MCUXpresso IDE Zephyr RTOS Debug Guide

Rev. 11.6.0 — 13 July, 2022 User guide

13 July, 2022

1. Introduction

Zephyr is a scalable real-time operating system (RTOS) supporting multiple hardware architectures, optimized for resource constrained devices and built with security in mind. It is based on a small-footprint kernel designed for use on resource-constrained systems, supports ARM architecture (Cortex-A, Cortex-R, Cortex-M) and a large number of NXP boards. For the complete list of supported hardware please consult the official Zephyr Project documentation.

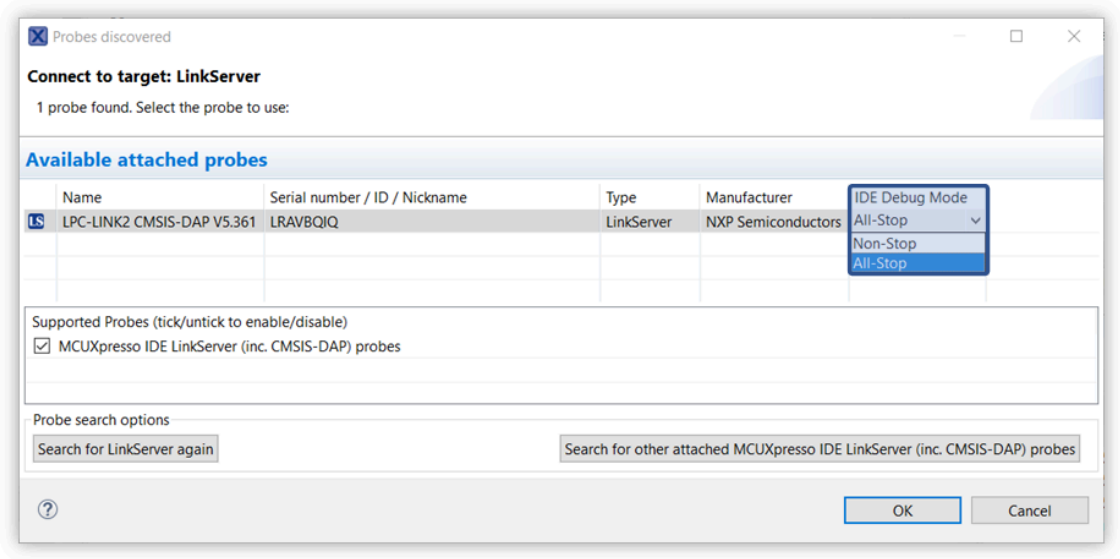
This guide is intended to highlight debugging capabilities of the MCUXpresso IDE when dealing with a Zephyr RTOS based application. It does not provide any information on Zephyr RTOS itself or on developing applications that use Zephyr RTOS.

For more information on Zephyr RTOS please visit www.zephyrproject.org/

2. LinkServer Zephyr RTOS Thread Aware Debugging

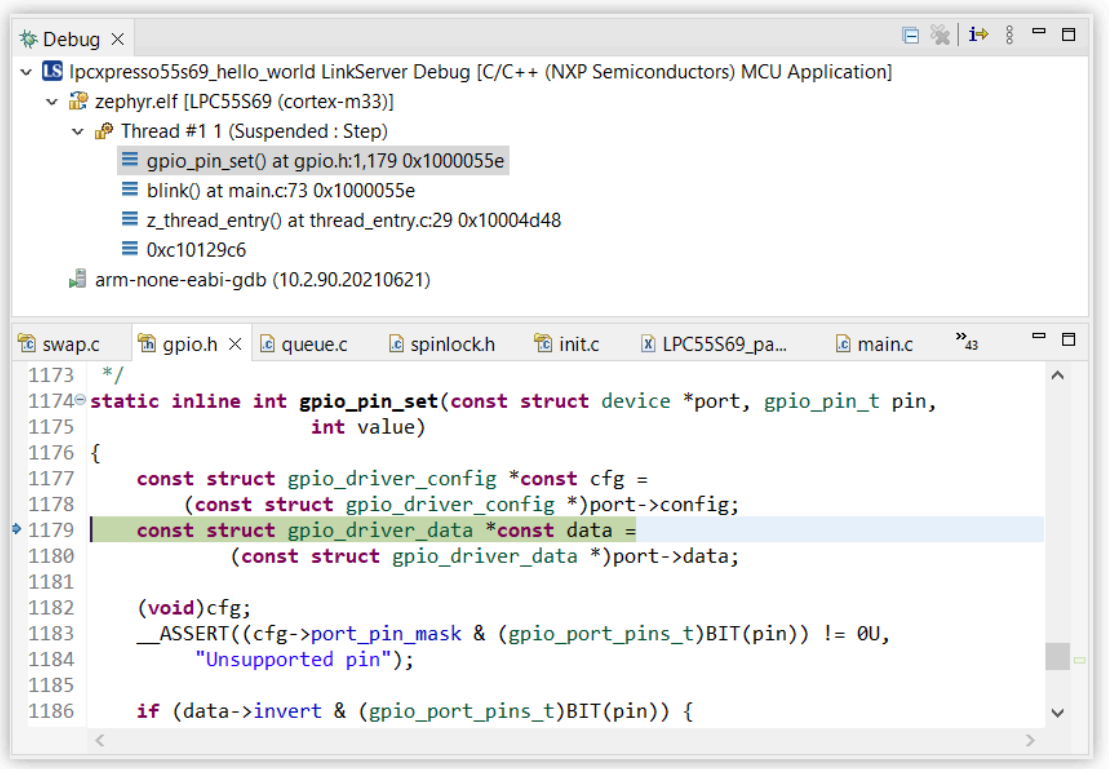
When debugging via LinkServer debug probes, the MCUXpresso IDE debugger can provide Zephyr RTOS thread aware debug if :

1. Debugging is carried out in *All-Stop* mode (rather than the default *Non-Stop* mode). This selection is made when first making a debug connection for a particular project (or after deleting an existing launch configuration). For more details, please see the MCUXpresso IDE User Guide.
2. Zephyr project is configured with `CONFIG_DEBUG_THREAD_INFO=y`. If `CONFIG_DEBUG_THREAD_INFO` is undefined or disabled, debugger will be unable to extract required information from the Zephyr RTOS kernel, thus not capable of offering GDB thread awareness functionality, nor offering insight of Threads details using the dedicated Zephyr RTOS TAD view.

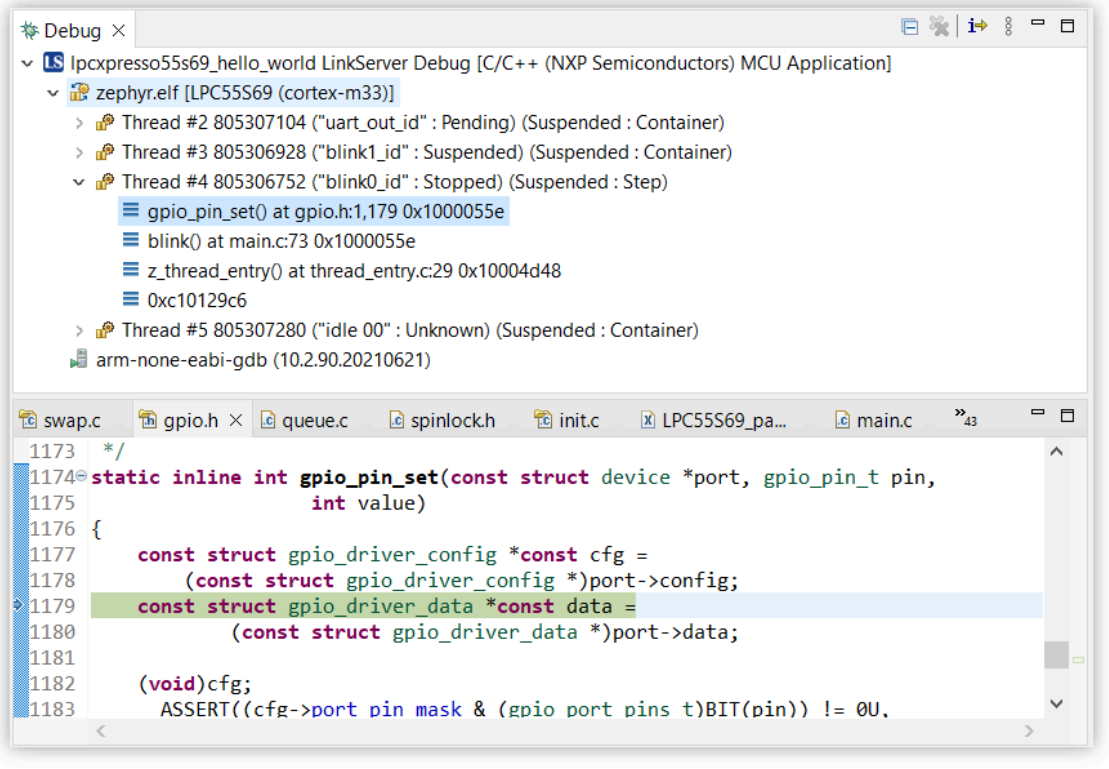


Note 1: Initial development of the GDB thread awareness feature was done using Zephyr RTOS version 2.6.0, and implicitly, the definitions of all the required data structures from the associated code base were used as references.

Note 2: If *Non-Stop* debug mode is used, only the current thread will be seen in the Debug View, as shown in the below screenshot:



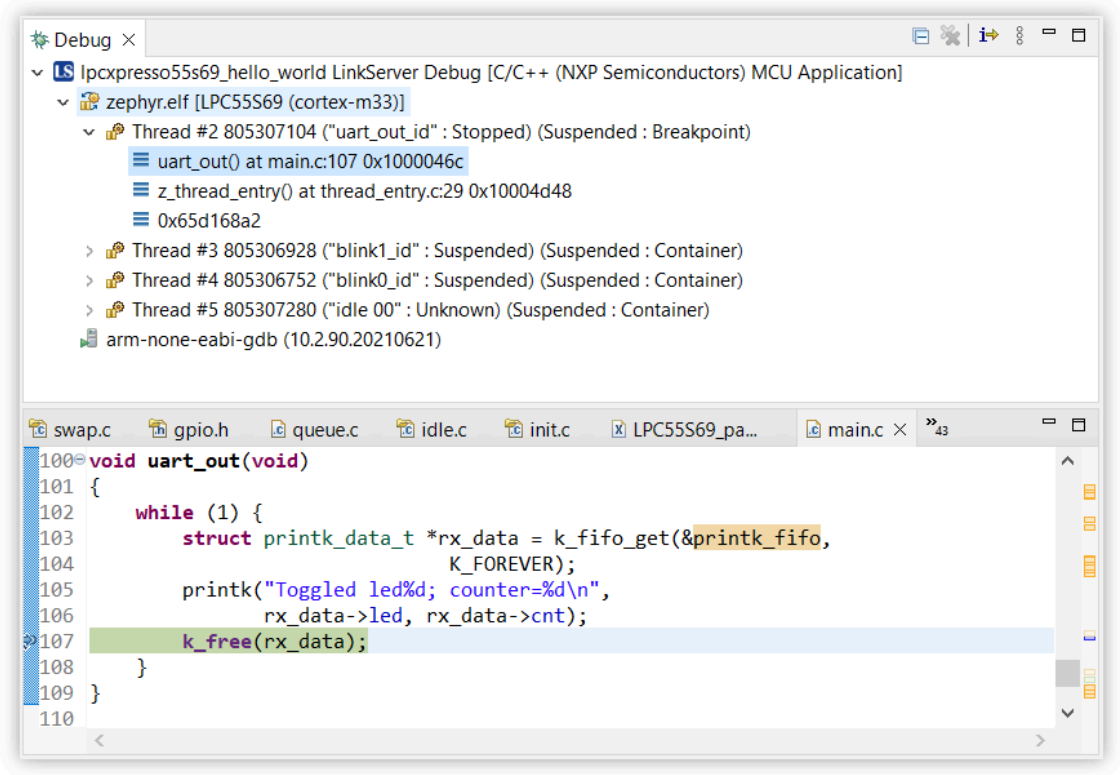
However, once all the necessary preconditions are met, the Debug View will display each thread individually, as shown in the next screenshot:



2.1 Behavior when thread aware debugging

MCUXpresso IDE LinkServer Zephyr RTOS thread aware debugging is available once the Zephyr kernel has started. Debug works in stop mode. In other words, if execution of a user task is halted either through a user action (halt) or a debug event (breakpoint, watchpoint, fault, etc.), the stopped thread is current and no application thread executes in the background. The register context for any thread is available in the Registers view. For suspended or blocked threads, the register context is the context in effect when the thread was swapped out, regardless of which thread stack level is examined within the traceback window.

In the below example, the MCU is halted in Thread #2, but a backtrace for the other threads is also available:

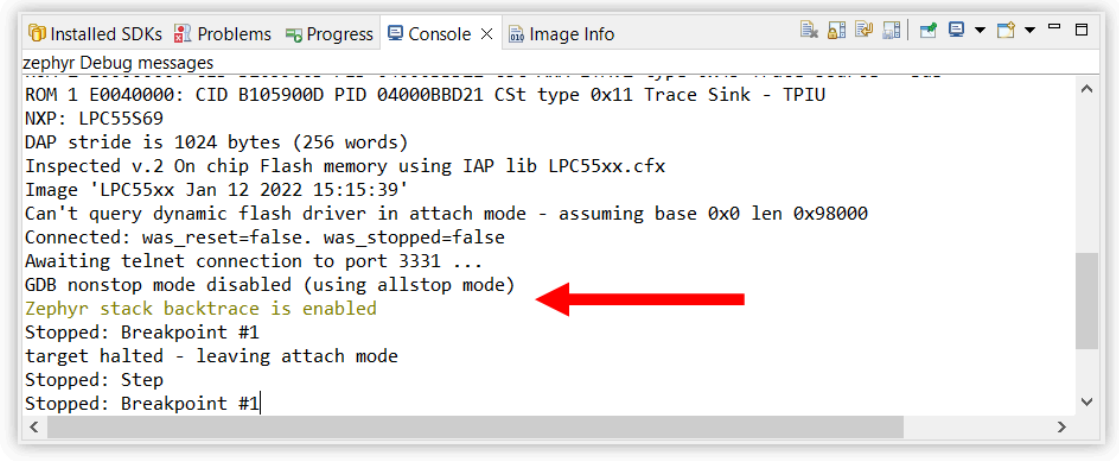


2.2 Debugger Messages

LinkServer Zephyr RTOS Thread Aware Debugging requires that the debug session is carried out in *All-Stop* mode. If this precondition is met, then when you start your debug session, confirmation that LinkServer Zephyr RTOS Thread Aware Debugging is active is recorded in the “Debug Messages” log inside the IDE’s Console view:

```
...
GDB nonstop mode disabled (using allstop mode)
Zephyr stack backtrace is enabled
...
```

as shown in the screenshot below:



If the image is debugged in *Non-Stop* mode, then the log will indicate that thread aware debugging will not be available:

```
...
GDB nonstop mode enabled
Zephyr stack backtrace is disabled in Non-stop mode (use All-stop)
...
```

2.3 Switching between all-stop and non-stop debug modes

When debugging a project for the first time using a LinkServer debug connection (or when you have deleted any existing launch configuration files), you can select whether to debug in *Non-Stop* or *All-Stop* mode (so that you can choose whether or not you wish to use Zephyr RTOS thread awareness).

However, you can also easily modify an existing launch configuration file to switch between *All-Stop* and *Non-Stop* as follows.

- Open the project up in the Project Explorer view and double click on the appropriate launch configuration file (typically “projname LinkServer Debug.launch”)
- Switch to the GDB Debugger tab
- Tick / Untick the " *Non-Stop* mode" option, as required.
- Click on Apply to save, then Continue

Any further debug sessions of your project will now use the newly selected debug mode.

3. Zephyr RTOS Thread Aware Debug Views

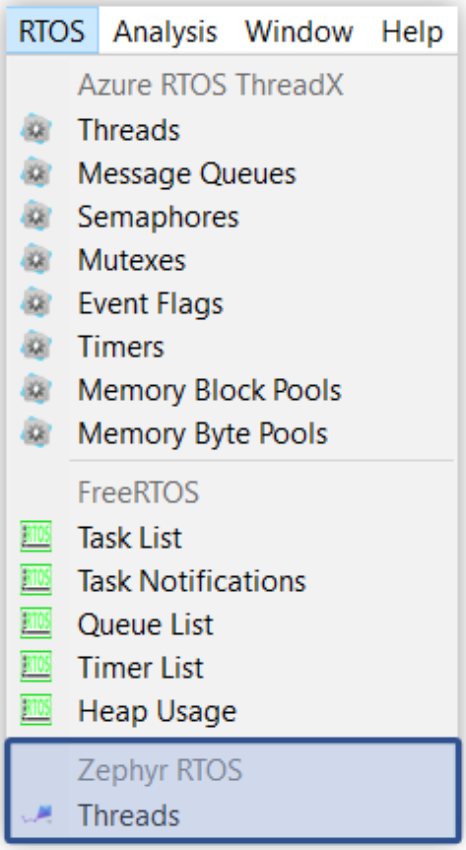
MCUXpresso IDE includes additional Views to further simplify Zephyr RTOS application debugging, known collectively as the *Zephyr RTOS TAD* (Thread Aware Debugger for GDB):

- **Threads** : shows the list of threads with status information

Note: TAD views are independent of the debug probe being used, as they just use GDB commands to query information from the target.

3.1 Showing the Zephyr RTOS TAD Views

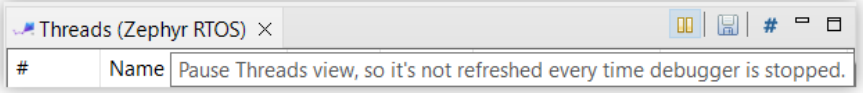
The Zephyr RTOS TAD views can be opened using the common “RTOS” main menu in the MCUXpresso IDE.



The views are “stop mode” views: with the target halted or stopped, the views will query the device under debug and read the necessary information through the debug connection.

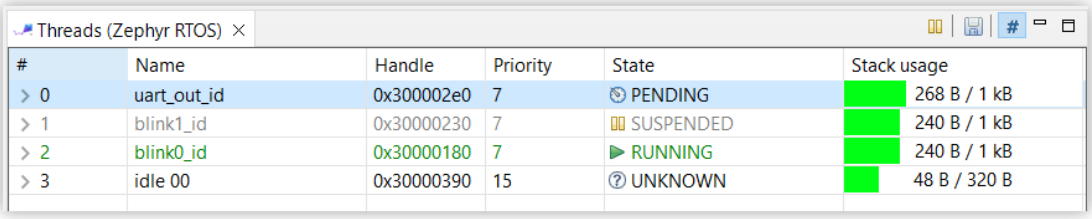
This will also happen during single stepping, so to improve stepping performance it is advisable to:

1. Only have the needed views in the foreground/visible, or close the views if they are not used.
2. Make use of the Pause View feature, allowing you to single step without the views constantly reloading data.



3.2 Threads View

This View shows the threads in a table:



- # - Thread Control Block index.
- **Name** - Name of the thread.
- **Handle** - Address of the thread handle.
- **Priority** - Priority of the thread.
- **State** - Current task state (e.g. running, suspended, queued, etc.). See `kernel_structs.h` for all the possible states.
- **Stack Usage** - Graphical view of current stack usage, with current allocation and stack size available to the thread.

Unfolding a thread line item shows the following items:

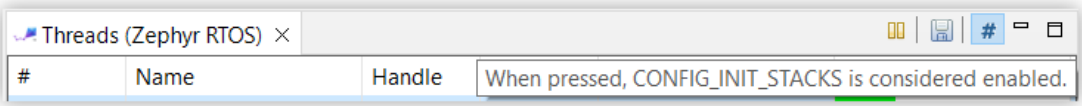
- **Entry** - Entry function for current thread.
- **Stack start** - Stack start address.
- **Stack end** - Stack end address.
- **Stack size** - Size of stack.
- **Saved stack ptr** - Thread's saved stack pointer during the last context switch.
- **Stack high watermark** - Highest address used by stack.
- **User options** - User facing 'thread options'. Values defined in `kernel.h`.
- **Entry param 1** - The first entry point parameter.
- **Entry param 2** - The second entry point parameter.
- **Entry param 3** - The third entry point parameter.

Note 1: Generic stack information depends on the enablement of `CONFIG_THREAD_STACK_INFO`.

Note 2: Stack usage can be computed using two different methods:

- Using SP value saved in the task-specific data structure during context switch.
- Using a high water mark that can only be identified by inspecting the content of the stack. This method depends on the enablement of `CONFIG_INIT_STACKS` in the Zephyr RTOS code. If this should be the preferred method, then make sure the project configuration contains `CONFIG_INIT_STACKS=y` at build time. Byte pattern used for stack initialization is considered `0xAA`.

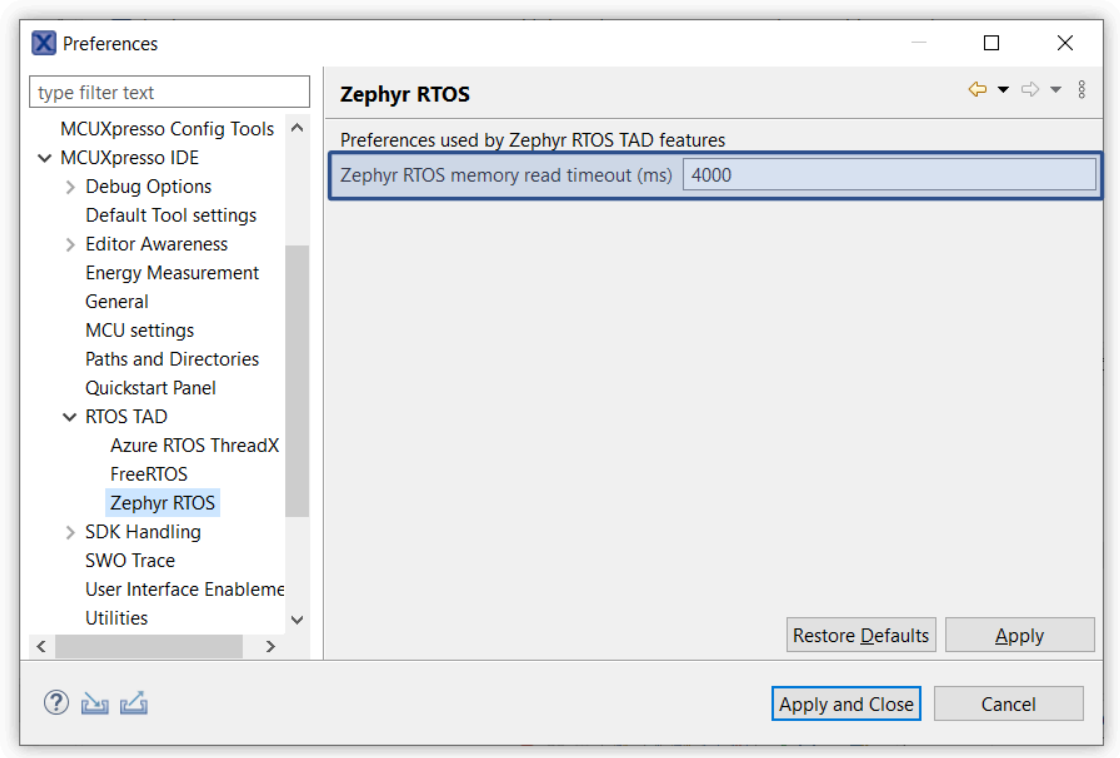
The view also allows manual switching between the two stack usage computation methods, by using the dedicated button.



3.3 Timeouts

When using slow debug probes, it is possible that timeouts will be reported within the IDE.

The timeout period can be extended if this occurs using the Workspace preference as shown below:



4. Thread Aware Debugging with Other Debug Probes

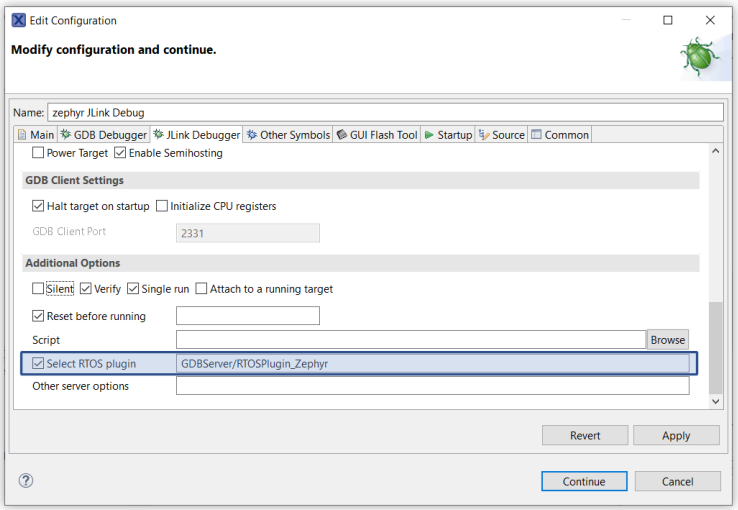
4.1 PEmicro Probes

Zephyr RTOS thread aware debugging with PEmicro debug probes is currently unsupported.

4.2 SEGGER J-Link Probes

Zephyr RTOS thread aware debugging for SEGGER J-Link debug probes is disabled by default.

To turn it on, enable the “Select RTOS plugin” option for “GDBServer/RTOSPlugin_Zephyr” in the J-Link Launch Configuration for your project:



Note: J-Link’s GDB thread awareness feature relies on the prerequisites and on the same data structures as described in LinkServer Zephyr RTOS Thread Aware Debugging [2] section.

5. Legal information

How To Reach Us:

Home Page: https://www.nxp.com

Web Support: https://www.nxp.com/support