NXP Semiconductors
Application Note

Document Number: AN5172
Rev. 11.3.2, 07/2018

Using Trace Compass with CodeWarrior for ARMv8

Contents

1 Introduction

This document describes how to use the Trace Compass feature from CodeWarrior for ARMv8 release. **Trace Compass** is an open source toolkit that integrates open source trace frameworks/toolkits. It is based on Eclipse plugins (views) and binaries, and shared libraries. The trace compass tool allows you view and analyze trace data in various forms, like views, graphs, and metrics. The tool helps in extracting the useful information from traces in a much simpler and user-friendly manner.

NOTE

The Eclipse plugin is the architecture and Operating System agnostic based on a Java implementation. The C binaries are Operating System dependent. All of them can run only on a Linux-based system.

This application note includes the following sections:

- Setup requirements
- Collecting trace data using LTTng command-line tools
- Importing trace data

2 Setup requirements

For the SDK based on DASH, before proceeding the trace collection, you have to enable LTTng within kernel and associated modules.

Ensure that LTTng layer is available and is compiled successfully, using the following commands:

1. Set CONFIG_BUILD_LTTNG to "y", replacing default "n" in

   ```
   flexbuild/configs/build_lsdk.cfg
   ```
2. Build linux kernel

   ```
   flex-builder -c linux
   ```
3. Build LTTng modules

   ```
   flex-builder -c lttng-modules
   ```
4. Prepare boot partition

   ```
   flex-builder -i mkbootpartition
   ```
5. Prepare Ubuntu rootfs

   ```
   flex-builder -i mkrfs -a arm64
   ```
6. Merge modules into rootfs and obtain its compressed .tgz version

   ```
   flex-builder -i merge-component
   flex-builder -i compressrfs
   ```
7. Prepare board with the boot partition and rootfs images. For steps to obtain boot partition and rootfs images, refer LSDK documentation or use the flex-installer utility Help menu.

3 Collecting trace data using LTTng command-line tools

After the built image is boot up, you can start a trace session using LTTng toolkit. Perform the following steps to start the trace collection:

1. Create a trace session using the `create` command. By default, the traces are written in the root directory.
2. Filter the trace session by enabling only certain events/functions.
3. Start the trace session after all the settings are applied, using the `start` command. The `stop` command stops the tracing.
4. View the generated trace data using the `view` command, which is by default calling *babeltrace*.
5. Use Babeltrace for further processing of the trace log. You need to specify the trace path.
6. Destroy the trace session, using the `destroy` command followed by the name of the session.

The following list of commands shows an example of how to collect trace data using LTTng commands.

```
root@Ubuntu:~# lttng create mySession
Session mySession created.
Traces will be written in /root/lttng-traces/mySession-20171124-231444
root@Ubuntu:~# lttng enable-event sched_switch -k
Kernel event sched_switch created in channel channel0
root@Ubuntu:~# lttng start
Tracing started for session mySession
root@Ubuntu:~# lttng stop
Waiting for data availability....
Tracing stopped for session mySession
root@Ubuntu:~# lttng view
[23:15:20.516894700] (+?.??????????) Ubuntu.ls1088ardb sched_switch: { cpu_id = 2 },
{ prev_comm = "swapper/2", prev_tid = 0, prev_prio = 20, prev_state = 0, next_comm = "lttng-
consumerd", next_tid = 3812, next_prio = 20 }
[23:15:20.518855700] (+0.001961000) Ubuntu.ls1088ardb sched_switch: { cpu_id = 5 },
{ prev_comm = "swapper/5", prev_tid = 0, prev_prio = 20, prev_state = 0, next_comm =
"rcu_preempt", next_tid = 8, next_prio = 20 }
[23:15:20.518868540] (+0.000012840) Ubuntu.ls1088ardb sched_switch: { cpu_id = 0 },
{ prev_comm = "swapper/0", prev_tid = 0, prev_prio = 20, prev_state = 0, next_comm =
```

```
"rcu_sched", next_tid = 9, next_prio = 20 }
[23:15:20.518895420] (+0.000026880) Ubuntu.ls1088ardb sched_switch: { cpu_id = 5 },
{ prev_comm = "rcu_preempt", prev_tid = 8, prev_prio = 20, prev_state = 1026, next_comm =
"swapper/5", next_tid = 0, next_prio = 20 }
[23:15:20.518895980] (+0.000000560) Ubuntu.ls1088ardb sched_switch: { cpu_id = 0 },
{ prev_comm = "rcu_sched", prev_tid = 9, prev_prio = 20, prev_state = 1026, next_comm =
"kworker/0:1", next_tid = 111, next_prio = 20 }
[23:15:20.518923380] (+0.000027400) Ubuntu.ls1088ardb sched_switch: { cpu_id = 0 },
{ prev_comm = "kworker/0:1", prev_tid = 111, prev_prio = 20, prev_state = 1026, next_comm =
"swapper/0", next_tid = 0, next_prio = 20 }
[23:15:20.522714620] (+0.003791240) Ubuntu.ls1088ardb sched_switch: { cpu_id = 0 },
{ prev_comm = "swapper/0", prev_tid = 0, prev_prio = 20, prev_state = 0, next_comm = "mmcqd/
0", next_tid = 1582, next_prio = 20 }
[23:15:20.522731620] (+0.000017000) Ubuntu.ls1088ardb sched_switch: { cpu_id = 2 },
{ prev_comm = "lttng-consumerd", prev_tid = 3812, prev_prio = 20, prev_state = 2, next_comm
= "swapper/2", next_tid = 0, next_prio = 20 }
[23:15:20.522769420] (+0.000037800) Ubuntu.ls1088ardb sched_switch: { cpu_id = 0 },
{ prev_comm = "mmcqd/0", prev_tid = 1582, prev_prio = 20, prev_state = 1, next_comm =
"swapper/0", next_tid = 0, next_prio = 20 }
...
root@Ubuntu:~# lttng destroy mySession
Session mySession destroyed
root@Ubuntu:~#
```

4 Importing trace data

After collecting the trace data on the board using LTTng command-line utilities.

NOTE

Before importing and viewing the trace data, open the **Project Explorer** view and switch to **LTTng Kernel** perspective.

Perform the following steps to import a tracing project in the CodeWarrior:

1. Select **File > New > Project**.

   The **New Project** wizard appears.

2. Expand the **Tracing** folder and select **Tracing Project**.
3. Click **Next**.
   The **Tracing Project** page appears.

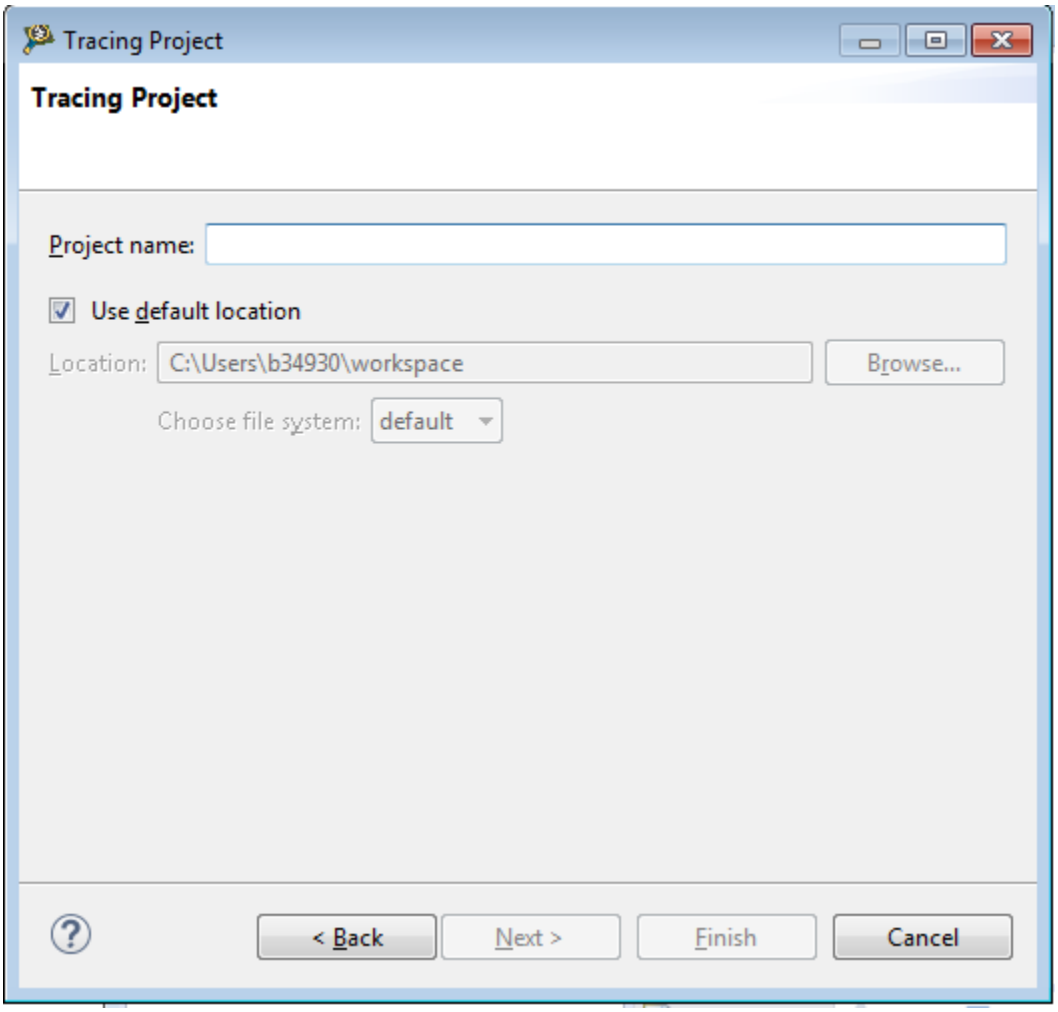

Figure 1. Tracing Project wizard

4. Specify the **Project name** and click **Finish**.

   The new project is added in the Projects view area.

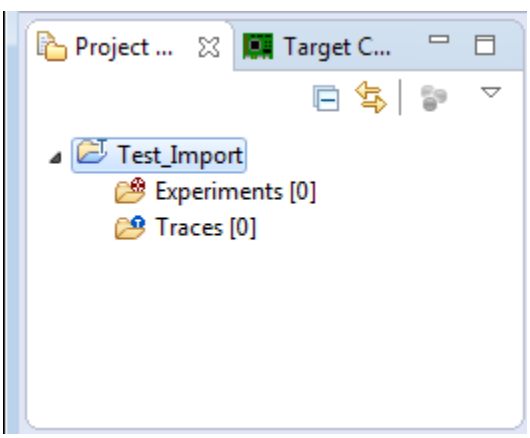

Figure 2. Project View

5. Expand the node of the selected project. It contains the following folders:

- **Experiments** folder contains a set of traces that needs to be analyzed. These traces are selected from *Traces* folder. It displays several trace files by overlapping the data in the user interface.
- **Traces** folder contains all the imported trace files.

6. Right-click the **Traces** folder and select **Import** from the menu.
   The **Trace Import** wizard appears.

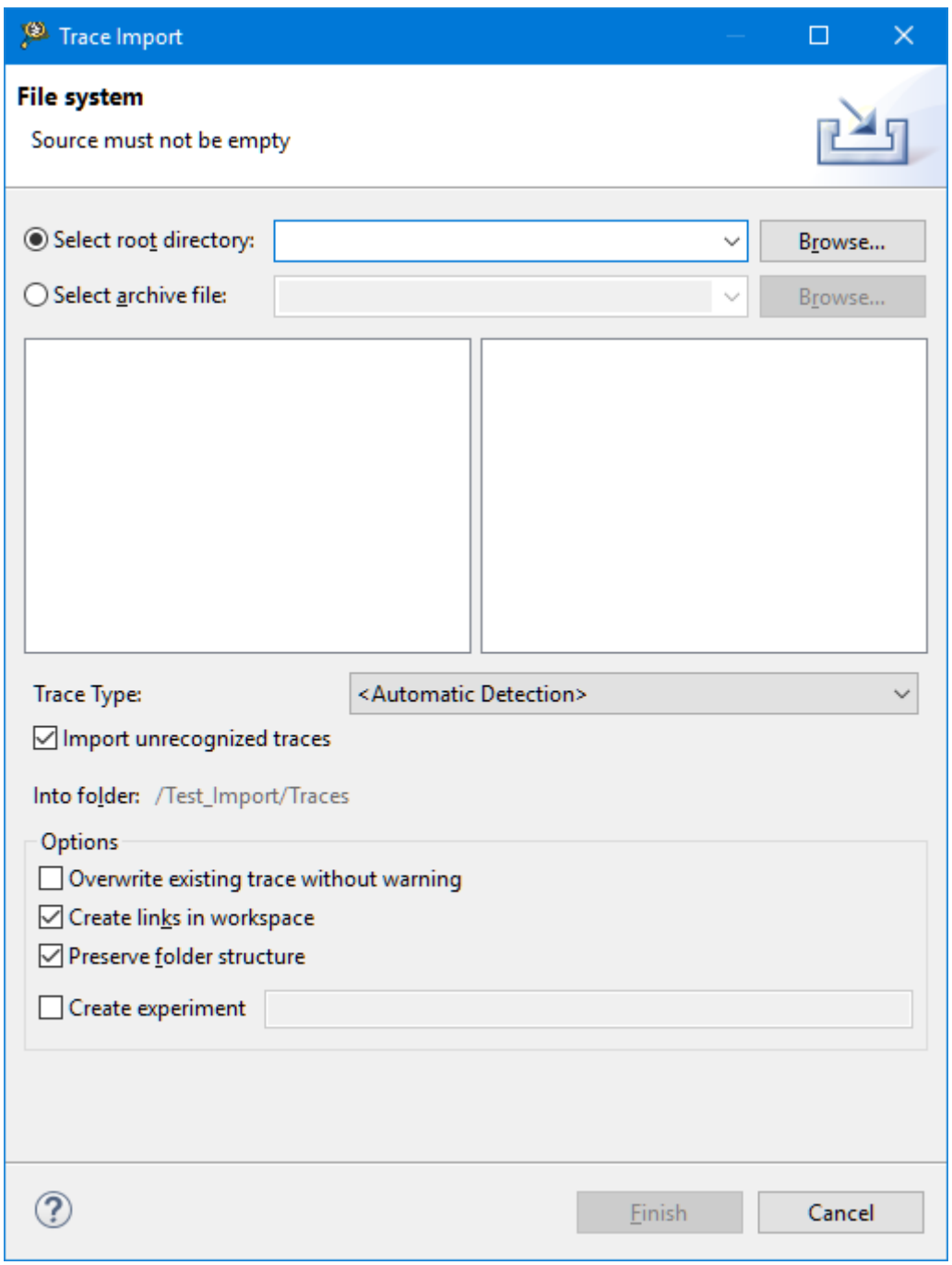

Figure 3. Trace Import wizard

7. Browse the location of the **Source directory** that contains the tracing session folder.
8. Expand the node of the tracing session folder and select the trace folder.
9. Select the type of trace using **Trace type** field and click **Finish**.

   The folder that contains the trace data gets added under the **Traces** folder of the Projects view.

Double-click the trace file to access the trace data. You can now use all the advantages and features offered by Trace Compass toolkit.

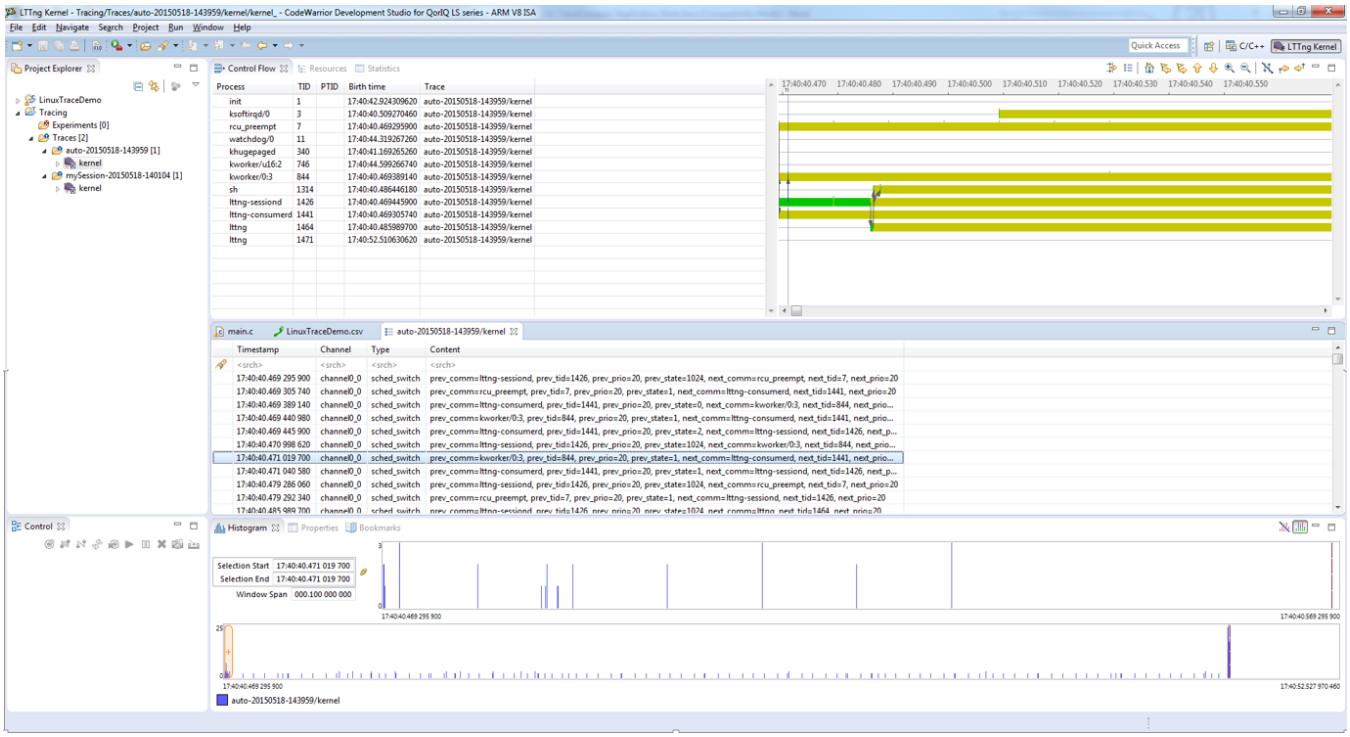

Figure 4. LTTng Perspective

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number AN5172
Revision 11.3.2, 07/2018