AN1256: Using the Silicon Labs RCP with the OpenThread Border Router

A Thread Border Router connects a Thread Network to other IP-based networks, such as Wi-Fi® or Ethernet®. A Thread Network requires a Border Router to connect to other networks. The Border Router provides services for devices within the Thread Network, including routing services for off-network operations, bidirectional connectivity over IPv6 infrastructure links, and service registry to enable DNS-based service discovery. Silicon Labs provides a Border Router Add-On Kit containing a Raspberry Pi device and an example Radio Co-Processor (RCP) application required to build Border Router software.

NOTE: Refer to the OpenThread release notes for the stable version commits of OpenThread (openthread) and OpenThread Border Router (ot-br-posix) repos supported by a Silicon Labs release. OpenThread release notes are installed with the SDK and are also available on docs.silabs.com. This applies to all default containers provided by Silicon Labs for the release, and the copies of these repos included in the release. While we support building using any commit on GitHub (using the ot-efr32 repo), note that the latest public code on GitHub can be unstable.

KEY POINTS

- Build and installation instructions for the RCP images
- Build and installation instructions for the Border Router Host
- OpenThread Border Router configuration information
- OpenThread resources

1 Introduction

This application note is intended for software engineers who wish to develop an OpenThread Border Router (OTBR). It assumes some familiarity with OpenThread and basic Thread concepts. For an introduction to OpenThread and information on Thread concepts, visit https://openthread.io/. For information on OTBR setup and installation, refer to https://openthread.io/guides/border-router.

This application notes assumes that you have downloaded Simplicity Studio 5 (SSv5) and the Silicon Labs OpenThread SDK and are generally familiar with the SSv5 Launcher perspective. SSv5 installation and getting started instructions, along with a set of detailed references, can be found in the online *Simplicity Studio 5 User's Guide*, available on https://docs.silabs.com/ and through the SSv5 help menu. Refer to *QSG170: Silicon Labs OpenThread Quick Start Guide* for more information about configuring, building, and flashing OpenThread sample applications.

This application note addresses the following topics:

- **Build and Installation Instructions for the RCP Images**
  Explains the build and installation procedure for the Radio Co-Processor (RCP) image using UART as well as SPI interfaces.
- **Build and Installation Instructions for the OpenThread Border Router**
  Defines the build and installation procedure for the OpenThread Border Router on POSIX-based platforms, including an option to deploy a pre-built Docker container for the Raspberry Pi.
- **OTBR Configuration Information**
  Provides OTBR information such as how to configure various Border Router features and the Network Address Translation (NAT64) interface.
- **Additional OpenThread Resources**
  Includes links to OpenThread Resources.

1.1 Hardware Requirements

A Thread Border Router has two components:

- A Raspberry Pi host with Thread Border Router support (Recommended: Raspberry Pi 3 Model B+ or above)
- A Thread-capable Silicon Labs Radio Co-processor (RCP)

To create the RCP, you need the following:

- EFR32MG Wireless Starter Kit or Thunderboard Sense 2 Sensor-to-Cloud Advanced IoT Kit
- Silicon Labs board capable of Thread communication

Note: See Table 2-1 for a list of Silicon Labs Precompiled RCP Images.

2 Build and Installation Instructions for the RCP Images

Note: The following instructions only apply to RCP images built using Simplicity Studio for a given GSDK release.

To build an RCP image using the latest OpenThread, follow instructions on the ot-efr32 repo.

2.1 Use Precompiled RCP Images

Silicon Labs has precompiled images available for these boards with their associated image locations. The default precompiled images are configured for UART interface.

Note: By default, the Silicon Labs GSDK uses Thread protocol version 1.3, which also applies to the following precompiled RCP images.

Table 2-1. Silicon Labs Precompiled RCP Images

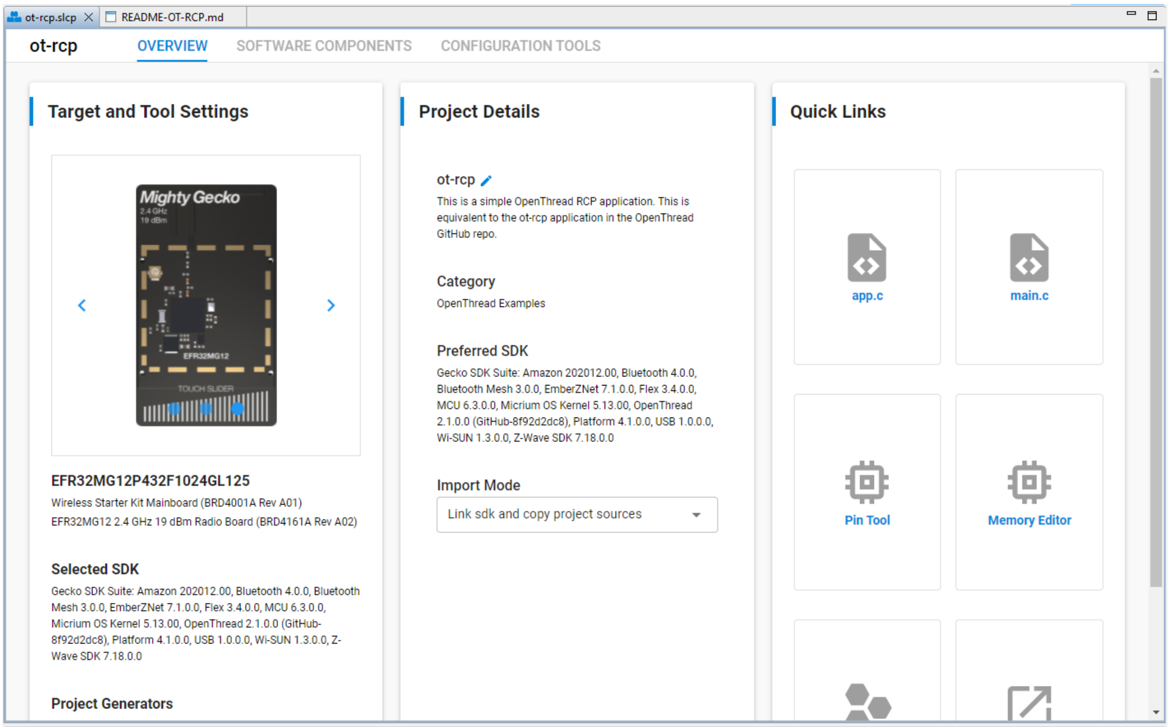
Board Name	Image Location
brd4161a	protocol/openthread/demos/ot-rcp/ot-rcp-brd4161a.s37
brd4166a	protocol/openthread/demos/ot-rcp/ot-rcp-brd4166a.s37
brd4168a	protocol/openthread/demos/ot-rcp/ot-rcp-brd4168a.s37
brd4180a	protocol/openthread/demos/ot-rcp/ot-rcp-brd4180a.s37
brd4304a	protocol/openthread/demos/ot-rcp/ot-rcp-brd4304a.s37

2.2 Build RCP Images Using Simplicity Studio 5

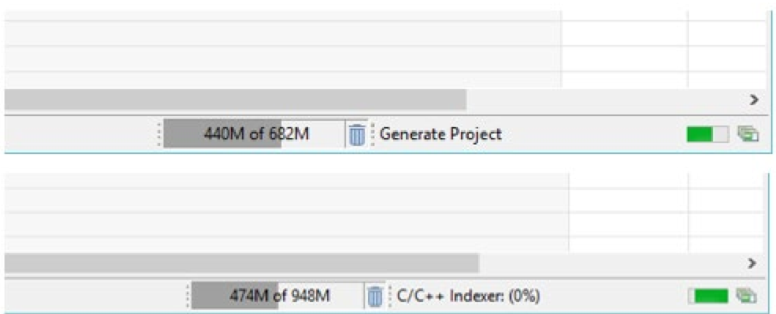
Silicon Labs has sample applications for several standard OpenThread images.

1. Select **ot-rcp** as an example for the default RCP image for the OpenThread Border Router over UART interface.
2. With your target part connected to your computer, open Simplicity Studio 5's **File** menu and select **New > Silicon Labs Project Wizard**. The Target, SDK, and Toolchain Selection dialog opens. Click **NEXT**.
3. The Example Project Selection dialog opens. Use the **Technology Type** and **Keyword** filters to search for **ot-rcp** as an example for the default RCP image for the OpenThread Border Router. Select it and click **NEXT**.
4. The Project Configuration dialog opens. Rename your project, change the default project file location, and determine if you will link to or copy project files. Note that, if you change any linked resource, it is changed for any other project that references it. Click **FINISH.**

5. The Simplicity IDE Perspective opens with the Project Configurator open to the **OVERVIEW** tab. See the online *Simplicity Studio 5 User's Guide* for details about the functionality available through the Simplicity IDE perspective and the Project Configurator.



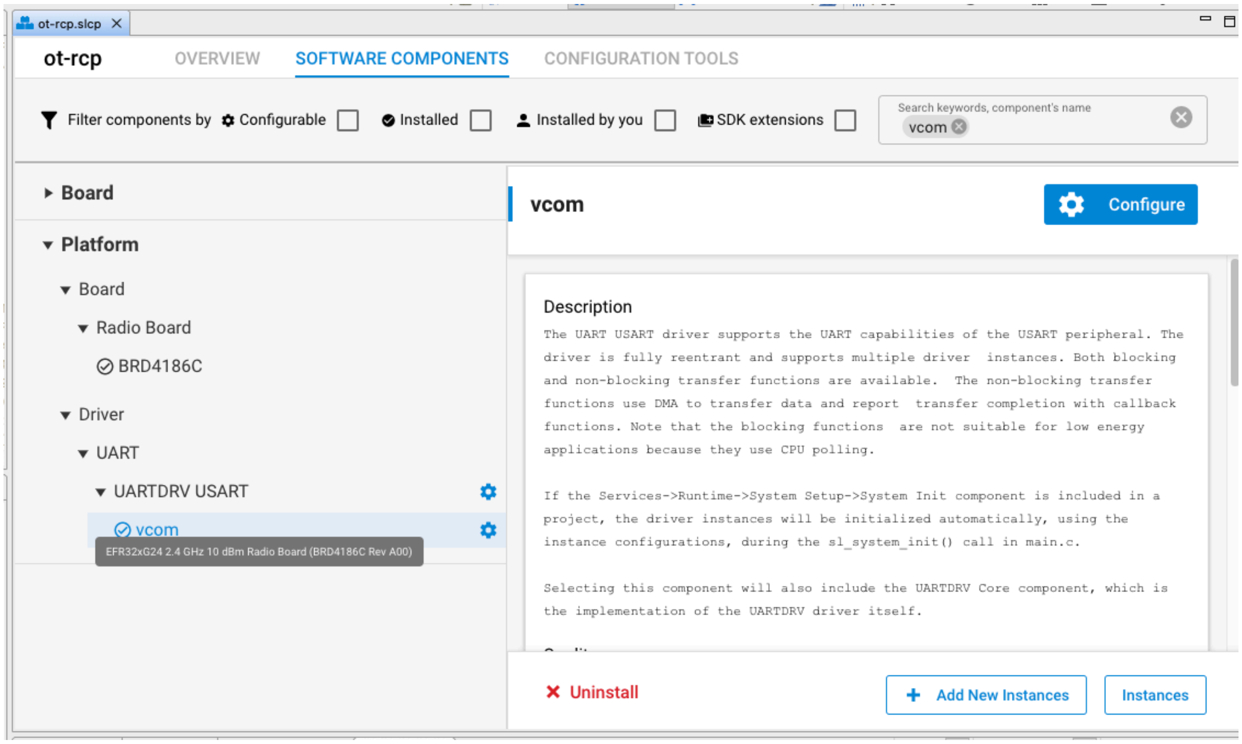
6. Make any configuration changes to the software components, as described in the next section. The autogeneration progress is available in the bottom right of the Simplicity IDE perspective. Make sure that progress is complete before you build.



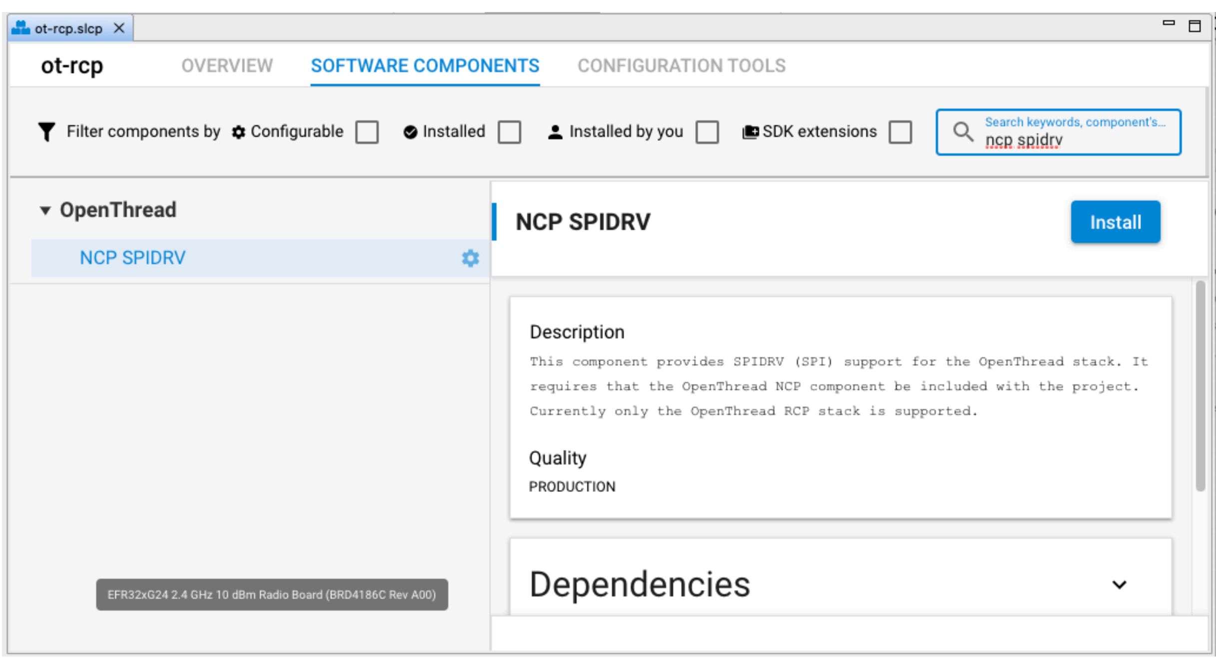
7. Compile and flash the application image as described in *QSG170: Silicon Labs OpenThread Quick Start Guide*.

2.3 Configure RCP Image for SPI From Default OT-RCP Application Using Simplicity Studio 5

1. Generate ot-rcp application as described in section 2.2 steps 1 – 4.
2. Under the **SOFTWARE COMPONENTS** tab in your RCP project (.slcp), expand the **Platform** menu. Select **vcom** under the **UARTDRV USART** component.



3. Click **Uninstall** to remove the component, which uninstalls the **UARTDRV Core** component as well.
4. Under the **SOFTWARE COMPONENTS** tab in your RCP project (.slcp), expand the **OpenThread** menu. Select the **NCP SPIDRV** component and click **Install**.

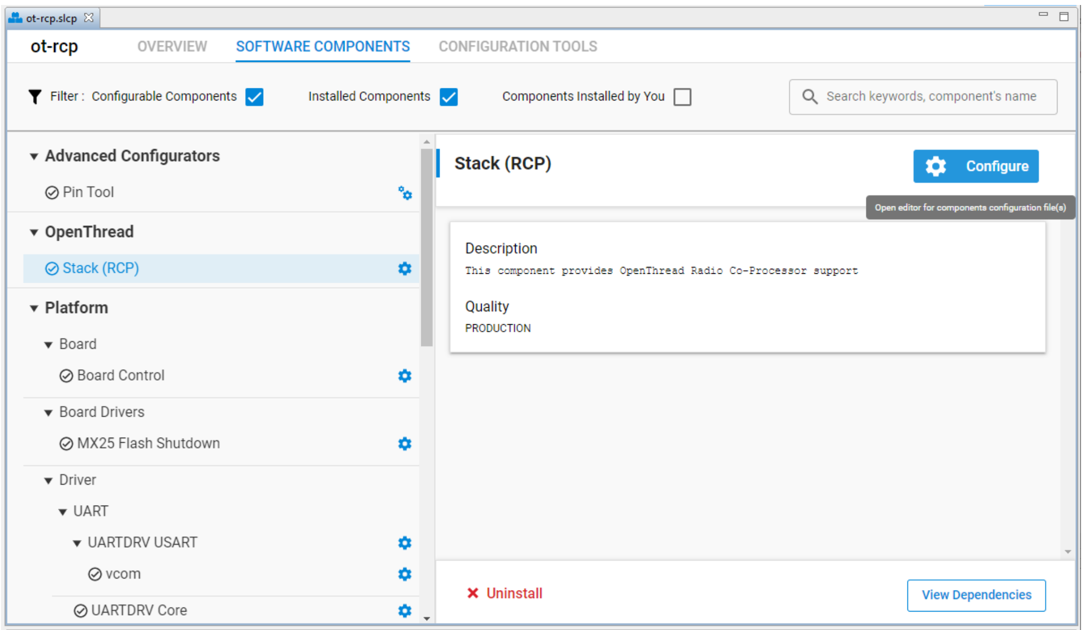


5. Compile and flash the application image as described in *QSG170: Silicon Labs OpenThread Quick Start Guide.*

2.4 Configure OpenThread Options in the RCP Images Using Simplicity Studio 5

1. Under the **SOFTWARE COMPONENTS** tab in your RCP project (.slcp), expand the **OpenThread** menu. Select **Stack (RCP)** for an RCP build.
2. Click **Configure** to change the settings associated with the OpenThread build.

Note: You can select the Configurable Components and Installed Components checkboxes to filter to only those components you can configure successfully**.**

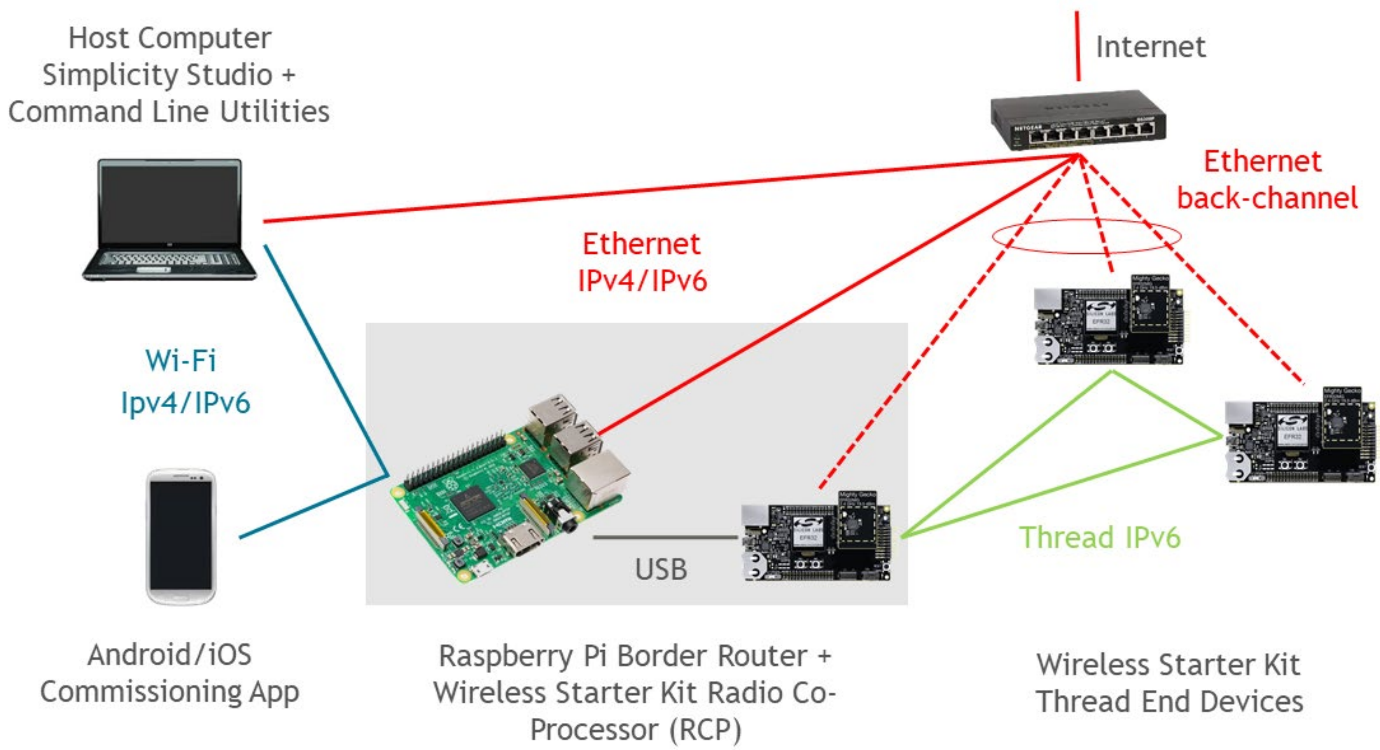


3. Configure the various compile-time settings for your RCP project. The various build options are explained in the OpenThread documentation at https://openthread.io/guides/build.
4. For Coexistence with WiFi configurations, see *AN1017 Zigbee and OpenThread Coexistence with WiFi.*

3 Build and Installation Instructions for the Border Router Host

3.1 Install the Hardware

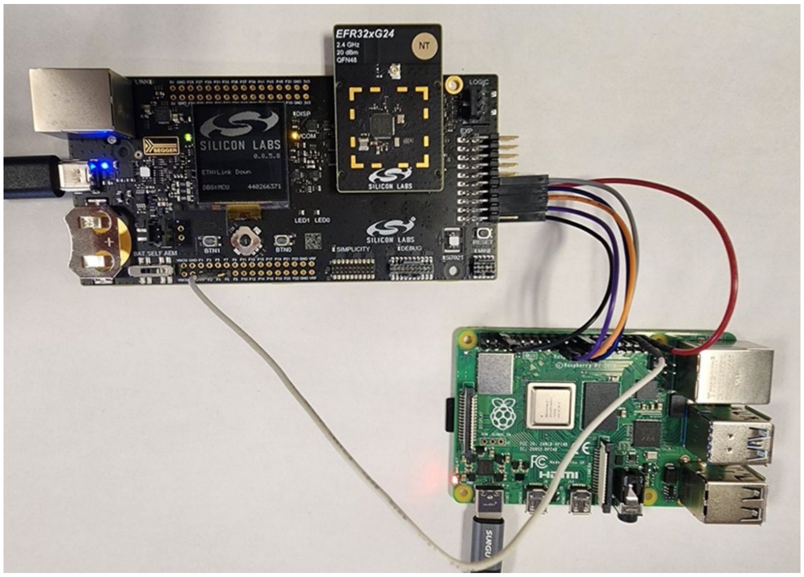
Connect each Wireless Starter Kit main board and the host computer to an Ethernet switch with an Ethernet cable as shown in the following figure. These connections will permit programming and network analysis of the RCP and end devices. Optionally, end devices may be connected to the host computer by USB rather than Ethernet. To connect Raspberry Pi Border Router with RCP over SPI, you can either hardwire the SPI pins with WSTK's expansion connector or you can use wireless expansion board (brd8016), which mounts on the top of Raspberry Pi.



3.1.1 Hardwire SPI Connections Between Raspberry Pi and WSTK

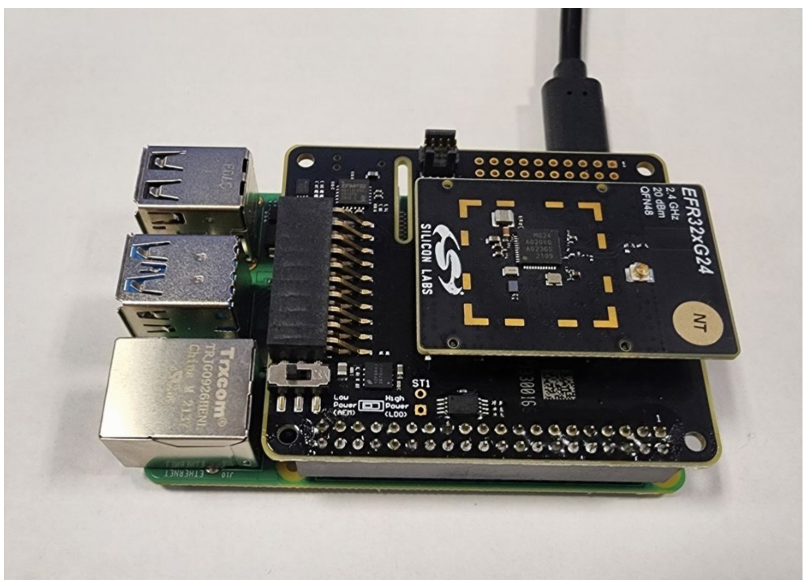
Connect the SPI pins as shown below:

Raspberry Pi Connector (SPI Pins)	WPK's Expansion Connector (brd4002)
GPIO10 / Pin19 (MOSI)	Pin 4
GPIO9 / Pin21 (MISO)	Pin 6
GPIO11 / Pin23 (SCLK)	Pin 8
GPIO7/8 / Pin24/26 (CS0/CS1)	Pin 10
GPIO21 / Pin40 (Interrupt line)	Pin 7
GND / Pin 6	Pin 1
GPIO20 / Pin38 (Reset line)	Pin F4 on breakout connector



3.1.2 Wireless Expansion Board for SPI Connections Between Raspberry Pi and WSTK

You can also use a wireless expansion board which mounts on the top of Raspberry Pi to avoid hardwire connection as shown below.



Note: Use correct `OTBR_AGENT_OPTS` as described in section 3.2.3 OTBR Configurations using SPI Interface depending on the SPI connections.

3.2 Install and Configure a Raspberry Pi for Use with RCP

Two different methods for installing the OTBR software in a Raspberry Pi for use with an RCP are:

- Manual installation
- Docker container

3.2.1 Manual Installation

This guide covers how to build the OTBR using the tools provided by the Silicon Labs GSDK. You may also install the OTBR using the instructions detailed at https://openthread.io/guides/border-router/build. Note that this is a different build option and either guide may be followed to build the OTBR.

For the manual setup, use the included version of the ot-br-posix in the Silicon Labs SDK:

```
util/third_party/ot-br-posix
```

You can findthe Thread specification for a selected GSDK under util/third_party/openthread/README.md. If you have already downloaded the GSDK as part of Simplicity Studio, skip the following step.

Begin by cloning the GSDK repository:

```
git clone https://github.com/SiliconLabs/gecko_sdk.git
```

Check out the desired GSDK branch. For this guide, we use the default branch.

In the util/third_party /ot-br-posix directory run:

```
sudo ./script/bootstrap
```

After running the bootstrap step, make sure to check out the right version of the openthread stack in third_party/openthread/repo, by either copying or creating a symbolic link to the OpenThread stack:

```
util/third_party/ot-br-posix/third_party/openthread/repo -> util/third_party/openthread
```

Make sure that the git commit under third_party/openthread/repo is supported by the release. Refer to the OpenThread Release Notes for the stable commits supported by a given release.

Now run the setup script:

```
sudo ./script/setup
```

Note that to enable border routing, you must specify your platform's Ethernet or Wi-fi interface:

For Ethernet run `sudo INFRA_NAME=eth0 ./script/setup`

For Wi-Fi run `sudo INFRA_NAME=wlan0 ./script/setup`

To use Silicon Labs-specific configuration settings for border router, you can grab the special configuration header hosted in the GSDK under **<gsdk_location>/protocol/openthread/platform-abstraction/posix/openthread-core-silabs-posix-config.h.**

Then run the setup as follows:

```
# Copy the config file to a known include path

sudo cp <gsdk_location>/protocol/openthread/platform-abstraction/posix/openthread-core-silabs-posix-config.h <gsdk_location>/util/third_party/openthread/src/posix/platform/

# Run the setup by specifying the above config header

sudo INFRA_IF_NAME=eth0 "OTBR_OPTIONS="-DOT_CONFIG=openthread-core-silabs-posix-config.h" ./script/setup
```

To build OTBR with SPI interface, specify RCP bus as follows:

```
sudo INFRA_IF_NAME=eth0 "OTBR_OPTIONS="-DOT_POSIX_RCP_SPI_BUS=ON" ./script/setup
```

Refer to /src/core/config and openthread/examples/README.md for compile-time constants and cmake build options, respectively.

3.2.2 OTBR Configurations Using UART Interface

- Configure the desired tty* port to use for the OTBR to connect your RCP at startup. Look for the tty* port of the RCP device. The easiest way to do this is to look for a `/tty/dev…` entry once the RCP is connected. It should generally either be `/dev/ttyUSB0` or `/dev/ttyACM0`.
- Edit the `/etc/default/otbr-agent` file and look for the `OTBR_AGENT_OPTS` configuration. Include the tty* port name in that parameter as follows:
  ```
  OTBR_AGENT_OPTS="-I wpan0 spinel+hdlc+uart:///dev/ttyACM0"
  ```
- If running a Backbone Border Router (Thread protocol version 1.2 or above), add the backbone interface as follows:
  ```
  OTBR_AGENT_OPTS="-I wpan0 -B eth0 spinel+hdlc+uart:///dev/ttyACM0"
  ```
- If running a Thread 1.3 Border Router, specify the Thread Radio Encapsulation Link (TREL) interface to enable Thread over Infrastructure links as follows:
  ```
  OTBR_AGENT_OPTS="-I wpan0 -B eth0 spinel+hdlc+uart:///dev/ttyACM0 trel://eth0"
  ```

3.2.2.1 UART Baud Rate Settings

- Update the radio URL options with a higher baud rate for the following scenarios (recommended: `uart-baudrate=460800`):
  - Fragmentation / reassembly issues over UART
  - Costly operations such as DTLS with long IPv6 packets
  - High data rate use cases
  - Time sensitive operations such as CSL transmissions

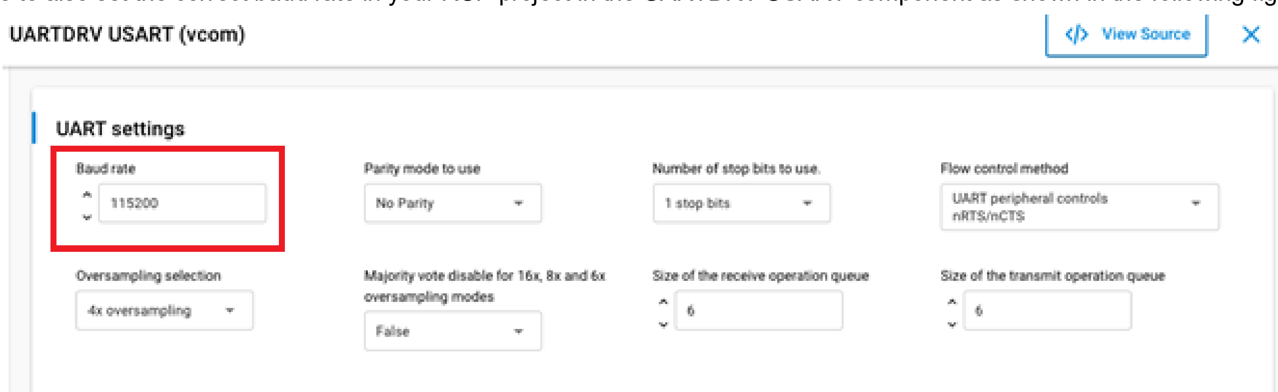
Example:

```
OTBR_AGENT_OPTS="-I wpan0 spinel+hdlc+uart:///dev/ttyACM0?uart-baudrate=460800"
```

If you are using this higher baud rate on the host, make sure to also change the baud rate on the RCP by issuing the following command on the admin console of the RCP's WSTK:

> serial vcom config speed 460800

Make sure to also set the correct baud rate in your RCP project in the UARTDRV USART component as shown in the following figure:



- To verify the baud rate in use, issue the command:
  ```
  stty -F /dev/ttyACM0
  ```

3.2.3 OTBR Configurations using SPI Interface

- Configure the desired SPI ports to use for the OTBR to connect your RCP at startup. Ensure the SPI interface is enabled on the Raspberry Pi. If not:
- Enable it by adding following in `/boot/config.txt`

```
dtparam=spi=on
dtoverlay=disable-bt        #Maybe not required
```

- Remove/comment `dtoverlay=spi0-1cs,cs0_pin=26` if it is defined in /boot/config.txt.
- Reboot the Raspberry Pi.
- Use this command to validate the SPI port `'ls /dev | grep spi'`.
- Two devices should be visible, `spidev0.0` and `spidev0.1`, depending on your Raspberry Pi version.
- Edit the `/etc/default/otbr-agent` file and look for the `OTBR_AGENT_OPTS` configuration. Include the correct GPIOs in that parameter as follows as per the hardware setup, see sections 3.1.1 and 3.1.2.
- Use the following parameters if Raspberry Pi is hardwired to SPI pins on WSTK's expansion connector. Use `spidev0.0` if using Raspberry Pi's CS0 or replace to `spidev0.1` for CS1 pin.

```
OTBR_AGENT_OPTS="-I wpan0 -B eth0 spinel+spi:///dev/spidev0.0?gpio-int-device=/dev/gpio-chip0&gpio-int-line=21&gpio-reset-device=/dev/gpiochip0&gpio-reset-line=20&no-reset=1&spi-speed=1000000"
```

- Use the following parameters if the radio board is mounted on Raspberry Pi using wireless expansion board (brd8016A).

```
OTBR_AGENT_OPTS="-I wpan0 -B eth0 spinel+spi:///dev/spidev0.0?gpio-int-device=/dev/gpio-chip0&gpio-int-line=22&gpio-reset-device=/dev/gpiochip0&gpio-reset-line=23&no-reset=1&spi-speed=1000000"
```

- If using custom hardware connections, then make sure to provide respective GPIOs pins.
- Start `otbr-agent` service by either rebooting the Raspberry Pi or issue `> sudo systemctl restart otbr-agent`.
- Issue the `> sudo ot-ctl state` command on the Raspberry Pi to see the status of the connection between the host and RCP.
- Check whether all required services are running on the OTBR.

`sudo systemctl status` should not report any services as running in a "degraded" state.

- Check `/var/log/syslog` for a running log of otbr-agent.

3.2.4 OTBR Configuration for CSL

- When running OTBR as a CSL transmitter, OTBR can sometimes fail to transmit CSL packets with the error "`Handle transmit done failed: Abort`". This can happen if `OPENTHREAD_CONFIG_MAC_CSL_REQUEST_AHEAD_US` is set too low.

  The default SiLabs POSIX config header includes a recommended value (5000) for this parameter. When building your OTBR from the instructions in section 3.2.1, make sure to use the latest `openthread-core-silabs-posix-config.h`

  If building the OTBR on your own using the instructions on the OpenThread website, then either:

  - Modify the value of `OPENTHREAD_CONFIG_MAC_CSL_REQUEST_AHEAD_US` in `ot-br-posix/third_party/open-thread/repo/src/core/config/mac.h`
    or
  - Pass the value during setup as follows:

  ```
  sudo OTBR_OPTIONS="-DCMAKE_CXX_FLAGS='-DOPENTHREAD_CONFIG_MAC_CSL_REQUEST_AHEAD_US=5000'" ./script/setup
  ```

3.2.5 Installation Guidance

- Starting with GSDK 4.2.0, Silicon Labs OpenThread Border Router uses the OpenThread-based NAT64 implementation, which automatically sets up a NAT64 prefix for address translation purposes. See example at https://openthread.io/guides/border-router/docker/test-connectivity.

  Silicon Labs does not recommend using the default NAT configuration on a network using 192.168.x.x addresses because that NAT uses those addresses by default on the NAT64 interface.
- For properly resolving mDNS queries, make sure the "**hosts:**" line under **/etc/nsswitch.conf** looks like the following:

  ```
  hosts:          files mdns4 minimal mdns5 minimal dns
  ```
- For Thread 1.2 backbone routing and Thread 1.3 features, the onboard OTBR processes manage IPv6 prefixes and routing, so dhcpcd management of ipv6 should be disabled.

  Check that the following lines are present in **/etc/dhcpcd.conf:**

```
noipv6
noipv6rs
```

3.2.6 OTBR Status

- Issue the `> sudo ot-ctl state` command on your Raspberry Pi to see the status of the connection between the host and RCP.
- Check whether all required services are running on the OTBR.
  `sudo systemctl status` should not report any services as running in a “degraded” state.
- Check `/var/log/syslog` for a running log of otbr-agent.

3.2.7 Using ot-ctl to Configure and Control the OpenThread Border Router

For a full command list, run:

```
> sudo ot-ctl help
```

Refer to https://openthread.io/guides/border-router/external-commissioning for examples on how to manually set up a Thread Network and examples on how to enable an external commissioner.

Run these two commands to check for a running Thread Network:

```
> sudo ot-ctl state
```

and

```
> sudo ot-ctl ifconfig
```

Note: The error message `OpenThread Daemon is not running` indicates a problem with the RCP connection. Check for a valid `/dev/tty` entry and check that a valid RCP application was flashed onto the device.

3.2.8 OTBR Feature Configuration for Certification

For information on how to properly configure OpenThread Border Router features and services, visit https://openthread.io/guides/border-router.

The following flags are recommended for an OTBR “**Device Under Test (DUT)**” for Thread Certification:

(See section 3.2.1 for instructions on obtaining Silicon Labs-specific configuration settings for border router: openthread-core-silabs-posix-config.h)

```
sudo RELEASE=1 BACKBONE_ROUTER=1 NAT64=1 DNS64=1 ./script/bootstrap

sudo INFRA_IF_NAME=eth0 \
     RELEASE=1 BACKBONE_ROUTER=1 BORDER_ROUTING=1 NAT64=1 DNS64=1 \
     OTBR_OPTIONS="-DOT_THREAD_VERSION=1.3 \
                   -DOT_CONFIG=openthread-core-silabs-posix-config.h \
                   -DOTBR_DUA_ROUTING=ON
                   -DOTBR_DNSSD_DISCOVERY_PROXY=ON
                   -DOTBR_SRP_ADVERTISING_PROXY=ON" \
     ./script/setup
```

3.2.9 Docker Installation

Note: The following Docker containers are only supposed to be used with RCPs built using Simplicity Studio 5 for a given release. Be sure to match the container tag version with the GSDK version that you are testing with.

Silicon Labs recommends deploying the company’s Docker container with the OTBR. Running the OTBR in a container allows for creation of easily deployable artifacts and fast development prototyping and testing.

Silicon Labs provides the following pre-built Docker containers (with tags), hosted on DockerHub:

https://hub.docker.com/r/siliconlabsinc/openthread-border-router/tags

For proprietary radio support (alpha release):

https://hub.docker.com/r/siliconlabsinc/openthread-border-router-proprietary-na-915/tags

3.2.9.1 Prerequisites

- On the SD card, make sure to flash the Raspberry Pi OS Lite image or Raspberry Pi OS with Desktop.
- Make sure to update the local repositories and package manager (**apt-get update** and **apt-get upgrade** prior to installing Docker).
- Optional but recommended: Install Haveged for better entropy conditions.

3.2.9.2 Installation Guidance

- Make sure to reboot after any updates:

  ```
  curl -sSL https://get.docker.com | sh
  ```

- Once finished, you can modify the Docker user settings to not require `sudo` before each command:

  ```
  sudo usermod -aG docker $USER
  ```

- Raspberry Pi and Linux users, make sure to run:

  ```
  sudo modprobe ip6table_filter
  ```

  for OTBR firewall support. This allows OTBR scripts to create rules inside the Docker container before otbr-agent starts.
  To make sure this setting persists between reboots, add the following line to `/etc/modules`:

  ```
  ip6table_filter
  ```

  If this step is not completed, modprobe errors may be displayed when starting a Docker container.
- Issue the following commands to install the containers. Note that only one Border Router container can be running at one time with the RCP. Also, be sure to verify the RCP version (Thread protocol version 1.3) that should be run against this container.

  ```
  UART interface:
  docker pull siliconlabsinc/openthread-border-router:gsdk-4.4.0

  SPI interface:
  docker pull siliconlabsinc/openthread-border-router:gsdk-4.4.0_spi
  ```

- To run an OpenThread Border Router (default is Thread protocol version 1.3), issue the following command:
  - Example for UART interface

```
docker run -d --name "otbr" \
      --sysctl "net.ipv6.conf.all.disable_ipv6=0 net.ipv4.conf.all.forwarding=1 net.ipv6.conf.all.forwarding=1" \
      -p 8080:80 --dns=127.0.0.1 -it \
      --volume /dev/ttyACM0:/dev/ttyACM0 \
      --privileged siliconlabsinc/openthread-border-router:gsdk-4.4.0 \
      --radio-url spinel+hdlc+uart:///dev/ttyACM0?uart-baudrate=460800 \
      --backbone-interface eth0
```

(See sections 3.2.2 and 3.2.2.1 for notes on configuring the UART Radio URL)

  - Example for SPI interface (for more information, see section 3.2.3 OTBR Configurations using SPI Interface)

```
docker run -d --name "otbr" \
      --sysctl "net.ipv6.conf.all.disable_ipv6=0 net.ipv4.conf.all.forwarding=1 net.ipv6.conf.all.forwarding=1" \
      -p 8080:80 --dns=127.0.0.1 -it \
      --volume /dev/spidev0.0:/dev/spidev0.0 \
      --privileged siliconlabsinc/openthread-border-router:gsdk-4.4.0 \
      --radio-url     spinel+spi:///dev/spidev0.0?gpio-int-device=/dev/gpiochip0&gpio-int-line=21&gpio-reset-de-
vice=/dev/gpiochip0&gpio-reset-line=20&no-reset=1&spi-speed=1000000 \
      --backbone-interface eth0
```

(See section 3.2.3 OTBR Configurations using SPI Interface for notes on configuring the SPI Radio URL)

Use additional arguments to configure the containers. For more information, see the section below or the Dockerfile in the `ot-br-posix` installation directory.

3.2.9.3 Docker Configuration Notes

Note: Silicon Labs-hosted Docker containers are only supposed to be used with RCPs built using Simplicity Studio 5 for a given release. Be sure to match the container tag version with the GSDK version that you are testing with.

- Configure the desired TTY port for the OTBR to connect the RCP at startup. Look for the TTY port of the RCP device. The easiest way to do this is to look for a `/tty/dev…` entry once the RCP is connected. It should generally either be `/dev/ttyUSB0` or `/dev/ttyACM0`.
- Run the Docker installation as follows (example for UART interface):

```
docker run -d --name "otbr" \
      --sysctl "net.ipv6.conf.all.disable_ipv6=0 net.ipv4.conf.all.forwarding=1 net.ipv6.conf.all.forwarding=1" \
      -p 8080:80 --dns=127.0.0.1 -it \
      --volume /dev/ttyACM0:/dev/ttyACM0 \
      --privileged siliconlabsinc/openthread-border-router:gsdk-4.4.0 \
      --radio-url spinel+hdlc+uart:///dev/ttyACM0?uart-baudrate=460800 \
      --backbone-interface eth0
```

  - `-d` ensures that the container runs in detached mode.
  - Review the running logs for the container any time using the `docker logs` command.
  - `--name` is sticky until the docker container is properly closed (or removed).
  - Port `8080` indicates the port of the web server hosting the Border Router management webpage.
- Issue commands directly to the container without having to attach to it:
  ```
  docker exec -ti otbr sh -c "sudo ot-ctl state"
  ```
  For more information, see the docker exec documentation.
- Directly obtain an interactive shell above by issuing this command:
  ```
  docker exec -ti otbr sh -c "sudo ot-ctl"
  ```
- Check the window running the OTBR Docker container for running log output of the Border Router or follow the container log as follows:
  ```
  docker logs [container-id] -f
  ```
- Manage the containers as shown below if they are loaded improperly:
  ```
  # list all container images
  docker images otbr
  # remove existing container
  docker image rm -f <container ID>
  # list running containers
  docker ps -a
  # to remove running container
  docker rm -f <container name>
  ```

4 OpenThread Resources

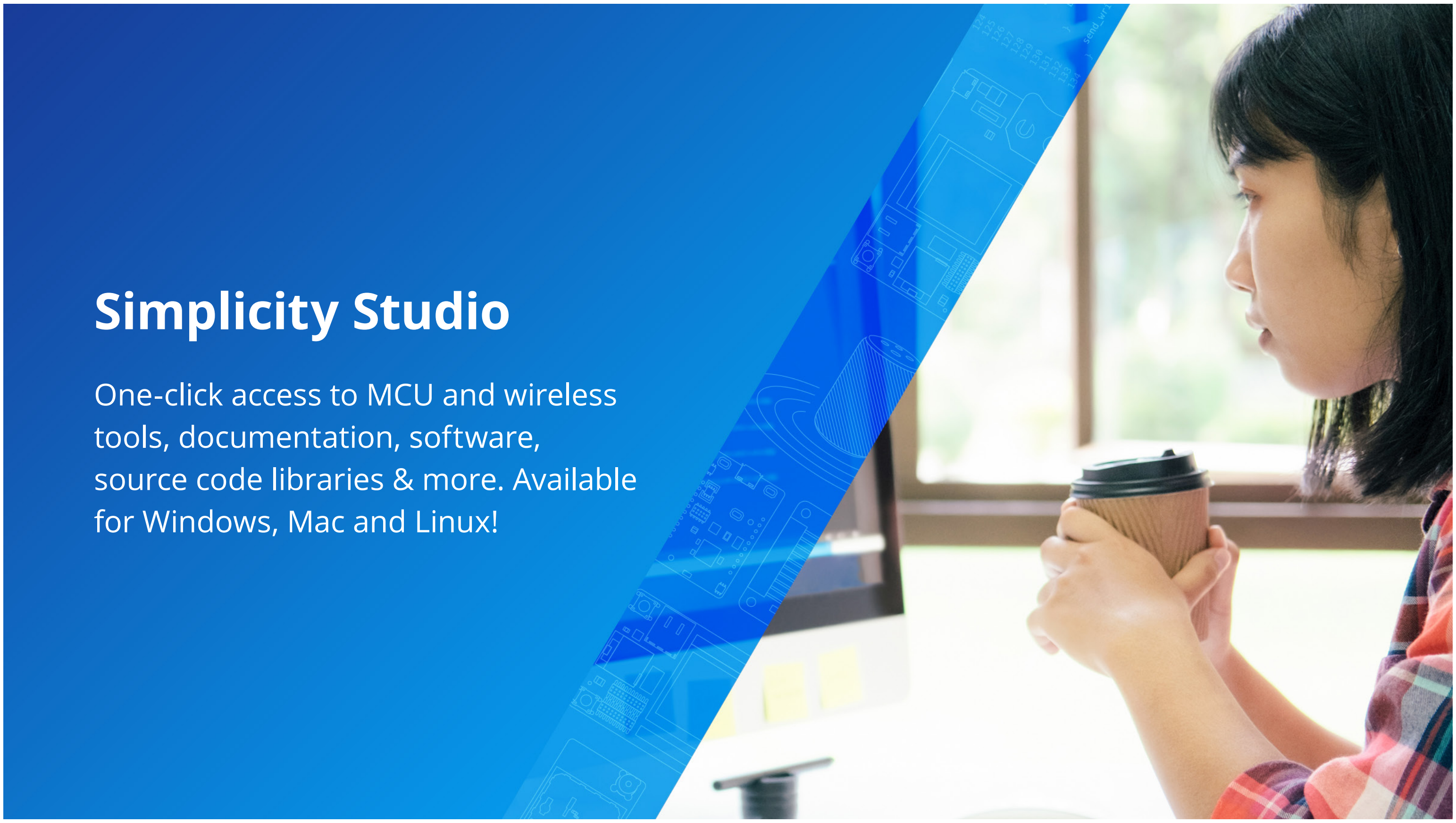
Silicon Labs provides components and configuration options that enable you to configure Thread 1.3 features with sample applications. For more information, see *AN1372: Configuring OpenThread Applications for Thread 1.3*.

To find more resources or take advantage of the OpenThread community pages, visit: https://openthread.io/resources.

For information about the OpenThread Border Router, visit: https://openthread.io/guides/border-router.

Consult these troubleshooting webpages for more information:

- https://openthread.io/guides/border-router/build#verify-services
- https://openthread.io/guides/border-router/access-point#troubleshooting



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

www.silabs.com