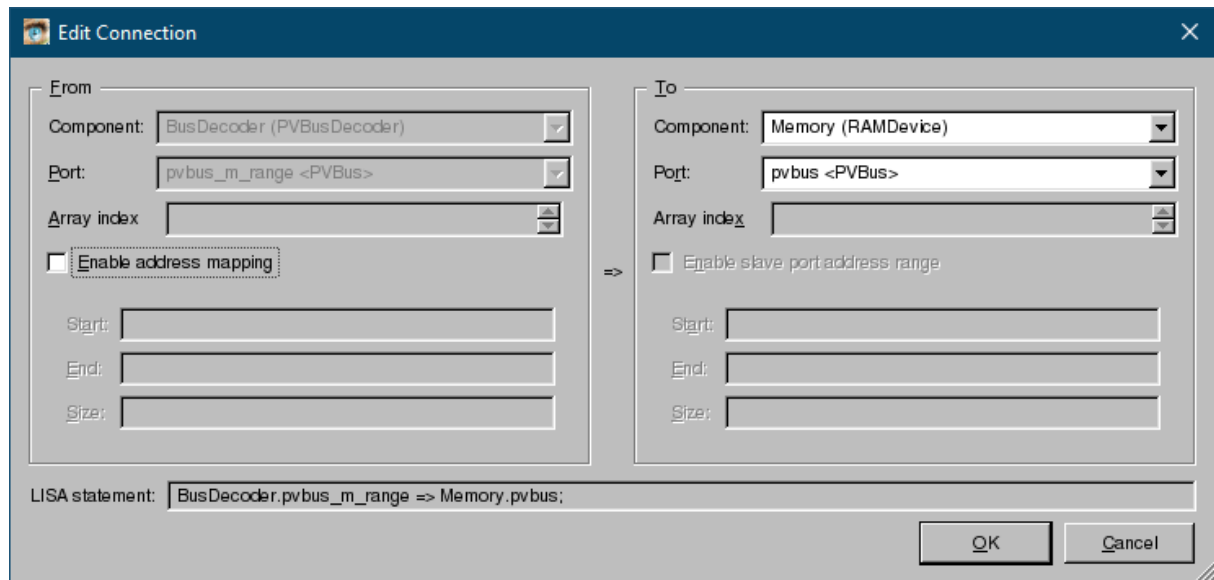
Follow this procedure to change the address mapping.

Procedure

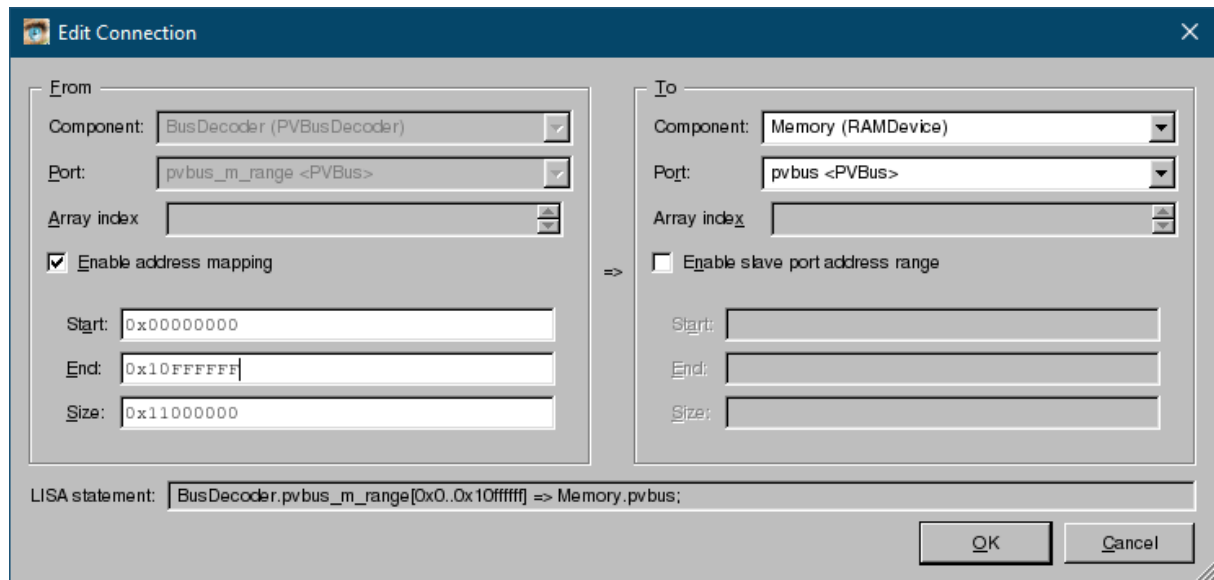
1. Double-click the `pvbus_m_range` port of the BusDecoder component to open the **Port Properties** dialog.

Figure 3-10: Viewing the address mapping from the Port Properties dialog

2. Open the **Edit Connection** dialog by doing either of the following:
 - Select the `Memory.pvbus` Slave Port line, and click **Edit Connection...**
 - Double click on the entry.

Figure 3-11: Edit Connection dialog

3. Select the **Enable address mapping** checkbox to activate the address text fields. The address mapping for the master port is shown on the left side of the **Edit Connection** dialog. **Start**, **End**, and **Size** are all editable. If one value changes, the other values are automatically updated if necessary. The equivalent LISA statement is displayed at the bottom of the **Edit Connection** dialog.
4. Enter a **Start** address of 0x00000000 and an **End** address of 0x10FFFFFFF in the active left-hand side of the **Edit Connection** dialog. The **Size** of 0x11000000 is automatically calculated. This step maps the master port to the selected address range. To map the master port to a different address range on the slave port, select **Enable slave port address range**. This makes the parameters for the slave port editable. The default values are the same as for the master port when the slave address range is enabled. Disabling the slave address range is equivalent to specifying the address range 0...size-1, and not the master address range. In this case, a slave port address range is not required, so deselect the **Enable slave port address range** checkbox.

Figure 3-12: Edit address map for master port

- Click **OK** to close the **Edit Address Mapping** dialog for the `Memory.pvbus` slave port.
- Click **OK** to close the **Port Properties** dialog.

3.1.13 Building the system

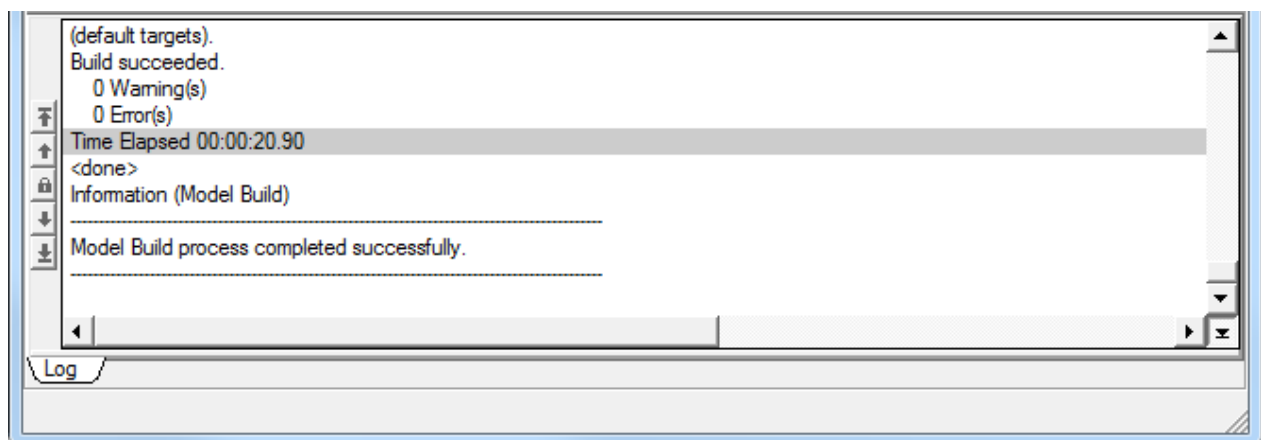
This section describes how to build the model as an `.so` or `.dll` library.

Procedure

Click the **Build** icon on the System Canvas toolbar to build the model.

System Canvas might perform a system check, depending on your preference setting. If warnings or errors occur, a window might open. Click **Proceed** to start the build.

The progress of the build is displayed in the log window.

Figure 3-13: Build process output

Depending on the speed of your computer and the type of build selected, this process might take several minutes.

You can reduce compilation time by setting the `simGen` options `--num-comps-file` and `--num-build-cpus` in the **Project Settings** dialog.

Related information

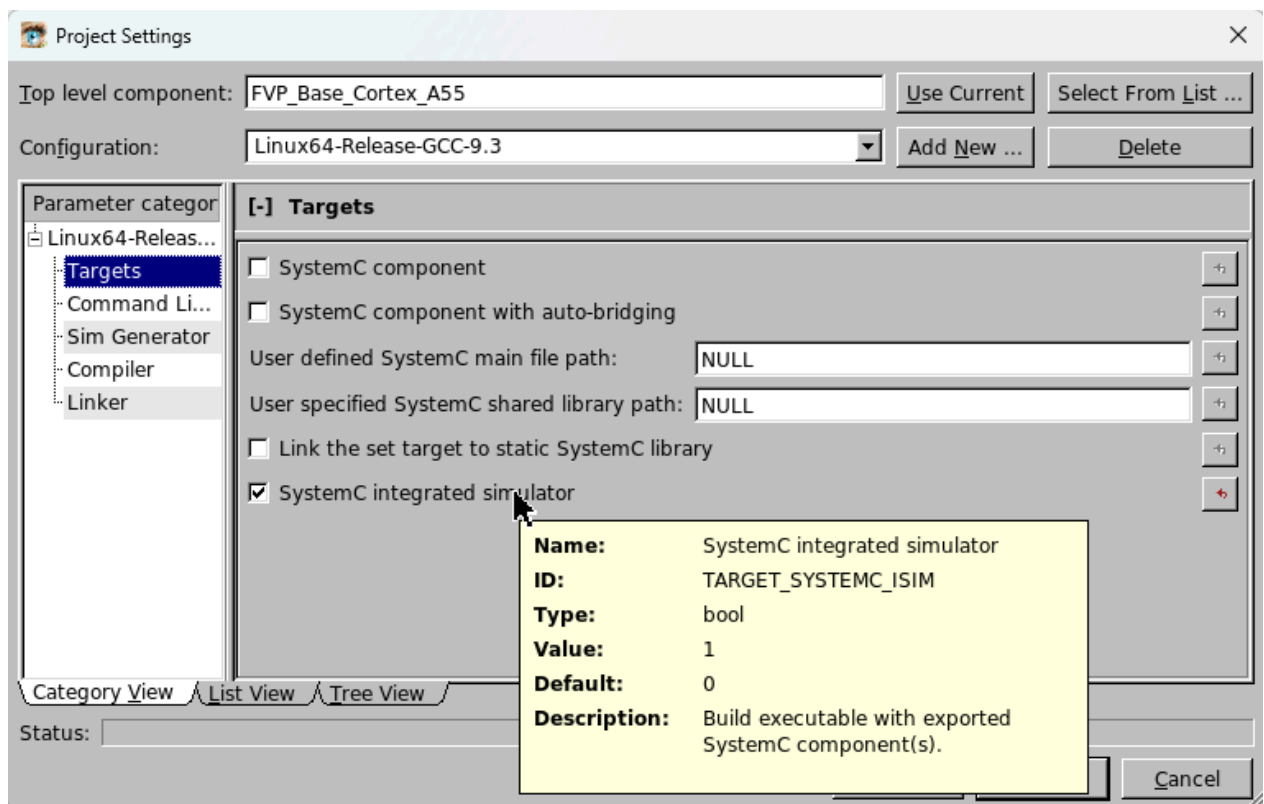
[Building a SystemC ISIM target](#) on page 37

3.1.14 Building a SystemC ISIM target

To build the platform as a standalone executable SystemC Integrated SIMulator (ISIM), tick the **SystemC integrated simulator** checkbox in the **Targets** category in the **Project Settings** dialog.

About this task

Figure 3-14: Building a SystemC integrated simulator target



This option is selected by default for new projects.

System Canvas generates a SystemC ISIM target by statically linking the model with the SystemC framework.

The output executable is called `isim_system`, and is generated in the build directory.

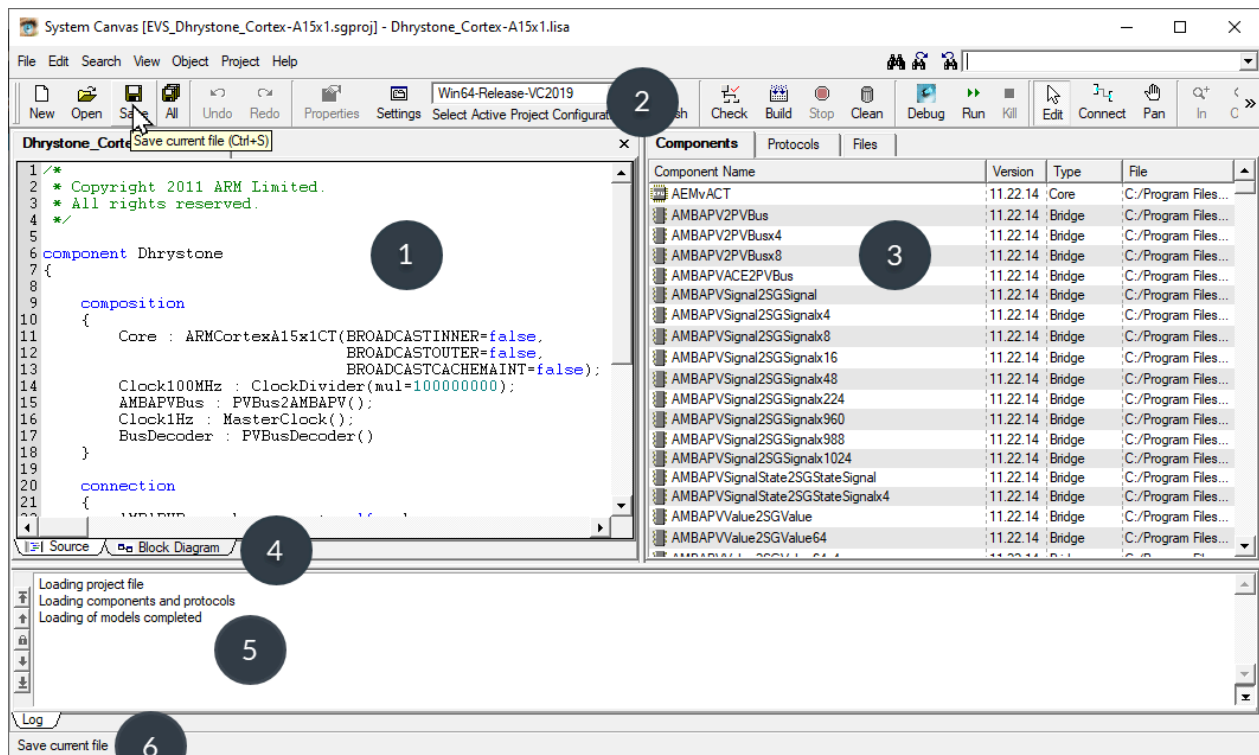
Related information

[Building Fast Models](#)

3.2 System Canvas GUI

This section describes System Canvas, the GUI to the Fast Models tools, which shows the components in a system, component ports, external ports (if the system itself is a component), and connections between ports.

Figure 3-15: Layout of System Canvas



1. **Workspace** displays either the model source code or a block diagram view of the model. You can edit the model through these views.
2. The main toolbar contains buttons for frequently-used options.
3. **Component list** lists all of the available components, protocols, and files in the current project.
4. **Workspace** tabs let you switch between the source code view and block diagram view.
5. **Output window** displays the output from the build or script command.
6. **Status bar** displays information about menu items, commands, and buttons.

3.2.1 Menu bar

The main bar provides access to System Canvas functions and commands.

3.2.1.1 File menu

The **File** menu lists file and project operations.

New Project

Create a new model project.

Load Project

Open an existing project.

Close Project

Close a project. If there are pending changes, the **Save changes** dialog appears.

Save Project

Save the changes made to a project.

Save Project As

Save a project to a new location and name.

New File

Create a new file. The **New File** dialog appears. Select the type from the **File type** drop-down list.

Open File

This displays the **Open File** dialog. Filter the types to display by selecting the type from the **File type** drop-down list. Non-LISA files open as text in the source editor.

Close File

Close a LISA file. A dialog prompts to save any changes.

Save File

Save the changes made to the current LISA file.

Save File As

Save a LISA file to a new location and name.

Save All

Save the changes made to the project and the LISA files.

Print

Print the contents of the **Block Diagram** window.

Preferences

Modify the user preferences.

Recently Opened Files

Display the 16 most recently opened LISA files. Click on a list entry to open the file.

To remove a file from the list, move the mouse cursor over the filename and press the **Delete** key or right click and select **Remove from list** from the context menu.

Recently Opened Projects

Display the 16 most recently opened projects. Click on a list entry to open the project.

To remove a project from the list, move the mouse cursor over the project name and press the **Delete** key or right click and select **Remove from list** from the context menu.

Exit

Close System Canvas. A dialog prompts to save any changes. Disable it by selecting **Do not show this message again**. Re-enable it in the preferences.

Related information

[New project dialogs](#) on page 66

[Preferences - Suppressed messages group](#) on page 73

3.2.1.2 Edit menu

The **Edit** menu lists content operations.

Undo

Undo up to 42 of the latest changes to a file in the **Source** view or to the layout in the **Block Diagram** view. These actions are undoable:

- Add an object such as a component, label, or connection.
- Paste or duplicate.
- Cut or delete.
- Edit object properties.
- Move.
- Resize.

Undo and **Redo** operations can affect **Block Diagram** view zoom and scroll actions.

Undo and Redo typically work normally. For example:



1. Change the system in the **Block Diagram** view by adding a RAMDevice component with name `RAM`.
2. Switch to **Source** view. The text `RAM : RAMDevice();` is present in the composition section.
3. Change the code by removing the line `RAM : RAMDevice();`.
4. Change the code by adding, for example, the line `PVS : PVBusSlave();`.
5. Click on the **Block Diagram** tab. The change to the source code is reflected by the `RAM` component being replaced by the `PVS` component.

6. Select **Undo** from the **Edit** menu. The **Block Diagram** view shows that **RAM** is present but **envs** is not.
 7. Select **Redo** from the **Edit** menu. The **Block Diagram** view shows that **envs** is present but **RAM** is not.
-

Redo

Redo the last undone change. This cancels the result of selecting **Undo**. Selecting **Redo** multiple times cancels multiple **Undo** actions.

Cut

Cut the marked element into the copy buffer.

Copy

Copy the marked element into the copy buffer.

Paste

Paste the content of the copy buffer at the current cursor position.

Duplicate

Duplicate the marked content.

Delete

Delete the marked element.

Select All

Select all elements.

Edit Mode

Change the **Workspace** to **Edit** mode. The cursor can select components.

Connect Ports Mode

Select **Connection** mode. The cursor can connect components.

Pan Mode

Select **Movement** mode. The cursor can move the entire system in the **Workspace** window.

3.2.1.3 Search menu

The **Search** menu lists find, replace and go to functions.

Find

Search for a string in the active window (with a thick black frame).

Find Next

Repeat the last search.

Find Previous

Repeat the last search, backwards in the document.

Replace

In the **Source** view, search for and replace strings in a text document.

Go To Line

In the **Source** view, specify a line number in the currently open LISA file to go to.



Use the search icons at the top right of the application window to search for text. Entering text in the search box starts an incremental search in the active window.

Related information

[Find and Replace dialogs](#) on page 64

3.2.1.4 View menu

The **View** menu lists the **Workspace** window display options.

Show Grid

Using the grid simplifies component alignment.

Zoom In

Show more detail.

Zoom Out

Show more of the system.

Zoom 100%

Change the magnification to the default.

Zoom Fit

Fit the entire system into the canvas area.

Zoom Fit Selection

Fit the selected portion into the canvas area.

3.2.1.5 Object menu

The **Object** menu lists system and system component operations.

Open Component

Open the source for the selected component.

Add Component

Display all of the components available for adding to the block diagram.

Add Label

The mouse cursor becomes a default label. To add the label, move it to the required location in the **Block Diagram** window and click the left mouse button. The **Label Properties** dialog appears.

Add Port

Display the **External Port** dialog. Specify the type of port to add.

Mirror Self Port

Switch the direction that the external port image points in. It does not reverse the signal direction, so a master port remains a master port. If an unconnected port is not selected, this option is disabled.

Expand Port

For a port array, display all of the individual port elements. Expanded is the default for port arrays with eight or fewer ports. Collapsed is the default for port arrays with more than eight elements.



Ports with many elements might expand so that elements appear on top of one another. Either: click and drag them apart, or collapse the port, increase the component size, then expand the port again.

Collapse Port

For a port array, hide the individual port elements and only display the top-level port name.

Hide Port

Disable the selected port and make it invisible.

Hide All Unconnected Ports

Hide all ports that are not connected to a component.

Show/Hide Ports of Protocol Types...

Hide all ports that use a specified protocol. The **Show/Hide Connection Types** dialog appears. Select the protocols to filter.

Show All Ports

Show all ports. Some might overlap if there is not enough space.

Autoroute Connection

Redraw the selected connection.

Autoroute All Connections

Redraw all of the connections.

Documentation

Open the documentation for the selected component.

Object Properties

Display the **Component Instance Properties** dialog to view and edit the properties for the selected component.

3.2.1.6 Project menu

The **Project** menu lists build, check, configure, run, and set options.

Check System

Check for errors or missing information. This feature does not check everything, but does give useful feedback.

Generate System

Generate the C++ source code, but do not compile it. After generation, click **Build System** and **Debug** to run the model.

Build System

Generate and compile the generated C++ source code, producing a library or a runnable model.

Stop Build

Cancel the active build process.

Clean

Delete all generated files.

Run

Run...

Open the **Run** dialog to specify the run command.

Run ISIM system

Execute the ISIM executable with command-line options taken from project settings and user preferences.

Clear History

Clear all recent run command entries.

Recent Command Entries (up to 10)

Call recent command entries.

Kill Running Command

Stop the running synchronous command.

Launch Host Debugger

Microsoft Windows

Launch Microsoft Visual Studio. Build the system there, and start a debug session.



You can take the command-line arguments for ISIM systems from Microsoft Visual Studio by selecting **Project > Properties > Configuration Properties > Debugging**.

Linux

Launch the executable or script set in the application preferences. The target must be an ISIM executable. Arm recommends this method for debugging at source-level.

Add Files

Add files to the system.

Add Current File

Add the currently open file to the system.

Refresh Component List

Update the **Component List** window to show all available components.

Setup Default Repository

Display the **Default Model Repository** section of the **Preferences** window, and select the default repositories for the next new project.



Note

This option does not affect the currently open project.

Set Top Level Component

Displays the **Select Top Component** dialog that lists all available components in the system.

The top component defines the root component of the system. It can be any component. This option enables building models from subsystems.



Note

If the value in the **Type** column is `system`, the component has subcomponents.

Active Configuration

Select the system build configuration from the project file list.

Project Settings

Display the **Project Settings** dialog.

Related information

[Preferences - Applications group](#) on page 70

3.2.1.7 Help menu

The **Help** menu lists documentation, software and system information links.

Fast Models User Guide

Display the Fast Models User Guide.

Fast Models Tools User Guide

Display this document.

LISA+ Language Reference Guide

Display the LISA+ Language for Fast Models Reference Guide.

AMBA-PV User Guide

Display the AMBA-PV Extensions to TLM 2.0 User Guide.

Release Notes

Display the Fast Models release notes.

Documents in \$PVLIB_HOME/Docs

List the PDF files in the directory `$PVLIB_HOME/Docs`.

End User License Agreement (EULA)

Display the license agreement.

About

Display the version and license information.

System Information

Display information about the tools and loaded models.

3.2.2 Toolbar

The toolbar sets out frequently used menu functions.

New

Create a new project or LISA file.

Open

Open an existing project or file.

Save

Save current changes to the file.

All

Save project and all open files.

Undo

Undo the last change in the **Source** or **Block Diagram** view.

Redo

Undo the last undo.

Properties

Display the **Properties** dialog for the selected object:

Nothing

The **Component Model Properties** dialog, with the properties for the top-level component.

Component

The **Component Instance Properties** dialog.

Connection

The **Connection Properties** dialog.

Port

The **Port Properties** dialog.

Self port

The **Self Port Properties** dialog.

Label

The **Label Properties** dialog.



The **Properties** button only displays properties for items in the block diagram.

Settings

Display the project settings.

Select Active Project Configuration

Select the build target for the project.

Refresh

Refresh the component and protocol lists.

Check

Perform a basic model error and consistency check.

Build

Generate a virtual system model using the project settings.

Stop

Stop the current generation process.

Clean

Delete all generated files.

Run

Execute the most recent run command. The down arrow next to the button opens the **Run** dialog.

Kill

End the simulation.

Edit

Edit mode: the cursor selects and moves components.

Connect

Connection mode: the cursor connects components.

Pan

Movement mode: the cursor moves the entire system in the **Workspace** window.

Zoom

Use the **In**, **Out**, **100%**, and **Fit** buttons to change the system view zoom factor in the **Workspace** window.

Related information

[Viewing the project settings](#) on page 31

[Edit menu](#) on page 40

[Component Instance Properties dialog](#) on page 56

[Component Model Properties dialog for the system](#) on page 58

[Connection Properties dialog](#) on page 61

[Label Properties dialog](#) on page 65

[New File dialog \(File menu\)](#) on page 66

[Open File dialog](#) on page 67

[Port Properties dialog](#) on page 68

[Self Port dialog](#) on page 80

3.2.3 Workspace window

This section describes the **Workspace** window, which displays editable representations of the system.

Related information

[Open File dialog](#) on page 67

[Preferences dialog](#) on page 69

3.2.3.1 Source view

The **Source** view displays the LISA source code of components. It can also display other text files.

The source text editor features:

- Similar operation to common Microsoft Windows text editors.
- Standard copy and paste operations on selected text, including with an external text editor.

- Undo/redo operations. Text changes can be undone by using **Ctrl-Z** or **Edit > Undo**. Repeat text changes with **Ctrl-Y** or **Edit > Redo**.
- Syntax highlighting for LISA, C++, HTML, Makefiles, project (*.sgproj) and repository (*.sgrepo) files.
- Auto-indenting and brace matching. Indenting uses four spaces not single tab characters.
- Auto-completion for LISA source. If you type a delimiter such as "." or ":", a list box with appropriate components, ports, or behaviors appears. Icons indicate master and slave ports.
- Call hint functionality. If you type a delimiter such as "(", a tooltip appears with either a component constructor or behavior prototype, depending on the context. Enable call hints by enabling tooltips in the **Appearance** pane of the **Preferences** dialog.



Every time System Canvas parses a LISA file, it updates lexical information for auto-completion and call hint functionality. This occurs, for example, when switching between the views.

3.2.3.2 Source view context menu

The **Source** view context menu lists text operations.

Undo

Undo the last change.

Redo

Undo the last undo.

Cut

Cut the selected text.

Copy

Copy the selected text.

Paste

Paste text from the global clipboard.

Delete

Delete the selected text.

Select All

Selects all of the text in the window.

3.2.3.3 Block Diagram view

The **Block Diagram** view displays a graphical representation of the system. It provides a rapid way to create and configure components or systems consisting of multiple components.

This view supports copy and paste operations on selected components, connections, labels, and self ports:

- Use the cursor to draw a bounding rectangle around the box.
- Press and hold shift while clicking on the components to copy.

Copied components will have different names. To copy connections, select both ends of the connection.



Changes made in one view immediately affect the other view.

Open files have a named workspace tab at the top of the **Workspace** window. An asterisk after the name indicates unsaved changes. A question mark means that the file is not part of the project.

Click the right mouse button in the workspace to open the context menu for the view.

Displaying the block diagram fails if:

- The file is not a LISA file.
- The syntax of the LISA file is incorrect.
- The LISA file contains more than one component.
- The LISA file contains a protocol.

3.2.3.4 Block Diagram view context menu

The **Block Diagram** view context menu lists object operations.

Open Component

Open a new workspace tab for the selected component.

Delete

Delete the object under the mouse pointer.

Add Port...

Add a port to the component.

Mirror Self Port

Mirror the port image.

Expand Port

For a port array, display all of the individual port elements.

Collapse Port

For a port array, hide the individual port elements.

Hide Port

Disable the selected port and make it invisible.

Hide All Unconnected Ports

Hide all ports that are not connected to a component.

Show/Hide Ports of Protocol Types...

Hide all ports that use a specified protocol.

Show All Ports

Show all ports of the component.

Autoroute connection

Redraw the selected connection.

Documentation

Open the documentation for the selected component.

Object Properties

Open the object properties dialog.

3.2.4 Component window

This section describes the **Component** window, which lists the available components and their protocols and libraries.

3.2.4.1 Component window views

The **Component** window has view tabs.

Components

The components, and their version numbers, types, and file locations. Drag and drop to place in the block diagram. Double click to open in the workspace.

Protocols

The protocols of these components, and their file locations. Double click to open in the workspace.

Files

The project files, in a fully expanded file tree with the project file as the root. Double click to open in the workspace. The project file can contain LISA files and component repositories. A repository can itself contain a repository.

**Note**

The order of file processing is from the top to the bottom. To move objects:

- Select and use **Up** and **Down** in the context menu, or use **Alt + Arrow Up** or **Alt + Arrow Down**.
- Drag and drop.

3.2.4.2 Component window context menu

The **Component** window context menu lists file operations and a documentation link.

Open

Open the associated file.

Add...

Add a repository, component or protocol file, or a library.

Add New...

Add a new file.

Add Directory...

Add an include path to be used by the compiler (**Files** tab only). To simplify navigation, the add dialog also shows the filename.

Remove

Remove an item.

Up

Move a file up the file list (**Files** tab only).

Down

Move a file down the file list (**Files** tab only).

Reload

Reload a component or protocol.

Refresh Component List

Refresh the entire component list.

Documentation

Open the documentation for the component.

Properties

Show the properties of the item.

3.2.5 Output window

The **Output** window displays the build or script command output.

The left side of the window has controls:

First

Go to the first message.

Previous

Go to the previous message.

Stop

Do not scroll automatically.

Next

Go to the next message.

Last

Go to the last message.

The right side of the window has controls:

Scroll bar

Move up and down in the output.

Stick

Force the window to show the latest output, at the bottom.

3.2.6 System Canvas dialogs

This section describes the dialog boxes of System Canvas.

3.2.6.1 Add Existing Files and Add New File dialogs (Component window)

This section describes these dialogs that add components, protocols, libraries, repositories, or source code to a project.

Related information

[Add Files dialog \(Project menu\)](#) on page 55

[New File dialog \(File menu\)](#) on page 66

3.2.6.1.1 Displaying the Add Existing Files and Add New File dialogs (Component window)

This section describes how to display dialogs that add components, protocols, libraries, repositories, or source code to a project.

Procedure

Display a dialog by right-clicking in the **Component** window and selecting from the context menu:

- **Add.**
- **Add New.**

3.2.6.1.2 Using the Add Existing Files and Add New File dialogs (Component window)

This section describes how to add a file using the **Component** window context menu.

Procedure

1. Select the **Components, Protocols, or Files** tab in the **Component** window.
To add a file at the top level of the file list, select the top entry. To add a file to an existing repository in the file list, select the repository.

2. Right-click in the **Component** window and select **Add** or **Add New** from the context menu.
- | Option | Description |
|---------|--|
| Add | In the Add Existing Files dialog, go to the file and select it. |
| Add New | In the Add New File dialog, go to the directory to contain the file and enter the name. |

- Save time with the **Recently selected files** drop-down list. To remove a file, mouse over it and press **Delete**, or right-click and select **Remove from list** from the context menu.
3. Click **Open** to add the file and close the dialog.

Next steps

Library files, those with `.lib` or `.a` extensions, need build actions and a platform.

Related information

[File/Path Properties dialog](#) on page 62

3.2.6.1.3 Using environment variables in filepaths

Environment variables in filepaths enable switching to new repository versions without modifying the project.

About this task

For example, using `$(PVLIB_HOME)/etc/sglib.sgrepo` as the reference to the components of the Fast Models Portfolio enables migration to future versions of the library by modifying environment variable `PVLIB_HOME`.



Note

On Microsoft Windows, Unix syntax is valid for environment variables and paths, for example `$PVLIB_HOME/etc/my.sgrepo`.

Edit a filepath through the **File Properties** dialog:

Procedure

1. Select the file and click select **Properties** from the context menu.

2. Edit the **File** entry to modify the filepath.

Related information

[File/Path Properties dialog](#) on page 62

3.2.6.1.4 Assigning platforms and compilers for libraries

This section describes how to set the operating system that a library is for, and the compiler that built it.

Procedure

Use the **File Properties** dialog to specify the operating system and compilers by checking the appropriate boxes in the **Supported platforms** pane.

Microsoft Visual Studio distinguishes between debug and release versions.

Related information

[File/Path Properties dialog](#) on page 62

[Project Settings dialog](#) on page 73

3.2.6.2 Add Files dialog (Project menu)

Add files to a project with this dialog.

Select **Add File** from the **Project** menu to add a new file to the project.

The behavior of this dialog is identical to that of the **Add Existing Files** dialog.

To create a new file from code in the **Source** view, select **Add Current File** from the **Projects** menu to add the file to the project. No dialog appears.



Save time with the **Recently selected files** drop-down list. To remove a file, mouse over it and press **Delete**, or right-click and select **Remove from list** from the context menu.

Related information

[Add Existing Files and Add New File dialogs \(Component window\)](#) on page 53

[File/Path Properties dialog](#) on page 62

3.2.6.3 Add Connection dialog

This dialog adds a connection to a component port.

To open the dialog:

1. Select a component port.
2. Display the **Port Properties** dialog by selecting **Object Properties** from the context menu or from the **Object** menu.
3. Click the **Add Connection** button.

The enabled fields for the dialog depend on whether a slave or master was displayed in the Port Properties dialog.

**Note**

This dialog also appears if you use the cursor in connect mode to connect two ports in the block diagram and one or more of the ports is a port array.

Related information

[Edit Connection dialog](#) on page 61

3.2.6.4 Component Instance Properties dialog

This dialog displays the properties of a component.

To open the dialog, select a component in the block diagram, and click on the **Properties** button in the toolbar or select **Object Properties** from the **Object** menu.

General

The component name, instance name, filename and path, and repository.

The **Instance name** field is editable.

**Note**

To view the properties of the top-level component, double-click in an area of the workspace that does not contain a component.

Properties

All properties for the component. If the properties are not editable, the tab says **Properties (read only)**.

If the property is a Boolean variable, a checkbox appears next to it.

Parameters

All editable parameters for this component. Enter a new value in the **Value** edit box.

The following controls are present:

Parameter name

The parameters for this component.

Value

Select a parameter and then click the text box in the **Value** column to set the default value for the parameter.

Integer parameters in decimal format can contain binary multiplication suffixes. These left-shift the bits in parameter value by the corresponding power of two.

Table 3-2: Suffixes for parameter values

Suffix	Name	Multiplier
K	Kilo	2^{10}
M	Mega	2^{20}
G	Giga	2^{30}
T	Tera	2^{40}
P	Peta	2^{50}

Ports

All the ports in the component.

For port arrays, display all of the individual ports or only the port array name by selecting **Show as Expanded** or **Collapsed**.

The properties of individual ports are editable:

1. Select a port from the list.
2. Click **Edit** and change the properties of the port.
3. Click **OK** to save the changes.



Note

If you click **OK**, the changes apply immediately.

Enable/disable individual ports with the checkboxes:

- Click **Show selected ports** to display the checked ports.
- Click **Hide selected ports** to hide the checked ports.



Note

Hiding the top level of a port array hides all of the individual ports but they retain their check mark setting.

Methods

All the behaviors (component functions) that the component implements.

Related information

[Component Model Properties dialog for the system](#) on page 58

[Component Properties dialog for a library component](#) on page 60

[Label Properties dialog](#) on page 65

[Port Properties dialog](#) on page 68

[Protocol Properties dialog](#) on page 79

[Self Port dialog](#) on page 80

3.2.6.5 Component Model Properties dialog for the system

This dialog displays the properties for the system.

To open the dialog, select a blank area in the block diagram, right-click and select **Object Properties** from the context menu to display the properties for the system or select **Object Properties** from the **Object** menu.

General

The system name, filename and path, and repository.

The **Component name** field is editable.

Properties

If the property is a Boolean variable, a checkbox appears next to it.

Changes in these dialogs alter the LISA code in the model.

Double-click in the **Value** column to change the property.

Table 3-3: Component properties

Property	ID	Default	Description
Component name	component_name	""	A string containing the name for the component.
Component category	component_type	""	A string describing the type of component. This can be "Processor", "Bus", "Memory", "System", or any free-form category text.
Component description	description	""	A textual component description.
Component documentation	documentation_file	""	A filepath or an HTTP link to documentation. Supported file formats are PDF, TXT, and HTML.
Executes software	executes_software	0	The component executes software and can load application files. 1 for processor-like components, 0 for other components.
Hidden	hidden	0	1 for components hidden from the Component window. Otherwise, hidden components behave exactly as normal components, and they do appear in the Workspace window.

Property	ID	Default	Description
Has CADI interface	has_cadi	1	1 for components with a CADI interface, permitting connection to the target with a CADI-compliant debugger. 0 for components with no CADI interface.
Icon pixmap file	icon_file	""	The XPM file that contains the system icon.
License feature	license_feature	""	The license feature string required to run this system model.
Load file extension	loadfile_extension	""	The application filename extension for this target. Example: ".elf" or ".hex".
Small icon pixmap file	small_icon_file	""	The XPM file that contains the 12x12 pixel system icon.
Component version	version	"1.0"	The version of the component.

Parameters

Parameter name

The parameters for this component.

Value

Select a parameter and then click the text box in the **Value** column to set the default value. Integer parameters in decimal format can contain binary multiplication suffixes. These left-shift the bits in parameter value by the corresponding power of two.

Table 3-4: Suffixes for parameter values

Suffix	Name	Multiplier
K	Kilo	2^{10}
M	Mega	2^{20}
G	Giga	2^{30}
T	Tera	2^{40}
P	Peta	2^{50}

Parameter ID in LISA code

The LISA ID for the component parameters.

Add

Click to add a new parameter.

Edit

Select a parameter and then click to change the name.

Delete

Select a parameter and then click to delete it.

Ports

All external ports.

If a port contains an array of ports, the **Size** column displays the number of ports in the array.

Enable/disable individual ports with the checkboxes:

- Click **Show selected ports** to display the checked ports.
- Click **Hide selected ports** to hide the checked ports.

Methods

The available LISA prototypes. The list is for reference only. It is not editable.

3.2.6.6 Component Properties dialog for a library component

This dialog displays the properties of a library component.

To open the dialog, select a component from the **Components** list, and right-click and select **Properties** from the context menu or select **Object Properties** from the **Object** menu.

General

Component name

The name of the component.

Type

The component category, for example `core` or `peripheral`.

Version

The revision number for the component.

File

The file that defines the component.

Repository

The repository that contains the component.

Description

Information about the component.

Properties (read only)

All the usable properties of the component.



Note

A valid `license_feature` string allows this component to work in a model.

Parameters (read only)

All the parameters for the component.

Ports (read only)

All the ports in the component.



No port arrays are expandable here.

Methods

The LISA prototypes of the methods, that is, behaviors, of the component. The list is for reference only. It is not editable.

3.2.6.7 Connection Properties dialog

This dialog displays port connection properties.

To open the dialog, double click on a connection between components in the workspace.

Name

The name of the port.

Type

The type of port and the protocol.

To change the address mapping, click **Master Port Properties** or **Slave Port Properties**.

Related information

[Port Properties dialog](#) on page 68

3.2.6.8 Edit Connection dialog

This dialog controls port connection properties.

To open the dialog and change the connected port or the address mapping, select a connection from the **Port Properties** dialog and click **Edit Connection....**

Component

For a slave port, the source component is editable. For a master port, the destination component is editable.

Port

For a slave port, the master port is editable. For a master port, the slave port is editable.

Array index

For port arrays, an index value for the element to use.

Enable address mapping

Set the port address range with the **Start** and **End** boxes.

Start

The start address for the port.

End

The end address for the port.

Size

The size of the address region. Given the **Start** and **End** values, System Canvas calculates this value.

OK/Cancel

Click **OK** to modify the connection. Click **Cancel** to close the dialog without changing the connection.

LISA statement

The code equivalent to the address range.

3.2.6.9 File/Path Properties dialog

This dialog displays properties for the file and controls build and compile options.



- On Microsoft Windows, the / and \ directory separators both appear as /. This simplification does not affect operation.
 - Avoid using Japanese or Korean characters in filepaths. They can cause failure to find libraries.
-

Select a component from the **Component** window **Files** tab, right click on it to open the context menu, then click **Properties** to display the dialog.

General**File or path**

The name of the file.



The **File Properties** dialog is modeless. You can select a different file without closing the dialog. A warning message prompts to save any changes.

Absolute path

The full path to the file.

Repository

The repository file that contains this component entry.

Type

A brief description of the component type.

Info

The status of the file. For example, `file does not exist`.

Supported platforms

Select the platforms that the component supports:

- Linux64.
- Win64 (release runtime library).
- Win64D (debug runtime library).

Compiler

Select the compiler for this component from the drop-down list:

- No preference.
- Specific Microsoft Visual C++ compiler.
- gcc version found in \$PATH at compile time.
- Specific gcc version.

Build actions

Default actions depending on file extension

.lisa

A LISA source file that SimGen parses.

.c .cpp .cxx

A C or C++ source file that the compiler compiles.

.a .o

A Linux object file that SimGen links to.

.lib .obj

A Microsoft Windows object file that SimGen links to.

.sgproj

A project file that SimGen parses.

.sgrepo

A component repository file that SimGen parses.

directory_path/

An include directory for the search path that the compiler uses. The trailing slash identifies it as an include path. For example, to add the directory that contains the `*.sgproj` file, specify `./` (dot slash), not only the dot.

All other files

Copy a deploy file to the build directory.

**Note**

Simulation Generator (SimGen) is one of the Fast Models tools.

Ignore

Exclude the selected file from build and deploy. This feature can be useful for examples, notes, or temporarily disabled files.

Customize actions

Ignore the file extension. Specify the actions with the check boxes:

LISA - input file passed to Simulator Generator as LISA

System Canvas passes the file to SimGen as a LISA file. Do not use this option for non-LISA files.

Compile - compile as C/C++ source code

To compile a file as C/C++ code during the build process, add it to this list of files.

Link - input file for linker

Link the file with the object code during the build process.

Deploy - copy to build directory

Copy the file into the build directory. This option can, for example, add dynamic link libraries for running the generated system model.

Include path - add the file's path to additional include directories

Add the path of the parent directory that holds the file to the list of include directories for the compiler.

Library path - add the file's path to additional library directories

Add the path of the parent directory that holds the file to the list of library directories for the compiler.

Related information

[Project parameter IDs](#) on page 76

3.2.6.10 Find and Replace dialogs

This dialog enables searching for and replacement of text in an editor window.

The **Find** dialog and the **Find and Replace** dialog are essentially the same dialog in two modes, find only, and find and replace. Switch modes by clicking the **Find mode** or **Find and replace mode** buttons. By default, matches are case sensitive but matches can appear as part of longer words. Change the default behavior by setting or clearing the relevant checkboxes in the dialog.

Open the **Find** dialog by clicking **Search > Find...** in the main menu. Type the text to find in the box and click the **Find Next** or **Find Previous** buttons to search upwards or downwards from the

current cursor position. You can re-use previous search terms by clicking on the drop-down arrow on the right of the text entry box.

Open the **Find and Replace** dialog by clicking **Search > Replace** in the main menu. Replace the current match with new text by clicking the **Replace** button, or all matches by clicking the **Replace All** button. You can re-use previous find or replacement terms by clicking on the drop-down arrow on the right of the text entry boxes.

Find and Replace mode is only available if the current active window is a source editor. In that mode, additional replace controls appear. The dialog is modeless, so you can change views without closing it.

3.2.6.11 Label Properties dialog

This dialog controls the text and display properties for a label.

Double-click on a label to display the dialog. Select **Add Label** from the **Object** menu to add a label to the component.

Label

Specify the text to display on the label.

Font

The text font. Click **Select Font...** to change it.

Select Text Color...

Click to select a color for the text.

Select Background Color...

Click to select the background color for the label.

Check Transparent Background

Check to make objects behind the label visible, and to ignore the background color setting.

Horizontal

Set the horizontal justification for the label text.

Vertical

Set the vertical justification for the label text.

Rotation

Set the orientation for the label.

Frame Thickness

Set the thickness of the label border.

Shadow Thickness

Set the thickness of the label drop shadow.

Display on Top

Check to display the label on top of any components below it.

Use these settings as default

Check to use the current settings as the default settings for any new labels.

3.2.6.12 New File dialog (File menu)

This dialog creates new projects and LISA source files.

To display the dialog, select **New File** from the **File** menu or click the **New** button.

Look in

Specify the directory for the new file.

File name

Enter the name for the new file.

File type

- If a project is not open, this box displays `.sgproj` by default to create a project.
- If a project is open, this box displays `.lisa` by default to create a LISA source file.

Add to

Active for non-`.sgproj` files. Check to enable the adding of the created file to the open project.

Select

Click to accept the name and path.

If the new file is of type `.sgproj`, System Canvas prompts for the top level LISA file.



Save time with the **Recently selected files** drop-down list. To remove a file, mouse over it and press **Delete**, or right-click and select **Remove from list** from the context menu.

Related information

[Select Top Component LISA File dialog](#) on page 67

3.2.6.13 New project dialogs

This section describes the dialogs that create new projects.

3.2.6.13.1 New Project dialog

This dialog creates new projects.

To display the dialog, select **New Project** from the **File** menu.

Look in

Specify the directory for the new project file.

File name

Enter the name for the new project.

If you select an existing file, the new project replaces the existing project.

File type

The default type for Fast Models projects is `.sgproj`.

Select

Click to accept the name and path.

For existing projects, System Canvas queries the replacement of the existing project with a new project of the same name.

After you click **Select**, the **Select Top Component LISA File** dialog appears.



The project file includes the path to the model repositories from the **Default Model Repositories** pane of the **Preferences** dialog.

Related information

[Preferences - Default Model Repository group](#) on page 71

3.2.6.13.2 Select Top Component LISA File dialog

This dialog controls the name of the top-level LISA file for a project.

After clicking **Select** in the **New Project** dialog, this dialog appears. By default, the filename for the top-level LISA file is the same as the project name. You can, however, specify a different name in this dialog.

3.2.6.14 Open File dialog

This dialog opens project files, LISA source files, and text documents.

To display the dialog:

- Select **Open File** from the **File** menu.
- Select a file in the **Component** window and select **Open** from the context menu.

Look in

Specify the directory.

File name

Enter the name of the file.

File type

Select the type of file.

Open

Click to open the file.

Open project file as text in source editor

Active for non-`.lisa` and for `.sgproj` files. Check to enable the opening of the file as plain text in the **Source** window.



- Use this option, for example, for a `.sgproj` file to manually edit the list of repositories. Such changes take effect after you close and reopen the file.
 - If you select a `.sgproj` file without checking this box, the project loads.
-



Save time with the **Recently selected files** drop-down list. To remove a file, mouse over it and press **Delete**, or right-click and select **Remove from list** from the context menu.

3.2.6.15 Port Properties dialog

This dialog controls port properties.

To display the **Port Properties** dialog, select a port or a connection.

- Select a component port in the **Block Diagram** view and:
 - Double-click on the port.
 - Click the **Properties** button.
 - Select **Object Properties** from the **Object** menu.
 - Right-click and select **Object Properties** from the context menu.
- Select a connection in the **Block Diagram** view and double-click to display the **Connection Properties** dialog. To display the **Port Properties** dialog:
 - Click the **Master Port Properties** button to display the properties for the master port.
 - Click the **Slave Port Properties** button to display the properties for the slave port.

Name

The name of the port.

Type

The type of port and the protocol.

Array size

For port arrays, the number of elements.

Show connections for port array index

For port arrays, enter an index value in the integer box to display only that element.

For individual ports of port arrays, this box displays the index for the selected port.

Port connections

- Sort the connections: click on the column headings.
- Change the connected port or address mapping: select a connection and click **Edit Connection**.
- Add a connection: select a connection and click **Add Connection**.
- Delete a connection: select it and click **Remove**.
- Change the priority of a single connection: select it and click **Increase Priority** or **Decrease Priority**.

Related information

[Connection Properties dialog](#) on page 61

3.2.6.16 Preferences dialog

This section describes the **Preferences** dialog (**File > Preferences**), which configures the working environment of System Canvas.

3.2.6.16.1 Preferences - Appearance group

This group sets the appearance of System Canvas.

Show Tool Tips

Display all tool tips.

Display tool bar text labels

Display the status bar labels.

Word wrap in source windows

Wrap long lines to display them within the source window.

Show splash screen on startup

Show the splash screen on startup.

Reload recent layout on startup

Reload the layout settings from the last modified project.

Recent files and directories

Set the number of directories and files shown in System Canvas file dialogs and menus, up to 32 directories and 16 files.

3.2.6.16.2 Preferences - Applications group

This group sets the application paths.



On Microsoft Windows, environment variables appear as `$MAXxxxx_HOME`. You can use this format instead of `%MAXxxxx_HOME%`.

Simulator Generator Executable

SimGen

Set the path to the `simgen.exe` file.

Command arguments

Set additional command-line options.

Path to Microsoft Visual Studio application 'devenv.com'

Select the path to the Microsoft Visual Studio `devenv.com` file. This application is the development environment and builds the model.

Reset to Defaults

Click to reset the application paths.

Apply

Click to save the changes.

Host Debugger Command Line

On Linux, set the command-line options. The default text is:

```
xterm -e gdb --args %ISIM%
```

where `%ISIM%` is a placeholder for the `isim_system` executable file.



On Linux, select the GCC compiler to build the model by using the SimGen command-line option `--gcc-path`.

Related information

[SimGen command-line options](#) on page 14

[Project Settings dialog](#) on page 73

3.2.6.16.3 Preferences - External Tools group

This group sets the tools that display the documentation.

use operating system file associations

Check to inactivate the external tool edit fields and buttons. Clear to activate them.



Note

This checkbox is not available on Linux.

3.2.6.16.4 Preferences - Fonts group

This group sets the application fonts.

Application

The application font.

Base fixed font

The **Source** view font.

Block Diagram Component Name

The component title block font.

Fonts depend on \$DISPLAY variable

Check to use the font set in the \$DISPLAY variable.

Reset to base size

Reset all font sizes to the selected value.

Reset to defaults

Click to reset the fonts to the factory settings.



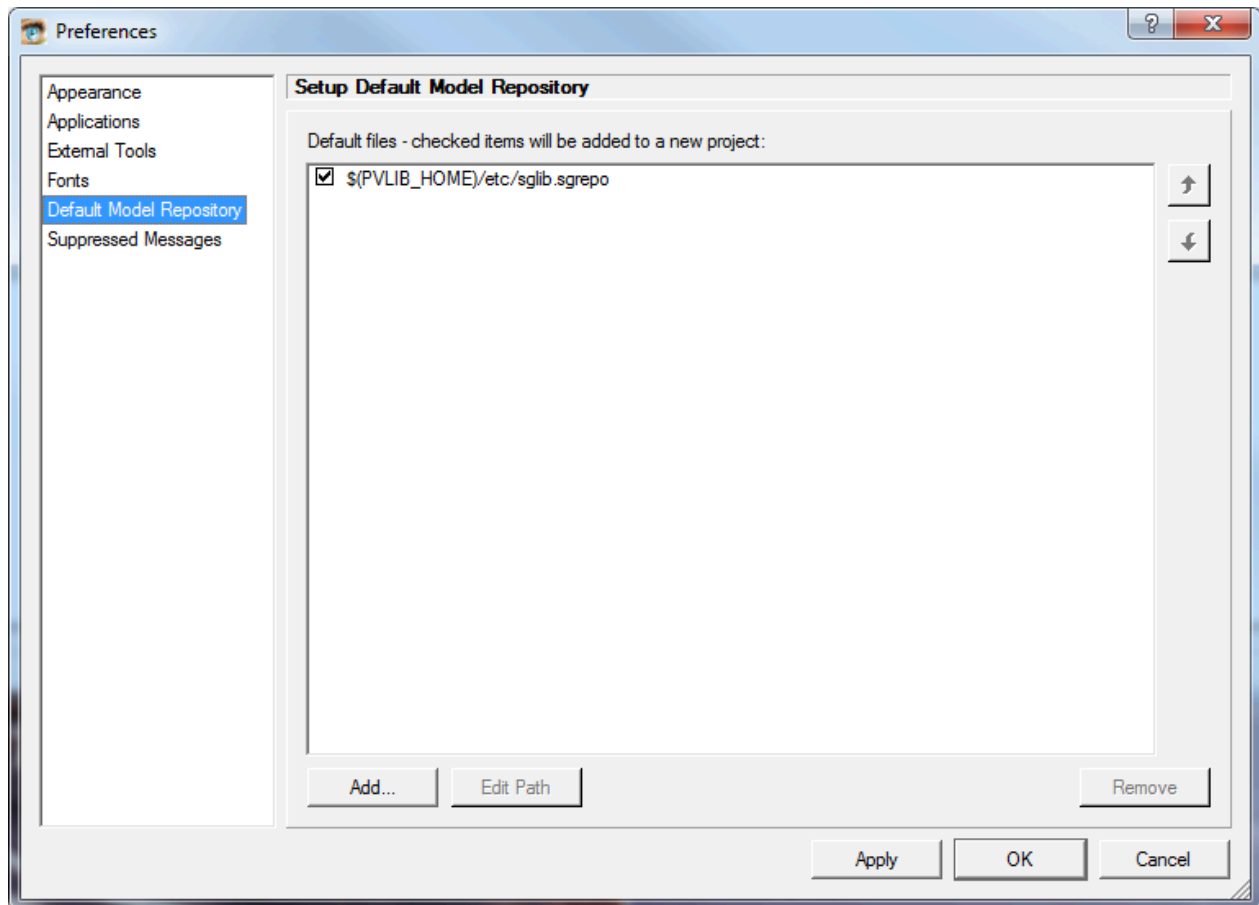
Note

If non-Latin characters are used in LISA code, the base fixed font must support them. The default font might not support non-Latin characters.

3.2.6.16.5 Preferences - Default Model Repository group

This group sets the default model repositories for new projects.

Figure 3-16: Preferences dialog, Setup Default Model Repository



To incorporate components into a system, System Canvas requires information about them, such as their ports, protocols, and library dependencies. For convenience, model repositories, such as `sglib.sgrepo`, group multiple components together and specify the location of the LISA files and the libraries that are needed to build them.

Default repositories are added by default to new projects. To add a repository to an existing project, use the Component window context menu.



Note

To enable the immediate use of models in new projects, System Canvas has a default entry `$(PVLIB_HOME)/etc/sglib.sgrepo`. This entry is not deletable, but clearing the checkbox deactivates it.

Add

Click **Add** to open a file selection dialog and add a new `.sgrepo` repository file to the list.

Select a directory to add all of the repositories in that directory to the list of repositories.

Edit Path

Select a repository and click **Edit** to edit the path to it.

The path to the default repository `$(PVLIB_HOME)/etc/sglib.sgrepo` is not editable.

Remove

Select a repository and click **Remove** to exclude the selected repository from new projects. This does not affect the repository itself.

The default repository `$(PVLIB_HOME)/etc/sglib.sgrepo` is not deletable.

File checkboxes

Check to automatically include the repository in new projects. Clear to prevent automatic inclusion, but to keep the path to the repository available.

Up/Down

Use the **Up** and **Down** buttons to change the order of repositories. File processing follows the repository order.

Related information

[Repository files](#) on page 9

3.2.6.16.6 Preferences - Suppressed messages group

This group lists the suppressed messages and controls their re-enabling.

Enable selected messages

Click to enable selected suppressed messages.

3.2.6.17 Project Settings dialog

This section describes the dialog (**Project > Project Settings**, or **Settings** toolbar button) that sets the project settings and customizes the generation process.

3.2.6.17.1 Project top-level settings

This part of the dialog sets the project build options.

Top level component

- Enter a name into the **Top Level Component** edit box.
- Click **Use Current** to set the component in the workspace as the top component.
- Click **Select From List** to open a dialog and select any component in the system.

Configuration

- Select an entry from the drop-down list to use an existing configuration.

- Click **Add New** to create a new configuration. A dialog prompts for the name and a description. Use **Copy values from** to select a configuration to copy the settings values from. This can be an existing configuration or a default set of configuration settings.
- Click **Delete** to delete the selected configuration from the list.

The values default to those of the active configuration.

Selecting a configuration in this dialog does not set the configuration in the **Select Active Project Configuration** drop-down box on the main window. System Canvas stores the configuration set in this dialog in the project file, to use if you specify it for a build. You can use this control to specify all of the configurations for a project, to simplify switching active configurations.



Note

If you build systems on Microsoft Windows workstations, other Microsoft Windows workstations need the matching support libraries to run the systems:

Debug builds

Microsoft Visual Studio.

Release builds

Microsoft Visual Studio redistributable package.

3.2.6.17.2 Parameter category panel

This section describes the **Parameter category** panel, which lists parameters for the selected build, under different views.

3.2.6.17.2.1 Parameters - Category View

This view lists categories and the parameters for the selected category.

Top-level configuration details

Select the top-most category item to configure the project settings.

Table 3-5: Configuration parameters in the Category View

Control name	Parameter
Configuration name	CONFIG_NAME
Platform/Linkage	PLATFORM
Compiler	COMPILER
Configuration description	CONFIG_DESCRIPTION
Build directory	BUILD_DIR

Targets

Select the **Targets** item to configure the build target parameters.

Table 3-6: Target parameters in the Category View

Control name	Parameter
SystemC component	TARGET_SYSTEMC
SystemC component with auto-bridging	TARGET_SYSTEMC_AUTO
Link the set target to static SystemC library	USE_STATIC_SYSTEMC_LIB Note: This parameter is deprecated and will be removed in a future release.
SystemC integrated simulator	TARGET_SYSTEMC_ISIM

Debugging

Select the **Debugging** item in the panel to configure the debug parameters.

Table 3-7: Debugging parameters in the Category View

Control name	Parameter
Enable model debugging	ENABLE_DEBUG_SUPPORT
Source reference	GENERATE_LINEINFO
Verbosity	VERBOSITY
SystemC executable	SYSTEMC_EXE
SystemC arguments	SYSTEMC_COMMAND_LINE

Sim Generator

Select the **Sim Generator** item in the panel to configure the Simulation Generator parameters.

Table 3-8: Simulation Generator parameters in the Category View

Control name	Parameter
Simgen options	SIMGEN_COMMAND_LINE
Warnings as errors	SIMGEN_WARNINGS_AS_ERRORS
Using namespace std	ENABLE_NAMESPACE_STD
Make options	MAKE_OPTIONS

Compiler

Select the **Compiler** item in the panel to configure the compiler parameters.

Table 3-9: Compiler parameters in the Category View

Control name	Parameter
Pre-Compile Actions	PRE_COMPILE_EVENT
Include Directories	INCLUDE_DIRS
Preprocessor Defines	PREPROCESSOR_DEFINES
Compiler Settings	ADDITIONAL_COMPILER_SETTINGS
SCX Library Settings	ADDITIONAL_SCX_LIB_SETTINGS
Enable pre-compiling	ENABLE_PRECOMPILE_HEADER

Linker

Select the **Linker** item in the panel to configure the linker parameters.

Table 3-10: Linker parameters in the Category View tab

Control name	Parameter
Pre-Link Actions	PRE_LINK_EVENT
Linker Settings	ADDITIONAL_LINKER_SETTINGS
Post-Build Actions	POST_BUILD_EVENT
Post-Clean Actions	POST_CLEAN_EVENT
Disable suppression of symbols	DISABLE_SYMBOL_SUPPRESSION

3.2.6.17.2.2 Parameters - List View

This view lists the parameters and their values. Reorder them by clicking on a column heading.

3.2.6.17.2.3 Parameters - Tree View

This view displays parameters in a tree structure, with expandable categories.

3.2.6.17.3 Project parameter IDs

The parameters that configure a project. These parameters can either be set in the System Canvas **Project Settings** dialog or in the sgproj file passed to SimGen.

Table 3-11: List of parameters shown in List View

Parameter ID	Parameter name	Default	Description
ADDITIONAL_COMPILER_SETTINGS	Compiler settings	-	Compiler settings. If your C++ source code uses C++14 syntax, specify <code>-std=c++14</code> in this parameter. For Microsoft Windows, consult the Visual Studio documentation.
ADDITIONAL_LINKER_SETTINGS	Linker settings	-	Linker settings. For Microsoft Windows, consult the Visual Studio documentation.
ADDITIONAL_SCX_LIB_SETTINGS	SCX library settings	-	Compiler flags specified here are added to the scx library compiler flags.
BUILD_DIR	Build directory	-	Build directory. The path can be absolute or relative to the location of the project file.

Parameter ID	Parameter name	Default	Description
COMPILER	Compiler	-	The compiler to use to build this configuration. One of the following: vc2019 Microsoft Visual Studio 2019. gcc The first gcc version in the Linux search path. gcc-9.3 GCC 9.3. gcc-10.3 GCC 10.3. gcc-12.3 GCC 12.3.
CONFIG_DESCRIPTION	Configuration description	-	Description of the configuration.
CONFIG_NAME	Configuration name	-	Name of the configuration. For example <code>Linux64-Debug-GCC-9.3</code> .
DISABLE_SYMBOL_SUPPRESSION	Disable suppression of symbols	0	If true, stop SimGen from adding the <code>-fvisibility=hidden</code> flag to mark symbols as hidden.
ENABLE_DEBUG_SUPPORT	Enable model debugging	0	Use implementation-defined debug support.
ENABLE_NAMESPACE_STD	Enable namespace std	1	Use namespace std: 1 (true) Generate using <code>namespace std</code> and place in the code. 0 (false) Specify the namespace. This setting might reduce compilation time.
ENABLE_PRECOMPILE_HEADER	Enable precompiling	0	Precompile headers if true.
GENERATE_LINEINFO	Source reference	"LISA Code (incl headers) "	Control line redirection in the generated model source code: "LISA Code" Source code. "LISA Code (incl. headers) " Source and header. "Generated Code" No line redirection at all.
INCLUDE_DIRS	Include directories	-	Additional include directories. Separate multiple entries with semicolons. Non-absolute paths are relative to the sgproj file.
MAKE_OPTIONS		-	Additional options to pass to make. Linux only.

Parameter ID	Parameter name	Default	Description
PLATFORM	Platform/linkage	-	Host platform for which this configuration is valid. One of: Linux64 Linux x86-64. Linux64_armv81 Arm® AArch64 Linux. Win64 64-bit Microsoft Windows, linked against release runtime library. Win64D 64-bit Microsoft Windows, linked against debug runtime library.
POST_BUILD_EVENT	Postbuild actions	-	Commands to execute after building the model. Separate multiple entries with semicolons.
POST_CLEAN_EVENT	Post-clean actions	-	Commands to execute after the standard <code>clean</code> has completed.
PRE_COMPILE_EVENT	Precompile actions	-	Commands to execute before starting compilation. Applies to Microsoft Windows only. Separate multiple entries with semicolons.
PREPROCESSOR_DEFINES	Preprocessor defines	-	Preprocessor defines. Separate multiple entries with semicolons.
PRE_LINK_EVENT	Prelink actions	-	Commands to execute before starting linking. Applies to Microsoft Windows only. Separate multiple entries with semicolons.
SIMGEN_COMMAND_LINE	SimGen options	-	Additional command line options to pass to SimGen.
SIMGEN_WARNINGS_AS_ERRORS	Warnings as errors	"0"	If 1 (<code>true</code>), treat LISA parsing and compiler warnings as errors.
SYSTEMC_COMMAND_LINE	SystemC arguments	-	Command-line arguments for SystemC executable.
SYSTEMC_EXE	SystemC executable	-	Name of custom executable running exported SystemC model.
TARGET_SYSTEMC	SystemC component	0	If 1 (<code>true</code>), build a library with an exported SystemC component.
TARGET_SYSTEMC_AUTO	SystemC component with auto-bridging	0	If 1 (<code>true</code>), build a library with a SystemC component with auto bridging.
TARGET_SYSTEMC_ISIM	SystemC integrated simulator	0	If 1 (<code>true</code>), build a SystemC ISIM executable.
USER_SYSTEMC_DYNLIB	User specified SystemC shared library path	NULL	Set this parameter to the full path of a static or dynamic SystemC library. This parameter overrides the default location where SimGen looks for it, which is specified by the <code>SYSTEMC_HOME</code> environment variable. For more information, see Linking against the SystemC library in the Fast Models User Guide.
USER_SYSTEMC_MAIN	User defined SystemC main file path	NULL	Set this parameter to a file that defines a custom <code>sc_main()</code> function, for an ISIM target (<code>TARGET_SYSTEMC_ISIM</code>). SimGen uses this function instead of generating a default one. This parameter must include the path relative to the build directory.

Parameter ID	Parameter name	Default	Description
USE_STATIC_SYSTEMC_LIB	Link the set target to static SystemC library.	0	Note: This parameter is deprecated and will be removed in a future release. Link to a static rather than dynamic SystemC library. By default, the <code>SYSTEMC_HOME</code> environment variable specifies the location of the library, but you can override this using parameter <code>USER_SYSTEMC_DYNLIB</code> .
VERBOSITY	Verbosity	"Off"	Verbosity level: "Sparse", "On", or "Off".

3.2.6.18 Protocol Properties dialog

This dialog displays the properties of protocols.

Select a protocol from the **Protocols** list, right-click on it and select **Properties** to display the properties.

Protocol name

The name of the protocol.

File

The file that defines the protocol.

Repository

The repository that contains the reference to the file path.

Description

A description dating from the addition of the file to the project.

Methods

A panel that displays the LISA prototypes of methods, or behaviors, available for the protocol. The values are for reference only. They are not editable.

Properties

A panel that displays the properties for protocol. The values are for reference only. They are not editable.

3.2.6.19 Run dialog

This dialog specifies the actions that execute to run a selected target.

There are actions for different targets, and additional options.

To display the dialog, click **Run** from the **Project** menu.

Select command to run

Select the executable to run.

Full command line

Adjust the command line that System Canvas generates, for example, add parameters or change the location of the application to load onto the executable.

Effective command line

Shows the complete command line with expanded macros and environment variables, ready for execution.

ISIM system

Run the model as an ISIM system. The initial command line options come from project settings and user preferences.

Custom

Specify the command line in **Full command line**.

Recent

Select a recent command.

Insert Placeholder Macro

Insert a macro or environment variable from drop-down list at the current cursor position in **Full command line**. System Generator expands them to build the complete command line.

%ISIM%

The full absolute path of the ISIM executable.

%BUILD_DIR%

The relative path to the build directory (relative to project path).

%DEPLOY_DIR%

The relative path to the deploy directory (identical to %BUILD_DIR%).

%PROJECT_DIR%

The full absolute path to the directory of the project.

Launch in background

Run an application asynchronously in a separate console window. Use this if the application requests user input or if the output is long.

Clear History

Remove all the recent entries from command history. This also removes corresponding items from the System Canvas main menu.

3.2.6.20 Self Port dialog

Use this dialog to add a port to the top-level component.

To display the dialog, without having anything selected in the **Block Diagram** view, click **Add Ports**, or click **Add Port** from the **Object** menu.

Instance name

The name of the port.

Array size

The number of ports, for a port array. Leave the box empty, or enter 1, for normal ports.

Protocol

The name of the protocol for the port. To display a list of protocols, click **Select...**

Type

Master port or **Slave Port**.

Attributes

- **Addressable** for bus ports.
- **Internal** for ports between subcomponents. The port is not visible if the component is added to a system.

Create LISA method templates according to selected protocol

Select an option from the drop-down list to create implementation templates for methods, or behaviors, for the selected protocol:

- Do not create method templates.
- Create only required methods. This is the default.
- Create all methods, including optional behaviors.

This creates only methods corresponding to the selected port type, that is, for either master or slave.

Editing the existing port might create new methods, but does not delete existing methods.

Mirror port image

Reverse the direction of the port image.

4. Iris Monitor

Iris Monitor is a debugger for Fast Models. It is designed to address the needs of virtual platform developers who need to see the detailed behavior of a Fast Models system.

It is not intended to act as a complete software development debugger. For application debugging environments, use other tools such as [Arm Development Studio](#).

Iris Monitor can connect to any model that supports the Iris debug interface. It supports full system debugging, and multiple instances of Iris Monitor stay fully synchronized while debugging different cores running within a single system.

Iris Monitor is included in the Fast Models package. For installation information for the Fast Models package, see [Installing Fast Models](#) in the Fast Models User Guide.

4.1 Key features of Iris Monitor

This section describes the key features of Iris Monitor.

- Full simulation control at instruction level.
- Low-level disassembly display.
- Complex register display including register groups and compound registers.
- Memory windows with support for multiple memory spaces and bit widths.
- Breakpoints on memory locations.
- A browser-based GUI with customizable views.



Not all Fast Models components support all of these features. For example, typically only components that execute code, such as CPUs, can show a memory view.

4.2 Retargetable debugger

Iris Monitor supports retargetable debugging of any target that supports the Iris debug interface.

All target-related information, such as the disassembly and resources, for example register files and flags, is contained in the target model library.

Iris Monitor communicates with the target using Iris to retrieve the static target-specific information, for example a register file. It can then determine the target state and control execution.

4.3 Get started with Iris Monitor

Iris Monitor is a browser-based web application that can connect to and debug a simulation running on either a local or remote host.

Before you begin

Iris Monitor is included in the Fast Models package. For installation information, see [Installing Fast Models](#) in the *Fast Models User Guide*.

Procedure

To connect Iris Monitor to a simulation, follow these steps:

1. Launch the simulation with the `-I` option to start the Iris server. This option enables Iris Monitor to connect to the simulation. For example:

```
./FVP_Base_Cortex-A710 ... -a cluster*.cpu*=./images/fireworks_Cortex-A710.axf -I
```



Note

You cannot use Iris Monitor to load an application on the simulation. Instead, use the `-a` option when launching the simulation to load the application.

The simulation starts running, and by default displays a visualization window. It shows an instruction count of zero because the simulation is waiting for the debugger to connect to the Iris server.

Figure 4-1: Visualization window shown while waiting for a connection



2. In another terminal, launch Iris Monitor using the `irismonitor` command:
 - If the simulation, Iris Monitor, and the web browser are all running on the same machine, launch Iris Monitor with no options.
 - If the simulation and Iris Monitor are running on different machines, launch Iris Monitor with the following options:
 - iris-host <IP address>**
Specifies the IP address of the remote machine on which the simulation is running.
 - iris-port <port number>**
Specifies the port number on the remote machine on which the Iris server is listening for new connections.

- If the web browser and Iris Monitor are running on different machines, launch Iris Monitor with the following options:

--http-host <IP address>

Specifies the hostname or IP address that the HTTP server binds to. If not set, an IP address of 127.0.0.1 (local host) is used, which prevents a web browser on a remote machine from connecting. Specify 0.0.0.0 to allow remote connections.

--http-port <port number>

Specifies the port number that the HTTP server binds to. The default is 8080.

Iris Monitor automatically connects to the running simulation and prints an IP address to the terminal.

3. Paste the IP address shown in the terminal into the address bar of the web browser.

Results

If the connection is successful, the browser displays the [4.4 Iris Monitor GUI](#) on page 84.

If the connection fails, for example because there is no Iris-enabled simulation running, a dialog is displayed. Click **Retry** to try again without having to restart Iris Monitor:

Figure 4-2: Iris Monitor UI connection failure dialog



Next steps

To disconnect from the simulation and close Iris Monitor, close the browser tab.

Related information

[FVP command line options](#)

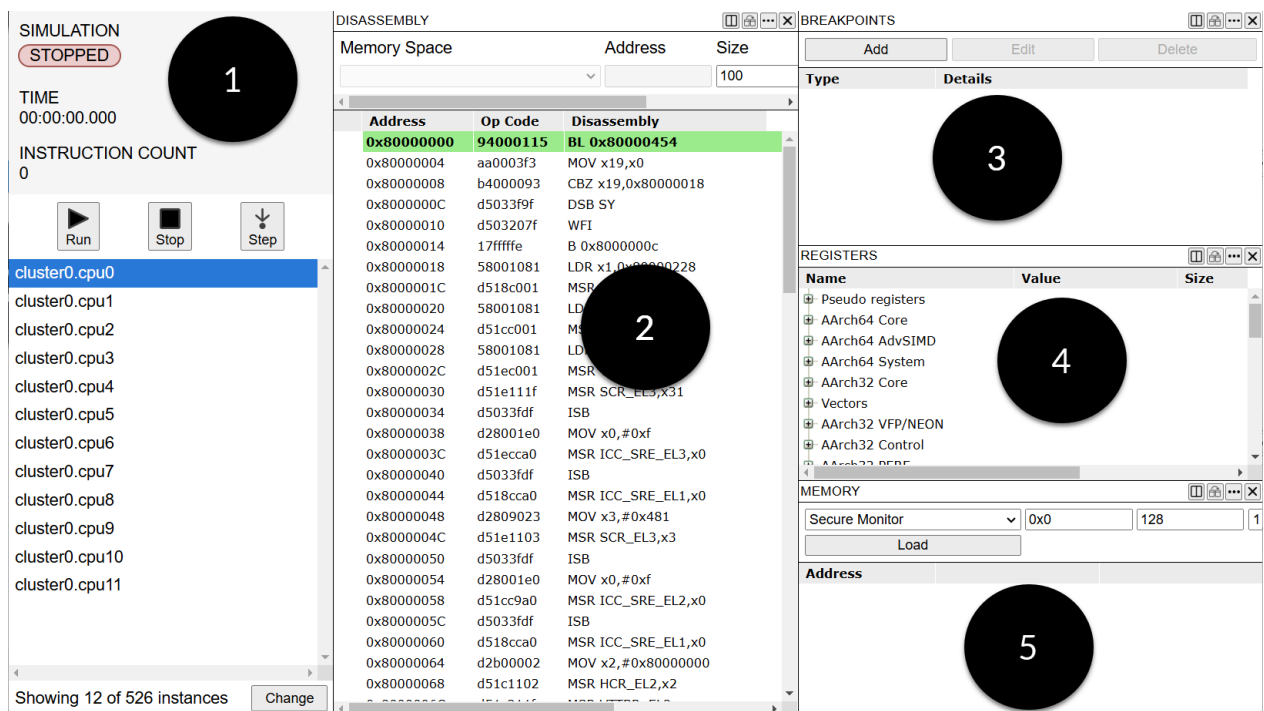
4.4 Iris Monitor GUI


The Iris Monitor GUI consists of a simulation status and control area and a customizable set of views which display information about the selected instance.

The **Disassembly**, **Breakpoints**, **Registers**, and **Memory** views are shown by default. To show one

of the other views, use the **Switch View** button . This figure shows the default layout:

Figure 4-3: Iris Monitor UI at startup



Each view has a **View menu** button,  which provides some extra, view-specific options, for example to refresh the view.

The numbered items in this screenshot are described in the following sections:

1. [4.4.1 Simulation status and control](#) on page 86
2. [4.4.2 Disassembly view](#) on page 87
3. [4.4.3 Breakpoints view](#) on page 89
4. [4.4.4 Registers view](#) on page 89
5. [4.4.5 Memory view](#) on page 90

Other available views, not shown by default:

- [4.4.6 Instances view](#) on page 91
- [4.4.7 Overview view](#) on page 92
- [4.4.8 Parameters view](#) on page 93

4.4.1 Simulation status and control

The simulation status and control area displays the simulation status and enables simulation run control and instance selection.

The simulation status area shows:

- Whether the simulation is running or stopped.

If the simulation was not stopped by the user clicking the **Stop** button, some extra information is provided about why the simulation was stopped:

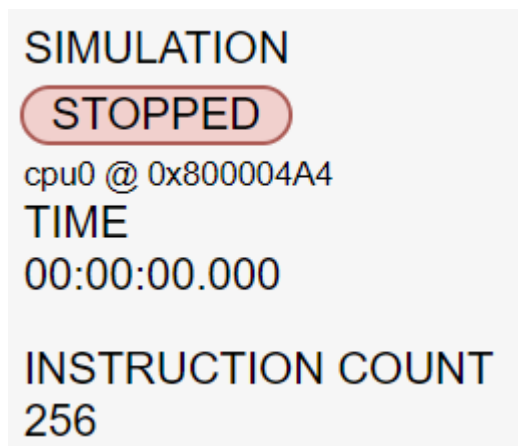
- Which component caused the simulation to stop
- The address it stopped at

This information can help to identify which breakpoint was hit, or to debug exceptions.

- Elapsed simulation time.
- The total number of instructions executed by the simulation.

For example:

Figure 4-4: Breakpoint detail



The control area provides the following buttons:

Run

Starts or resumes the simulation.

Stop

Stops the execution.

Step

Steps the execution of a stopped component by a single instruction.

Alternatively, use the following keyboard shortcuts:

- Run = **F5**
- Stop = **F6**
- Step = **F7**

The instance selection area shows a filtered list of the components in the model that have an Iris instance.

To select an instance to view or control, click it in the list. When you select a different instance, Iris Monitor populates the other views with the state of that instance.

When you start Iris Monitor, it only lists instances that support software execution. To change the list of instances shown:

1. Click **Change**
2. Select or clear instances in the list that appears
3. Click **Select**



Not all types of instances support all of the views. For example, typically only components that execute code, such as CPUs, can show a memory view.

4.4.2 Disassembly view

The **Disassembly** view displays the instruction disassembly if the selected instance can execute software.

The disassembly starts from either:

- The current PC address if the **Track PC** checkbox is selected.
- The address in the **Address** field if the **Track PC** checkbox is cleared.

When the **Track PC** checkbox is cleared, you can use the other controls to select the address range shown in the **Disassembly** view:

DISASSEMBLY							
Memory Space	Address	Size	Mode	<input type="checkbox"/> Track PC			
Secure Monitor	0x800004B0	1000	Auto	<input type="button" value="Load"/>			

These controls are:

- The **Memory Space** field lets you select which address space to view. For example, you can use this field to select a physical memory view instead of the virtual address space. The default value is the memory space of the initial execution address when the application is started.

For descriptions of the memory space names see the [Iris User Guide](#).

- The **Address** field provides the address that disassembly starts from. To update the view, press **Enter** or click the **Load** button after you change the address.
- The **Size** field is the number of instructions to show in the **Disassembly** view, by default 100.
- The **Mode** field allows you to force disassembly to be for a particular instruction set. By default this is set to **Auto**, which means that the **Disassembly** view detects the instruction set to use automatically.
- The **Load** button updates the **Disassembly** view after you have made changes to the other fields.



If you scroll to the end of the disassembly, there is a **Load More** button which loads an additional 50 lines of disassembly.

You can add code breakpoints to the disassembly by clicking in the gutter beside the **Address** column at the required line:

DISASSEMBLY			
Memory Space		Address	Size
<input type="text"/>		<input type="text"/>	100
	Address	Op Code	Disassembly
	0x80000000	94000115	BL 0x80000454
	0x80000004	aa0003f3	MOV x19,x0
	0x80000008	b4000093	CBZ x19,0x80000018

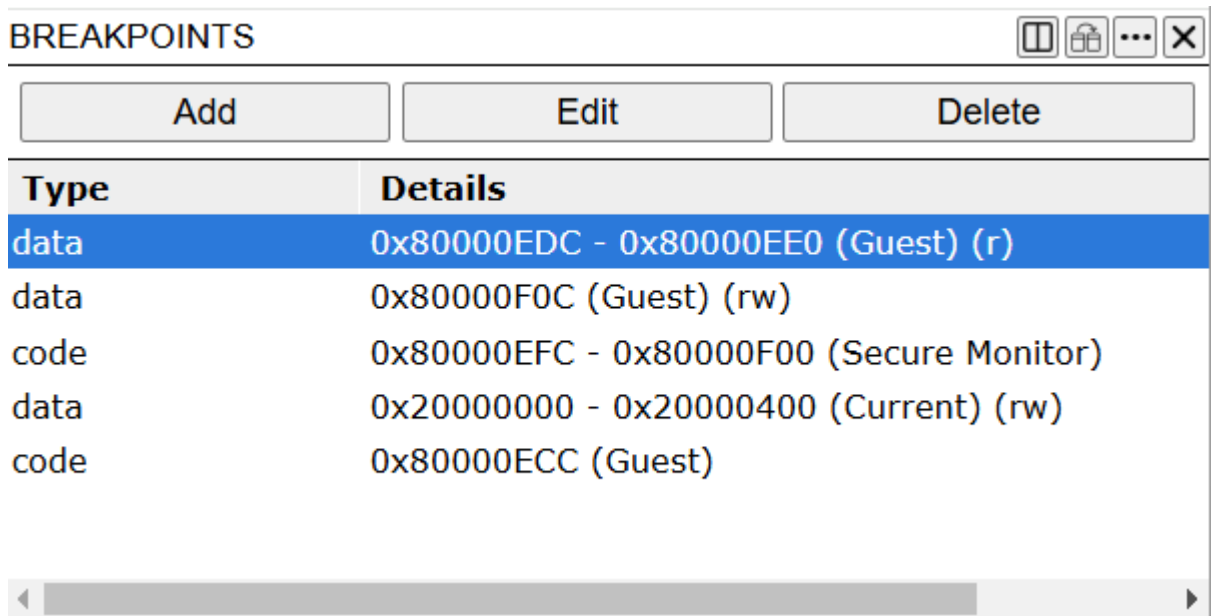
Alternatively, you can add, edit, and delete breakpoints using the [4.4.3 Breakpoints view](#) on page 89.

4.4.3 Breakpoints view

The **Breakpoints** view displays all code and data breakpoints for the currently selected instance.

For each breakpoint, it displays the type, the address or address range, and the memory space. For data breakpoints, it also shows the access type.

Figure 4-7: Breakpoints view



The following breakpoint types are supported:

Code

Hit when the instruction at the associated address or address range is executed.

Data

Hit when an instruction accesses the associated address or address range.

This view also allows you to:

- Add a breakpoint on the currently selected instance, if it supports breakpoints.

An alternative way to add a code breakpoint is to click in the gutter beside the **Address** column at the required line in the **Disassembly** view.

- Edit the properties of an existing breakpoint. First select the breakpoint, then click **Edit**.
- Delete a breakpoint from the current instance. First select the breakpoint then click **Delete**.

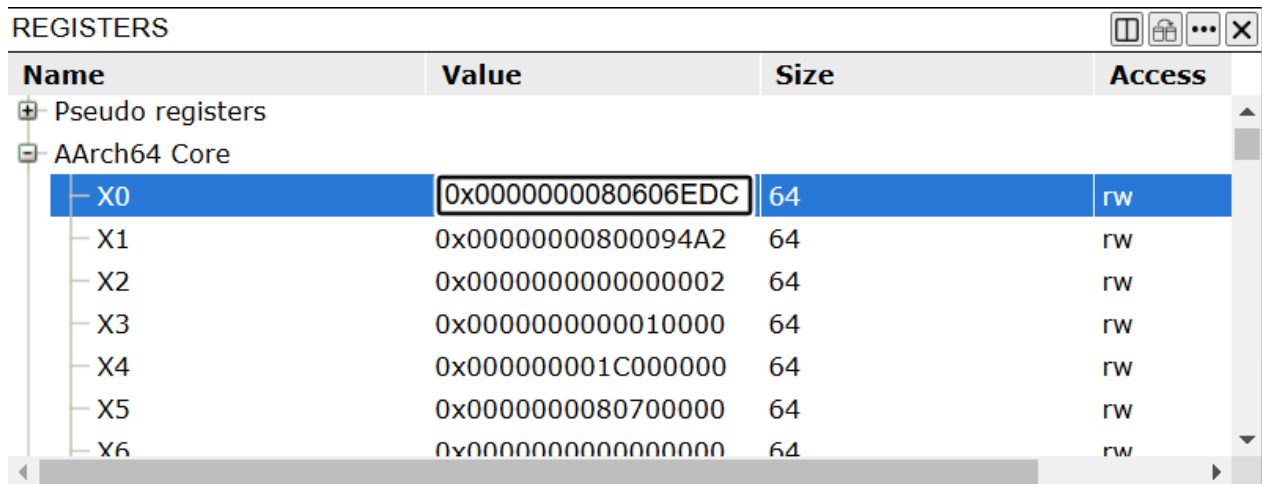
4.4.4 Registers view

The **Registers** view displays the hierarchy of register groups, registers, and register bit fields, and their values, for the selected instance.

For each register, its value, size, and whether it is read-only or read-write, is shown.

To change the value of a register, double click the **Value** field, update the value, then press **Enter** to save the new value. To close the editor without updating the value, press **Esc**.

Figure 4-8: Editing a register value



Name	Value	Size	Access
+	Pseudo registers		
+	AArch64 Core		
X0	0x0000000080606EDC	64	rw
X1	0x00000000800094A2	64	rw
X2	0x0000000000000002	64	rw
X3	0x00000000000010000	64	rw
X4	0x000000001C000000	64	rw
X5	0x0000000080700000	64	rw
X6	0x0000000000000000	64	rw

4.4.5 Memory view

The **Memory** view displays a table of memory values for an address or address range, for the selected instance, if it exposes memory.

It has the following fields:

- The **Memory Space** field lets you select which address space to view. For example, you can use this to select a physical memory view instead of the virtual address space.

For descriptions of the memory space names see the [Iris User Guide](#).

- The **Address** field provides the address of the start of the memory block. To update the view after you change the address, either press **Enter** or click **Load**.
- The **Size** field lets you change the size in bytes of the memory block. The default is 128.
- The **Access Width** field controls the access width in bytes. For example, to show memory in 32-bit words, set the access width to 4. The view automatically updates when you change the access width. The default is 1.
- The **Load** button updates the memory view when you have made changes to other fields.

For example, the following **Memory** view displays 128 bytes of memory, starting at address 0x80000000 in the Secure Monitor address space as 4-byte values:

Figure 4-9: Memory view

MEMORY

Secure Monitor

0x80000000

128

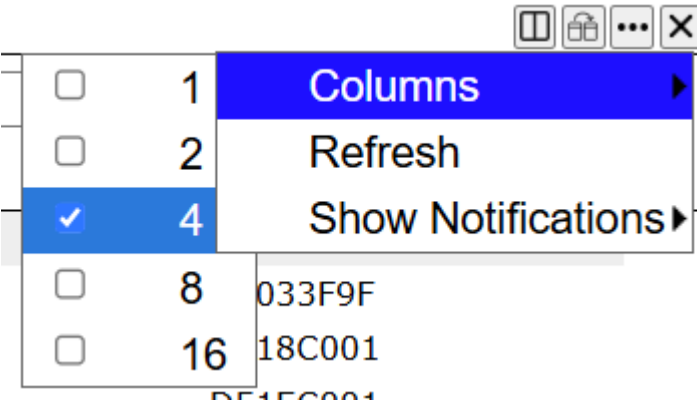
4

Load

Address				
80000000	00940115	AA0003F3	B4000093	D5033F9F
80000010	00FFFFFF	17FFFFFFE	58001081	D518C001
80000020	58001081	D51CC001	58001081	D51EC001
80000030	D51E111F	D5033FDF	D28001E0	D51ECCA0
80000040	D5033FDF	D518CCA0	D2809023	D51E1103
80000050	D5033FDF	D28001E0	D51CC9A0	D5033FDF
80000060	D518CCA0	D2B00002	D51C1102	D51C211F
80000070	D53800A0	D51C00A0	D5380000	D51C0000

To edit a memory value, double click the value, update it, then press **Enter** to save the new value. To close the editor without updating the value, press **Esc**.

To change the number of columns shown, click the **View** menu, select **Columns**, then select the required value:



4.4.6 Instances view

The **Instances** view displays a tree of all instances in the simulation and their sub-instances.

For each instance, it displays the component name and, if it is available, the component type.

Figure 4-11: Instances view

INSTANCES				
Instance Name	Component Name	Component Type		
address_map_terminator	PVBusMapper			
bp	BasePlatformPeripherals	System		
cci400	CCI400	SystemIP		
cciinterconnect	PVCache			
cciregisters	CCIRegisters	SystemIP		
clocktimer	ClockTimerThread	Clocking		
timer	ClockTimerThread64	Clocking		
thread	SchedulerThread	Scheduler		
thread_event	SchedulerThreadEvent	Scheduler		
clockdivider0	ClockDivider	Peripheral		
cluster0	Cluster_ARM_Cortex-A...	Cluster		
AMU	PVBusLogger			
mapper	PVBusMapper			



Instances view is not shown by default. To display it, click **Switch View** from any other view.

4.4.7 Overview view

The **Overview** view gives an overview of the state of all executable instances in the simulation.

It shows a tree of the instances and sub-instances. Typically, this represents CPU clusters and the individual CPU cores within them. For each instance it lists:

- The run state, either running or stopped.



Note

The **Run State** column shows the state of the core, not the simulation. It is not affected by stopping the simulation using the **Run**, **Stop**, or **Step** buttons or even by stopping on a breakpoint.

- The count of instructions executed.
- The current PC and SP.

Figure 4-12: Overview view

OVERVIEW			
Run State	Instruction count	PC	SP
Running	1,320,130,037	80001724 (Guest)	
cpu0 (ARM_Cortex-A710)			
Standby WFI	22	80000004 (Secure Monitor)	
cpu1 (ARM_Cortex-A710)			
Standby WFI	22	80000004 (Secure Monitor)	
cpu2 (ARM_Cortex-A710)			
Standby WFI	22	80000004 (Secure Monitor)	
cpu3 (ARM_Cortex-A710)			
Standby WFI	22	80000004 (Secure Monitor)	



Overview view is not shown by default. To display it, click **Switch View** from any other view.

4.4.8 Parameters view

The **Parameters** view lists all of the parameters supported by the selected instance. For run-time parameters, you can also edit their values.

For example:

Figure 4-13: Parameters view

PARAMETERS						
Name	Value	Modifiable	Type	Default	Min	Max
l2cache-maintenance_lat...	0x0000000000000000	run-time	int	0x0000000000000000...	0x00000000...	0xFFFFFFFFFFFF
CFGEND	0x0	init-time	bool	0x0	0x0	0x1
vfp-enable_at_reset	0x0	init-time	bool	0x0	0x0	0x1
semihosting-enable	0x1	init-time	bool	0x1	0x0	0x1
semihosting-ARM_SVC	0x00000000000123456	init-time	int	0x00000000000123...	0x00000000...	0x00000000000F...
semihosting-Thumb_SVC	0x000000000000000AB	init-time	int	0x00000000000000...	0x00000000...	0x000000000000...
semihosting-A64_HLT	0x0000000000000F00	init-time	int	0x0000000000000F...	0x00000000...	0x000000000000...
semihosting-A32_HLT	0x0000000000000F00	init-time	int	0x0000000000000F...	0x00000000...	0x000000000000...

The **Modifiable** column shows whether a parameter is modifiable at initialization (**init-time**), or when the model is running (**run-time**).

To edit the value of a run-time parameter, double click the **Value** field, update the value, then press **Enter** to save the new value. To close the editor without updating the value, press **Esc**. If you try to set an invalid value, an error message is displayed.



The **Parameters** view is not shown by default. To display it, click **Switch View** from any other view.

4.5 Iris Monitor command-line options

Use these configuration options when launching Iris Monitor.

Table 4-1: Iris Monitor command-line options

Long form	Short form	Description
--help	-h	List the Iris Monitor command-line options then exit.
--http-host=<host>	-	Local machine host name or IP address for Iris Monitor to bind to to accept web browser connections. The default is 127.0.0.1.
--http-port=<port>	-P	Port for Iris Monitor to bind to to accept web browser connections. The default is 8080.
--iris-host=<host>	-	Hostname or IP address of the model to connect to. The default is 127.0.0.1.
--iris-port=<port>	-p	Port of the model to connect to. The default is 7100.
--version	-	Print the Iris Monitor version number, then exit.

4.6 Iris Monitor tutorial

This tutorial demonstrates some common use cases for Iris Monitor.

Procedure

1. Launch your model, in this example FVP_Base_Cortex-A710, with the required parameters, including the **-I** option:

```
./FVP_Base_Cortex-A710 -C cluster0.NUM_CORES=1 -C bp.secure_memory=false -a cluster*.cpu*=../images/fireworks_Cortex-A710.axf -I
```

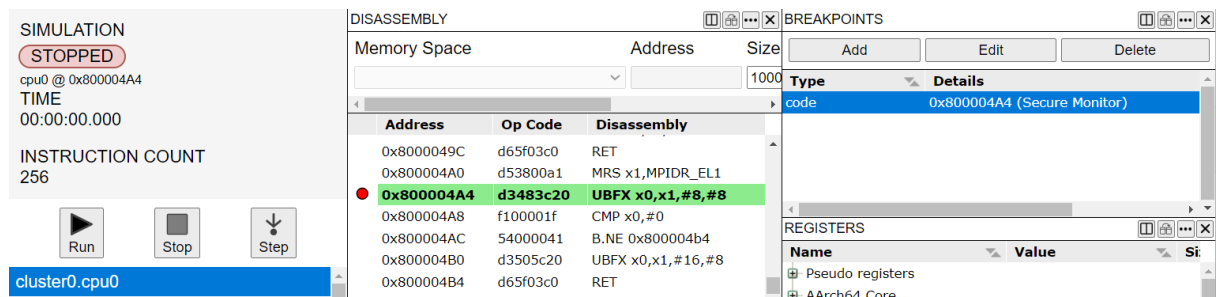
2. In another terminal, launch Iris Monitor:

```
./irismonitor
```

3. In a web browser, paste the address `http://127.0.0.1:8080` into the address bar.
 - The Iris Monitor UI with the default set of views appears in the browser. The simulation state is stopped, and the time and instruction count are both zero.
 - The **Disassembly** view displays the addresses and opcodes of the disassembled instructions, and the disassembly. As the **Track PC** checkbox is selected by default, the disassembly begins at the PC address, and shows 100 instructions by default.
4. Click **Run**:

Figure 4-14: Run controls

- The Simulation status changes from **STOPPED** to **RUNNING**.
 - The time and instruction count start incrementing.
- Click **Stop**
The Simulation status changes from **RUNNING** to **STOPPED**.
 - In the **Disassembly** view, clear the **Track PC** checkbox.
 - The **Memory Space** field displays the memory space of the PC.
 - The **Address** field contains the address of the first line that is displayed in the disassembly.
 - Set a breakpoint by clicking in the gutter beside the **Address** column at the required line of disassembly.
A red circle appears next to the corresponding disassembly line, and a new code breakpoint entry appears in the **Breakpoints** view.
 - Click the **Run** button to restart execution of the simulation.
Simulation time and instruction count start increasing. When execution reaches the instruction with the breakpoint, the simulation stops and the status view updates to show details about the hit breakpoint. If the disassembly range shown contains the location of the PC, that line is highlighted in green:

Figure 4-15: Hit breakpoint

- Change the breakpoint into a range breakpoint by selecting the breakpoint in the **Breakpoints** view and clicking the **Edit** button. Increase the size to 0x10 and click **Update**. The **Disassembly** view updates to show a marker for each row in the range:

Figure 4-16: Range breakpoint

DISASSEMBLY

Memory Space	Address	Size
<input type="text" value="Secure Monitor"/>	<input type="text" value="0x800004A0"/>	<input type="text" value="200"/>

Address	Op Code	Disassembly
0x800004A0	d53800a1	MRS x1,MPIDR_EL1
● 0x800004A4	d3483c20	UBFX x0,x1,#8,#8
0x800004A8	f100001f	CMP x0,#0
0x800004AC	54000041	B.NE 0x800004b4
0x800004B0	d3505c20	UBFX x0,x1,#16,#8
0x800004B4	d65f03c0	RET

10. View a block of memory in the **Memory** view:

- Change the **Memory space** to the value shown in the **Disassembly** view.
- Change the **Address** to a value shown in the **Disassembly** view and press **Enter**.
- Leave the default value for **Size** unchanged (128).
- Change the **Access width** value to 4. The number of columns displayed updates automatically and memory is reloaded.

Memory is displayed as a table of values organised by rows and columns. The default number of columns displayed depends on the access width. For an access width of 4, the default number of columns is 4.

You can change the number of columns using the **View** menu. For example the following **Memory** view shows 4 byte values with 8 columns:





Figure 4-17: Memory view

MEMORY

Secure Monitor	0x80000454	1024		
Load				

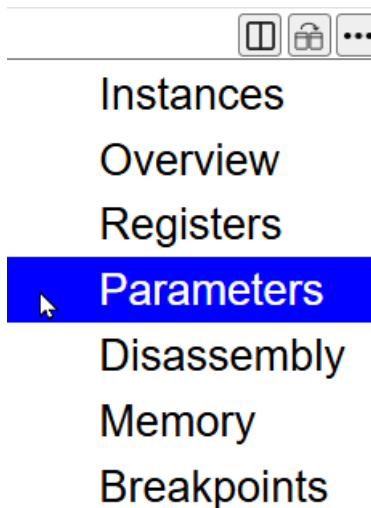
Address	D5380000	D3443C00	F1343C1F	540002C0	F134141F	540001C0	F100001F	54000041	D3505C20
80000454	D5380000	D3443C00	F1343C1F	540002C0	F134141F	540001C0	F100001F	54000041	D3505C20
80000554	F135281F	54000080	F134281F	54000105	1400000D	D53800A0			
80000654	D3401C00	8B000020	D65F03C0	D53800A1	D3483C20	F100001F			
80000754	D65F03C0	D53800A0	D3401C01	D3483C02	8B020820	D65F03C0	52800001	B9000001	
80000854	52800020	D65F03C0	52A00023	B8A30001	4AC14022	340000C2	D50320BF	D503205F	
80000954	485FFC03	4A414062	35FFFAA2	D65F03C0	52800021	7861001F	D65F03C0	D65F03C0	

11. Edit a register value. In the **Registers** view, expand the **AArch64 Core** register group. Double click the **Value** entry for register X0. As the register is writable, the value editor opens. Change the value to 0x1 and press **Enter**.

REGISTERS				   
Name	Value	Size	Access	
Pseudo registers				
PC_MEMSPACE	0x00000000	32	r	
AArch64 Core				
X0	0x0000000000000001	64	rw	
X1	0x0000000000000000	64	rw	
X2	0x0000000000000000	64	rw	

The new value is displayed.

- Open the **Parameters** view. As it is not open by default, you might have to open it using the **Switch View** button:



- Edit a parameter value. Double click the **Value** column of the `min-sync-level` parameter. This is a run-time parameter, so it is writable. Replace the value with `0x4`, then press **Enter**.
The message `Error writing parameter value` appears. This is because the value of 4 is invalid.
- Replace the value with `0x1`, then press **Enter**.

PARAMETERS

Name	Value	Modifiable
semihosting-stack_b...	0x00000000FF000000	init-time
semihosting-stack_li...	0x00000000FF000000	init-time
semihosting-cwd		init-time
enable_trace_specia...	0x0	init-time
trace_special_hlt_im...	0x0000000000000F00	init-time
RVBARADDR	0x0000000000000000	init-time
min_sync_level	0x0000000000000001	run-time
CRYPTODISABLE	0x0	init-time
max_code_cache_mb	0x0000000000000100	init-time

The new value is accepted and is displayed.

Proprietary Notice

This document is protected by copyright and other related rights and the use or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm Limited ("Arm"). No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether the subject matter of this document infringes any third party patents.

The content of this document is informational only. Any solutions presented herein are subject to changing conditions, information, scope, and data. This document was produced using reasonable efforts based on information available as of the date of issue of this document. The scope of information in this document may exceed that which Arm is required to provide, and such additional information is merely intended to further assist the recipient and does not represent Arm's view of the scope of its obligations. You acknowledge and agree that you possess the necessary expertise in system security and functional safety and that you shall be solely responsible for compliance with all legal, regulatory, safety and security related requirements concerning your products, notwithstanding any information or support that may be provided by Arm herein. In addition, you are responsible for any applications which are used in conjunction with any Arm technology described in this document, and to minimize risks, adequate design and operating safeguards should be provided for by you.

This document may include technical inaccuracies or typographical errors. THIS DOCUMENT IS PROVIDED "AS IS". ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, any patents, copyrights, trade secrets, trademarks, or other rights.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Reference by Arm to any third party's products or services within this document is not an express or implied approval or endorsement of the use thereof.

This document consists solely of commercial items. You shall be responsible for ensuring that any permitted use, duplication, or disclosure of this document complies fully with any relevant

export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arm’s customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of this document shall prevail.

The validity, construction and performance of this notice shall be governed by English Law.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its affiliates) in the US and/or elsewhere. Please follow Arm’s trademark usage guidelines at <https://www.arm.com/company/policies/trademarks>. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

PRE-1121-V1.0

Product and document information

Read the information in these sections to understand the release status of the product and documentation, and the conventions used in the Arm documents.

Product status

All products and Services provided by Arm require deliverables to be prepared and made available at different levels of completeness. The information in this document indicates the appropriate level of completeness for the associated deliverables.

Product completeness status

The information in this document is Final, that is for a developed product.

Revision history

These sections can help you understand how the document has changed over time.

Document release information

The Document history table gives the issue number and the released date for each released issue of this document.

Document history

Issue	Date	Confidentiality	Change
1130-00	19 November 2025	Non-Confidential	Update for v11.30.
1129-00	16 May 2025	Non-Confidential	Update for v11.29.
1128-00	19 February 2025	Non-Confidential	Update for v11.28.
1127-00	16 September 2024	Non-Confidential	Update for v11.27.
1126-00	19 June 2024	Non-Confidential	Update for v11.26.
1125-00	13 March 2024	Non-Confidential	New document for v11.25.

For information about the functional changes to Fast Models, see the [Fast Models Release Notes](#).

Conventions

The following subsections describe conventions used in Arm documents.

Glossary

The Arm Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the Arm Glossary for more information: developer.arm.com/glossary.

Typographic conventions

Arm documentation uses typographical conventions to convey specific meaning.

Convention	Use
italic	Citations.
bold	Interface elements, such as menu names. Terms in descriptive lists, where appropriate.
monospace	Text that you can enter at the keyboard, such as commands, file and program names, and source code.
monospace <u>underline</u>	A permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
<and>	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: <div>MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2></div>
SMALL CAPITALS	Terms that have specific technical meanings as defined in the Arm® Glossary. For example, IMPLEMENTATION DEFINED , IMPLEMENTATION SPECIFIC , UNKNOWN , and UNPREDICTABLE .



We recommend the following. If you do not follow these recommendations your system might not work.



Your system requires the following. If you do not follow these requirements your system will not work.



You are at risk of causing permanent damage to your system or your equipment, or of harming yourself.

**Note**

This information is important and needs your attention.

**Tip**

This information might help you perform a task in an easier, better, or faster way.

**Remember**

This information reminds you of something important relating to the current content.

Useful resources

This document contains information that is specific to this product. See the following resources for other useful information.

Arm documents are available on developer.arm.com/documentation.

Confidential documents are only available to licensees, when logged in. Each document link in the following tables provides direct access to the online version of the document.

Table 1: Arm publications

Document name	Document ID	Licensee only
Fast Models Reference Guide	100964	Non-Confidential
Fast Models User Guide	100965	Non-Confidential
User-based Licensing User Guide	102516	Non-Confidential