AN1418: Running Zigbee and OpenThread Concurrently on a System-on-Chip

This document describes how to run a combination of Zigbee and OpenThread networking stacks and the Zigbee application layer on a System-on-Chip (SoC). One of the main functions of a Concurrent Multiprotocol (CMP) device is to act as a bridge between Zigbee and OpenThread networks.

Note that, depending on the chip, memory size restrictions may prevent running Matter on SoC devices.

KEY POINTS

- Important features of the sample application
- Making a Zigbee-OpenThread CMP application from a Z3Light

1 Introduction

This document describes a Concurrent Multiprotocol (CMP) application that runs Zigbee and OpenThread stacks on a single EFR32 radio. The primary use for such an application is to allow Zigbee line-powered devices to also be part of an OpenThread network simultaneously and therefore serve as a bridge between the two networks.

2 Concurrent Multiprotocol (CMP) Sample Application (z3-light_ot-ftd)

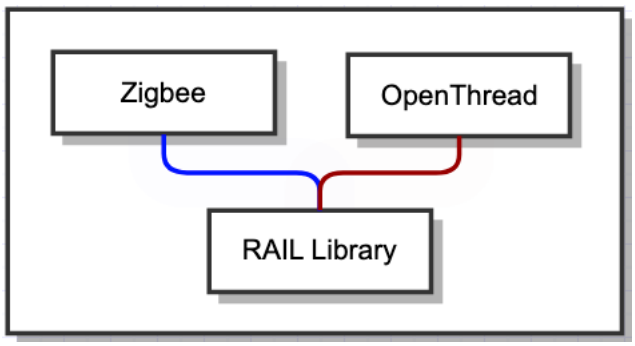


Figure 2-1. Zigbee + OpenThread Concurrent Multiprotocol Application

The CMP sample application consists of a Z3Light, which is a Zigbee router, and an OpenThread FTD (Full Thread Device). Both protocol stacks operate by multiplexing a single EFR32 radio. Both protocols need to use the same radio channel to ensure proper operation.

2.1 RTOS

Within the CMP application, scheduling is managed using a Real Time Operating System (RTOS). Each protocol runs in a dedicated RTOS task. The Zigbee and OpenThread tasks operate at the same priority while the Command Line Interface (CLI) is made available using a CLI RTOS task that operates at a lower priority.

Caution: It is critical to note that Zigbee and OpenThread APIs are not thread-safe. Calling them from different threads can result in unexpected behavior. In addition, any references to EmberMessageBuffer must be contained within the Zigbee task.

2.2 Command Line Interface

This application supports all CLI commands that can be found in the Z3Light sample application. A subset of the OpenThread CLI has been ported to demonstrate form, join and ping operations. This functionality can be extended further, if necessary, by following the example commands in the ot_up_cli.c file from the ot_up_cli component. Note that OpenThread APIs are only invoked from sl_ot_rtos_application_tick since they are not thread-safe.

2.2.1 OpenThread Commissioning

This device can be commissioned on to an OpenThread network out-of-band using CLI commands. Setting the OpenThread network parameters, such as network key and channel, before starting the network allows the CMP device to join a Thread network as a child or router device.

Table 2.1. OpenThread CLI commands

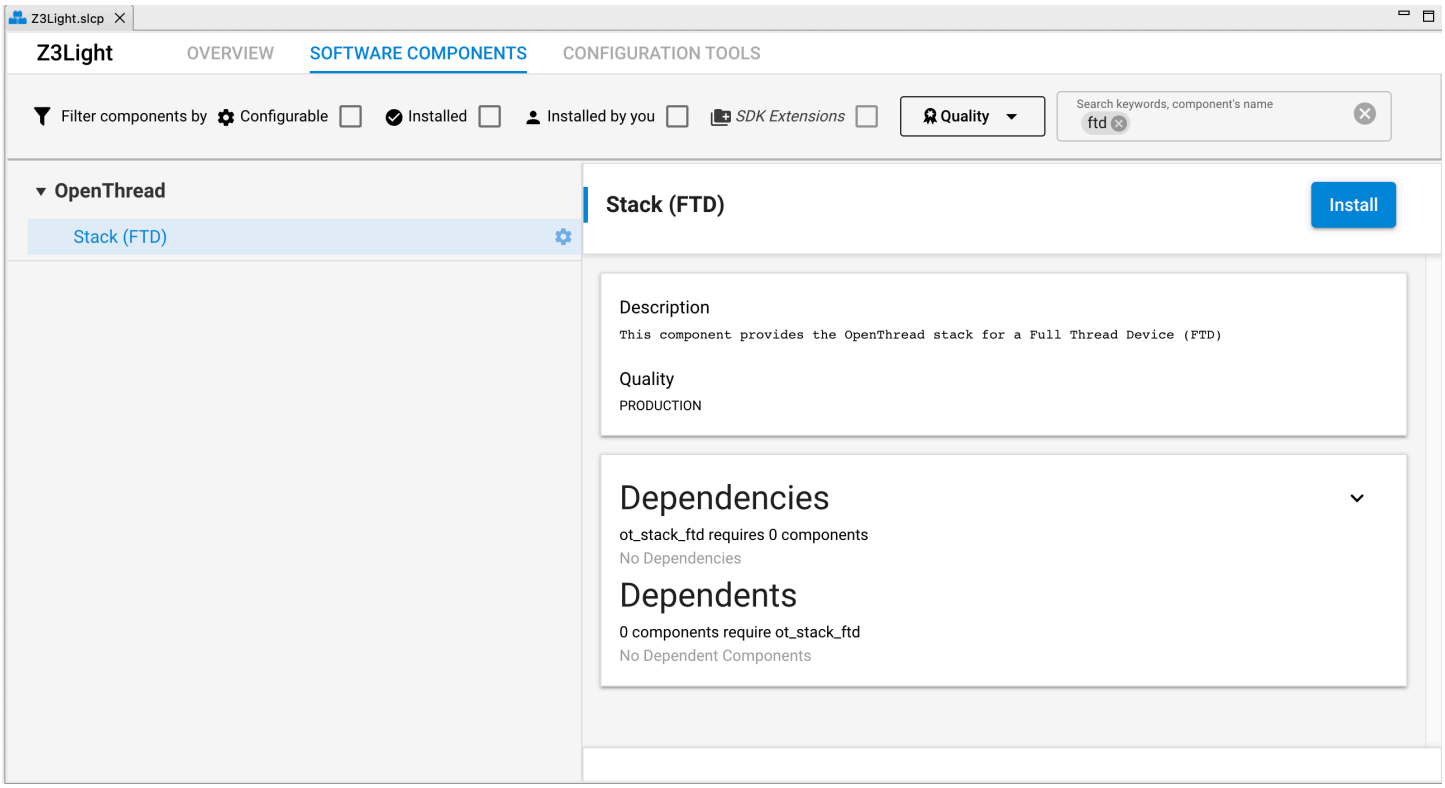
CLI Command	Description
dataset	View OpenThread network configuration.
dataset_new	Creates a new OpenThread dataset.
dataset_commit_active	Commits dataset to NVM.
factory_reset	Removes all NVM OpenThread settings.
dataset_networkkey	Presets the network key on the device to help with joining an existing OpenThread network out-of-band.
dataset_channel	Presets the radio channel used by the OpenThread network. This command can be used to force both Zigbee and OpenThread networks to use the same radio channel.

CLI Command	Description
dataset_pan_id	Presets the PAN ID on the device to help with joining the OpenThread network out-of-band.
dataset_extended_pan_id	Presets the extended PAN ID on the device to help with joining the OpenThread network out-of-band.
ifconfig_up	Enables OpenThread interface.
thread_start	Enables and attaches OpenThread protocol operation.
thread_state	Reads current status: offline, disabled, detached, child, router, or leader.

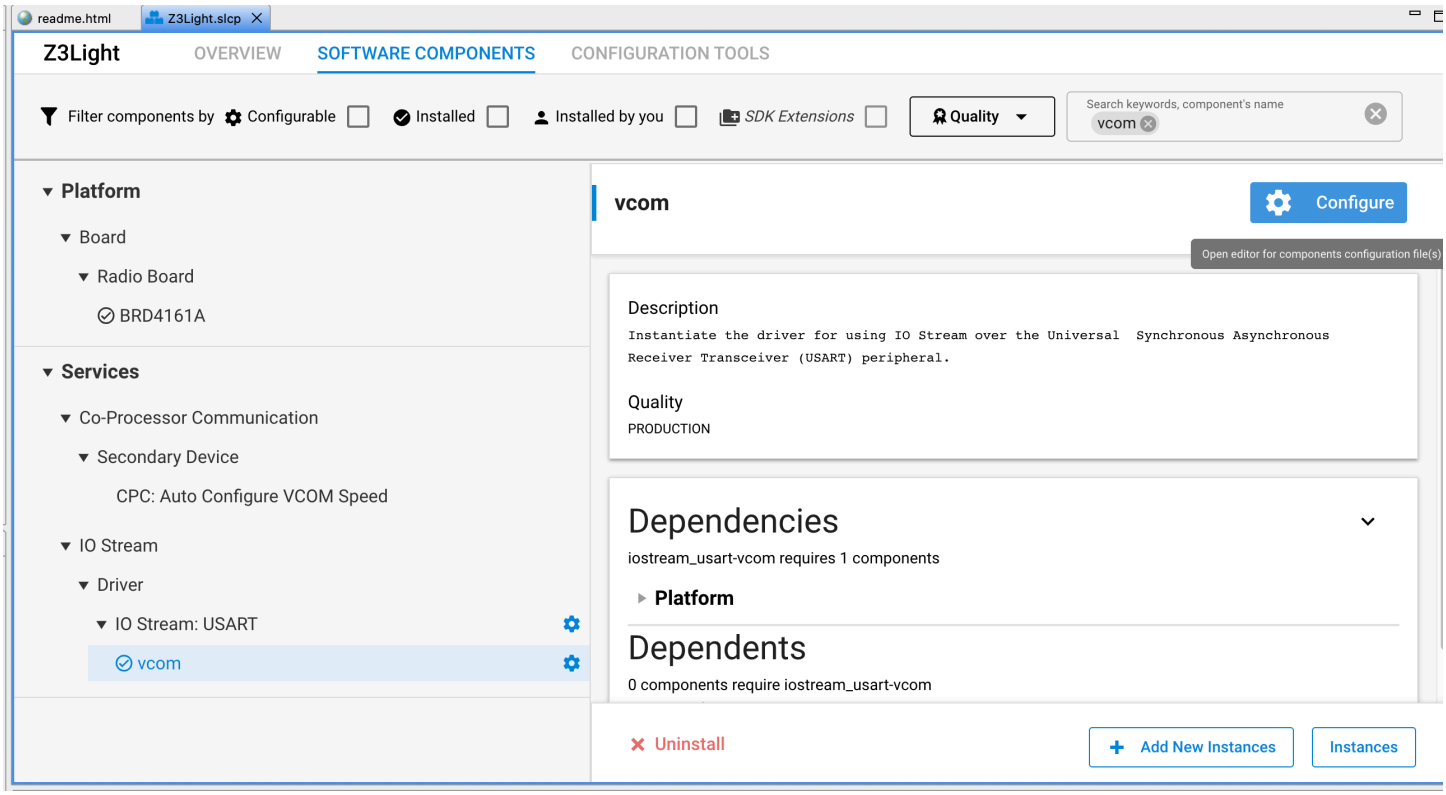
3 Converting a Zigbee Application into a Zigbee-OpenThread CMP Application

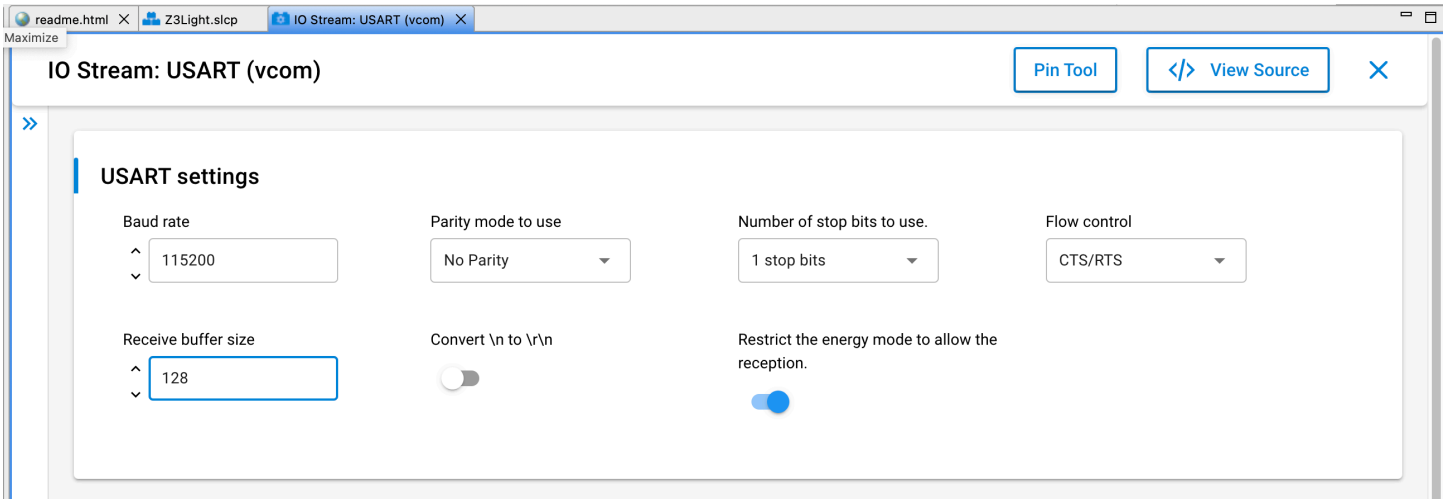
This section describes the steps involved in converting a Z3Light into a Concurrent Multiprotocol application that includes the OpenThread stack.

1. Use the Simplicity Studio “Create New Project” wizard to create a Zigbee – SoC Light project for your board of choice.
2. Open the Software Components tab of the generated project to add the OpenThread > **Stack (FTD)** component. Note that the addition of this component automatically adds a Real Time Operating System (RTOS) to the project.

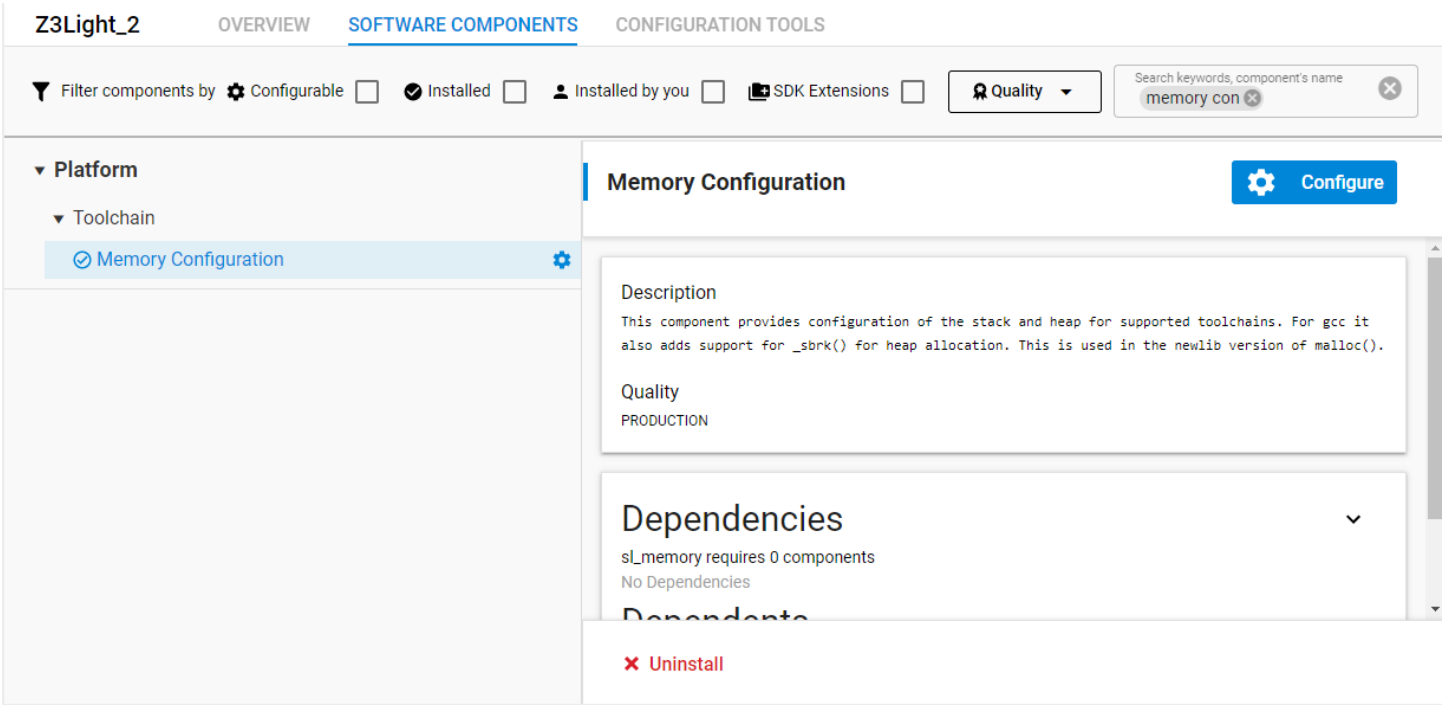


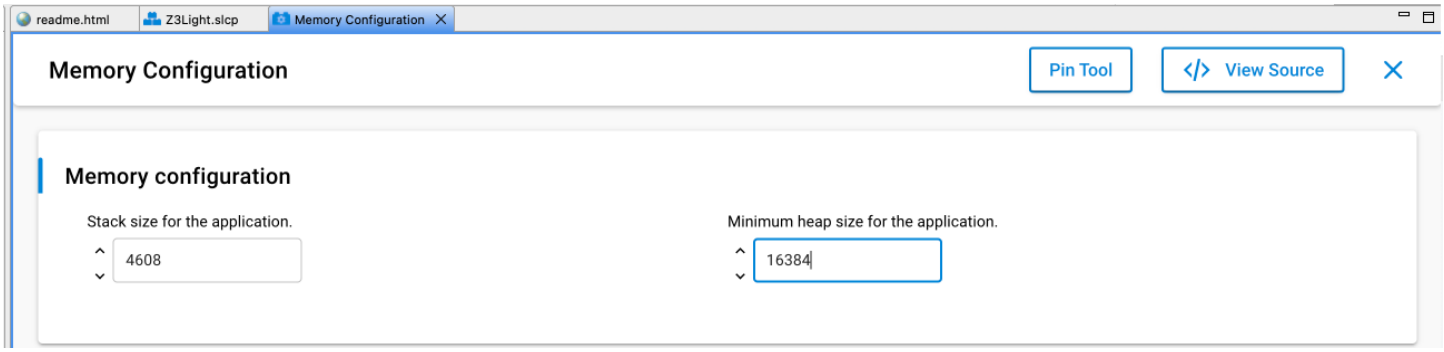
3. Select the IO Stream > Driver > IOS Stream: USART > **vcom** component and configure it by increasing “Receive buffer size” to 128.



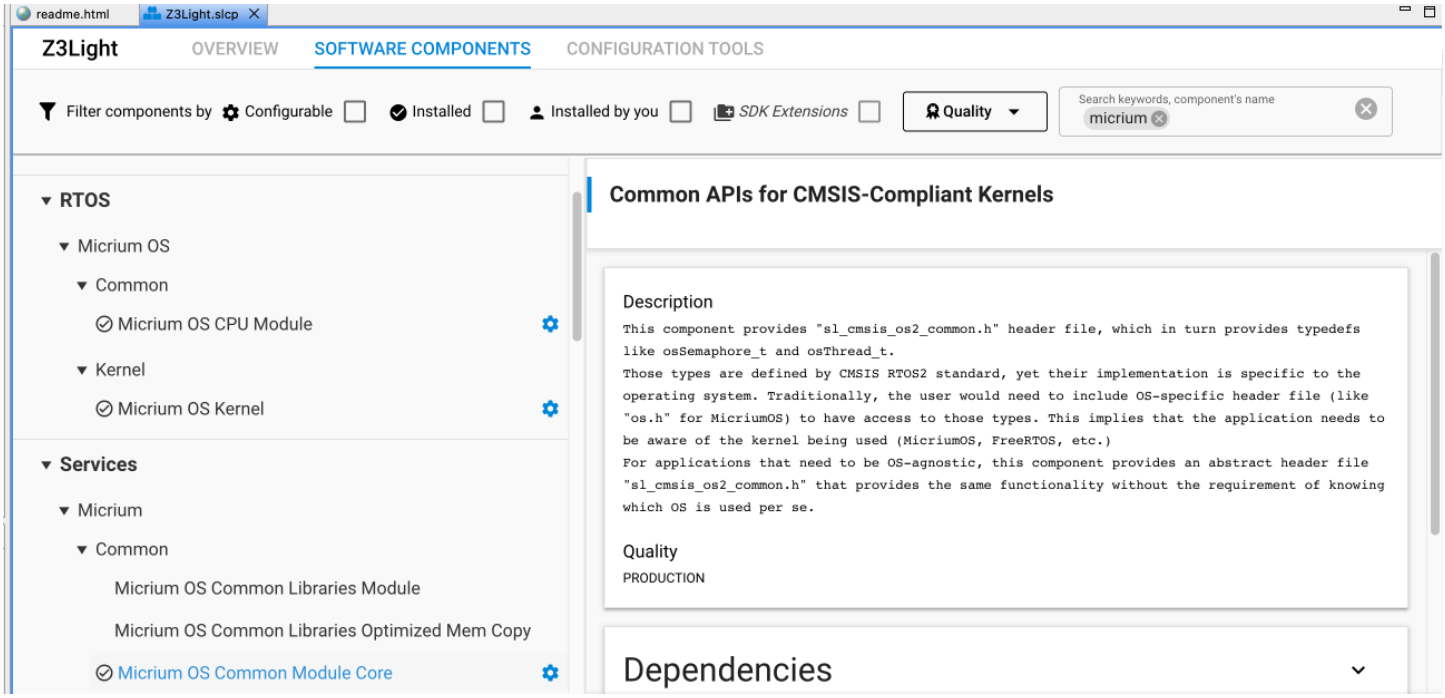


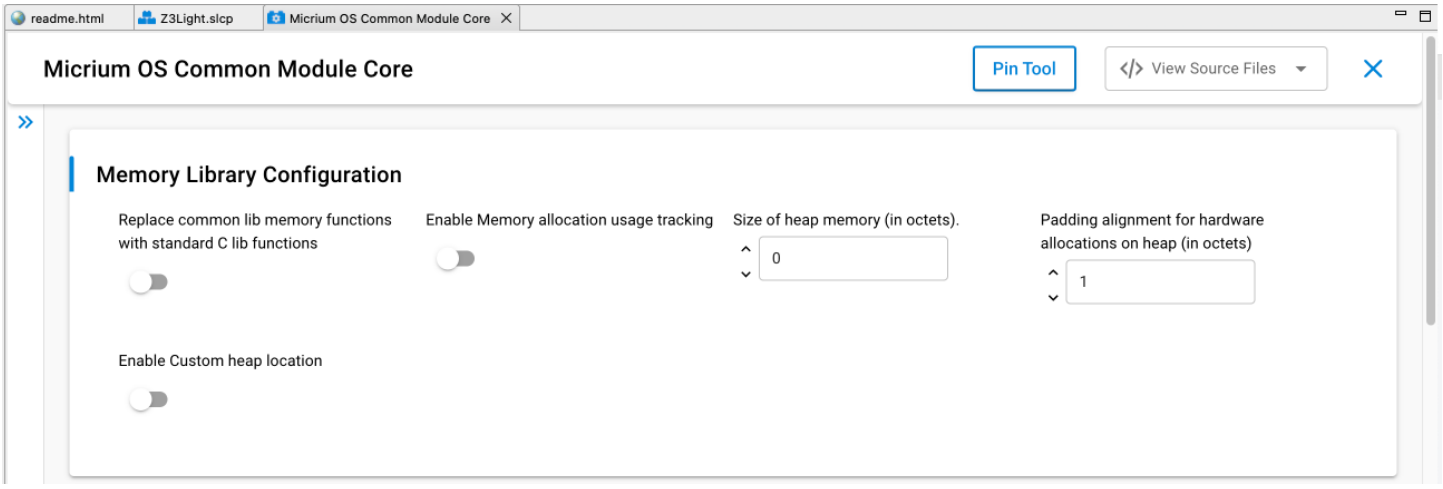
4. Select the Toolchain > **Memory Configuration** component and configure it by increasing stack size to 4608 and heap size to 16384 to account for the addition of OpenThread networking stack.



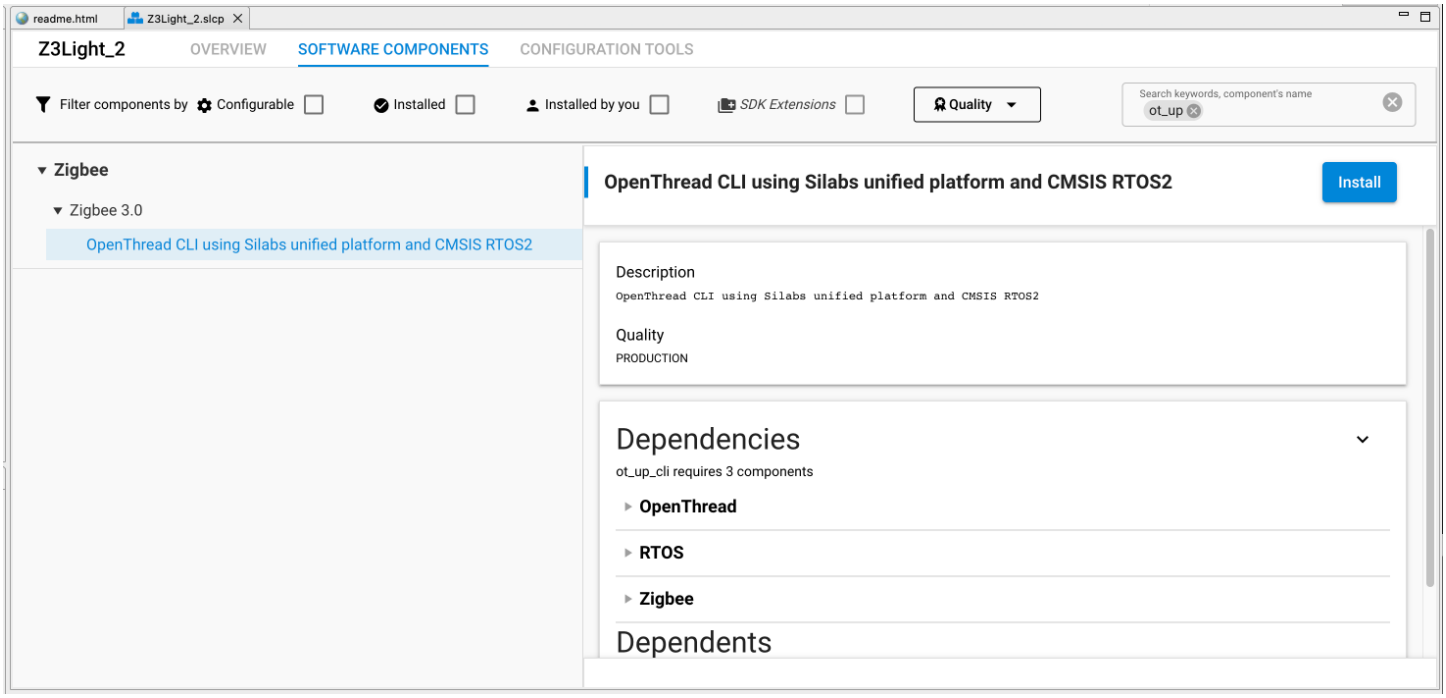


5. Select Micrium > Common > **Micrium OS Common Module Core** component and configure it to decrease “size of heap memory” to 0 to prevent Micrium RTOS from allocating its own heap memory.

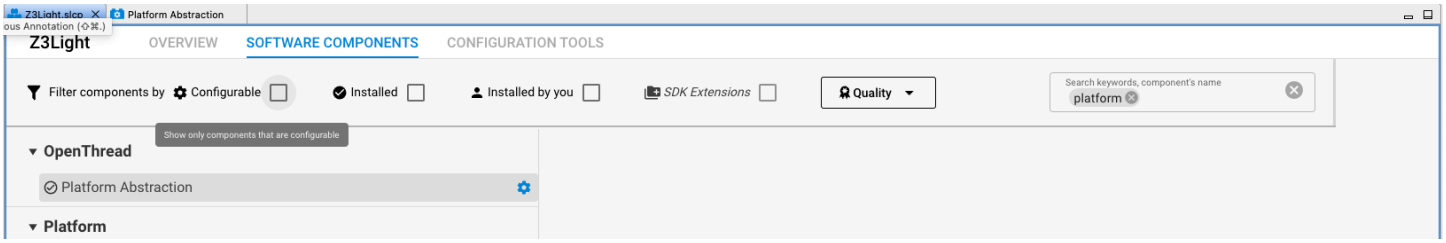


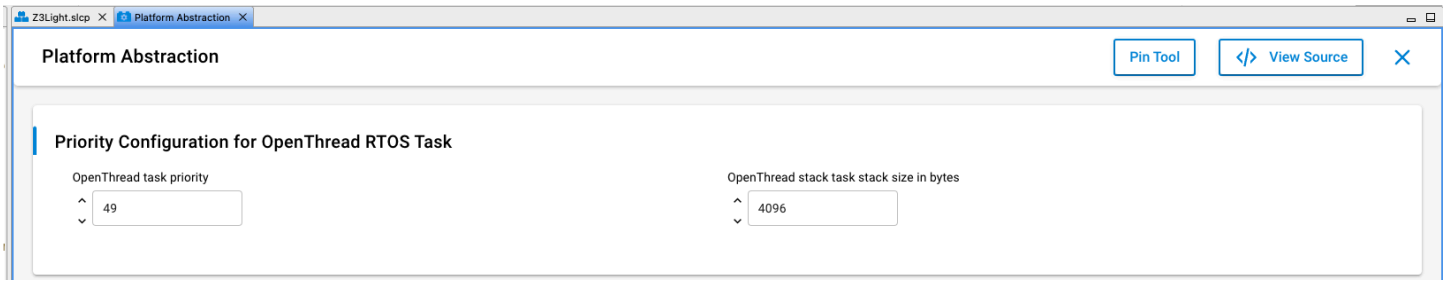


6. Add OpenThread CLI commands by installing the Zigbee > Zigbee 3.0 > **OpenThread CLI using Silabs unified platform** (ot_up_cli) component.



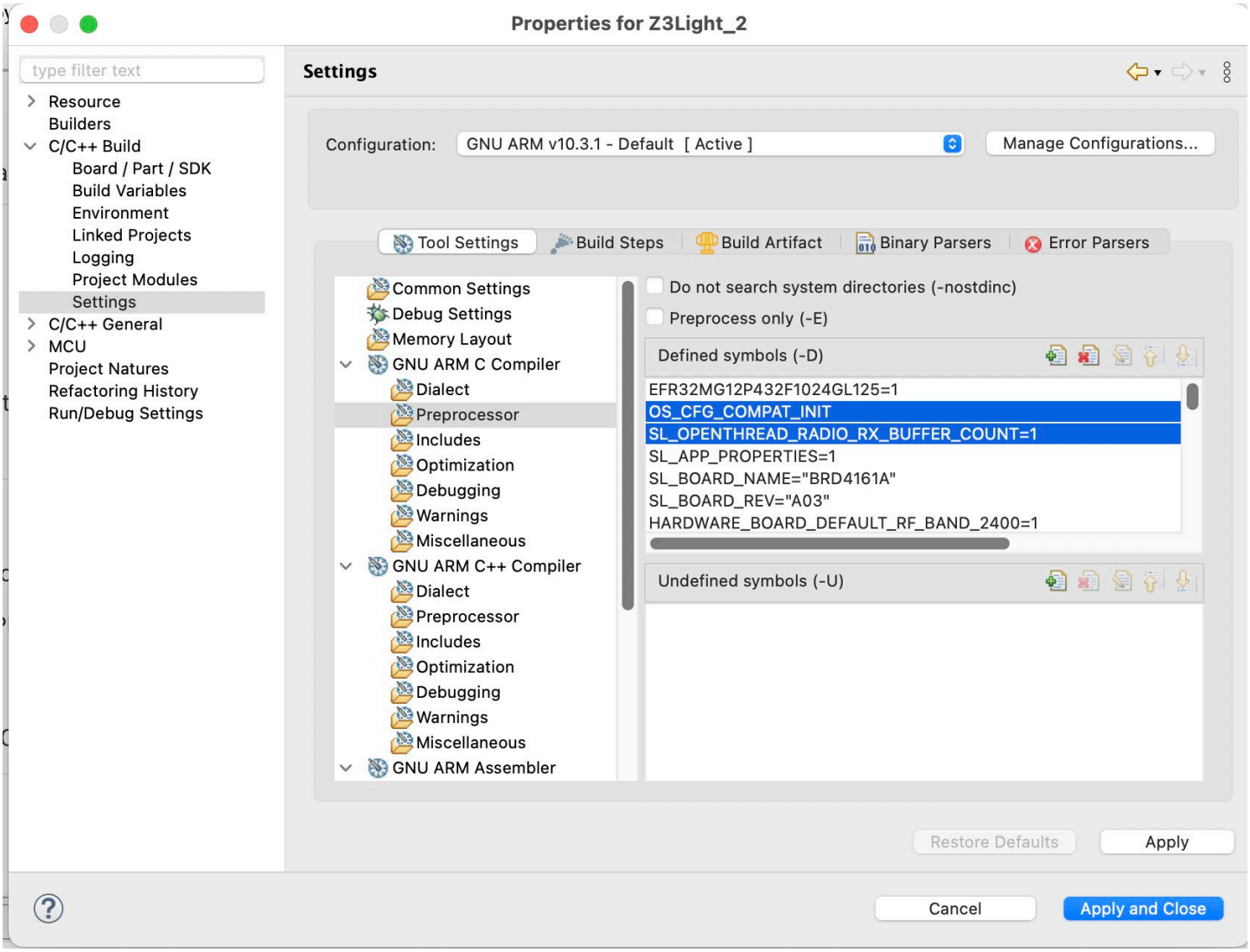
7. Select the OpenThread > **Platform Abstraction** component and configure it by setting priority to 49 to match the Zigbee RTOS task priority.





8. Right click the Z3Light project in Simplicity Studio's Project Explorer view and click Properties. Open C/C++ Build Settings and Under GNU ARM C Compiler, select Preprocessor. Add two preprocessor define symbols:
   - OS_CFG_COMPAT_INIT (Used in conjunction with LIB_MEM_CFG_HEAP_SIZE to allow the application to handle heap allocation)
   - SL_OPENTHREAD_RADIO_RX_BUFFER_COUNT=1 (This is a workaround for an issue where the Zigbee network cannot send beacons when the OpenThread network is up)

   Click Apply and Close to save.



9. Open app.c file in the project folder and add the code below to the beginning of the file to initialize OpenThread. Save file and build the project.

```
#if defined(OPENTHREAD_FTD)
  #include <assert.h>
  #include <openthread-core-config.h>
  #include <openthread/config.h>

  #include <openthread/ncp.h>
  #include <openthread/diag.h>
  #include <openthread/tasklet.h>

  #include "openthread-system.h"
```

```
static otInstance *     sInstance       = NULL;

void sl_ot_create_instance(void)
{
  #if OPENTHREAD_CONFIG_MULTIPLE_INSTANCE_ENABLE
  size_t   otInstanceBufferLength = 0;
  uint8_t *otInstanceBuffer       = NULL;

  // Call to query the buffer size
  (void)otInstanceInit(NULL, &otInstanceBufferLength);

  // Call to allocate the buffer
  otInstanceBuffer = (uint8_t *)malloc(otInstanceBufferLength);
  assert(otInstanceBuffer);

  // Initialize OpenThread with the buffer
  sInstance = otInstanceInit(otInstanceBuffer, &otInstanceBufferLength);
  #else
  sInstance = otInstanceInitSingle();
  #endif
  assert(sInstance);
}

otInstance *otGetInstance(void)
{
  return sInstance;
}
#endif //#if defined(OPENTHREAD_FTD)
```

This application can now form a distributed Zigbee network or join any Zigbee network (centralized or distributed). It can also function as a leader, child, or router on the OpenThread network.

Caution: It is imperative to ensure that both networks operate on the same radio channel.

Any channel changes will need to be done in a controlled fashion. A channel change on one protocol's network can cause the other protocol to stop working until its network is also switched to the same channel. It is important to note that only certain Zigbee device types (trust center) may initiate a channel change on the Zigbee side.

Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

www.silabs.com