UG435.05: Using Real-Time Operating Systems with Silicon Labs Connect v3.x

This chapter of the *Connect v3.x User's Guide* describes how to run the Silicon Labs Connect stack on top of one of the supported Real-Time Operating Systems (RTOS). The Connect stack is delivered as part of the Silicon Labs Proprietary Flex SDK v3.0 and higher. The *Connect v3.x User's Guide* assumes that you have already installed the Simplicity Studio development environment and the Flex SDK, and that you are familiar with the basics of configuring, compiling, and flashing Connect-based applications. Refer to *UG435.01: About the Connect v3.x User's Guide* for an overview of the chapters in the *Connect v3.x User's Guide*.

KEY POINTS

- Introduces Real-Time Operating System support.
- Discusses the CMSIS-RTOSv2 API.
- Discusses the Connect Stack IPC component features.
- Describes the Virtual NCP architecture details.

The *Connect v3.x User's Guide* is a series of documents that provides in-depth information for developers who are using the Silicon Labs Connect Stack for their application development. If you are new to Connect and the Proprietary Flex SDK, see *QSG168: Proprietary Flex SDK v3.x Quick Start Guide*.

Proprietary is supported on all EFR32FG devices. For others, check the device's data sheet under Ordering Information > Protocol Stack to see if Proprietary is supported. In Proprietary SDK version 2.7.n, Connect is not supported on EFR32xG22.

1. Introduction

The Silicon Labs Connect RTOS support is available as software components in Simplicity Studio. To enable this support requires very little effort on the part of application developers.

The RTOS-specific calls of Connect and other Silicon Labs components are implemented using the ARM CMSIS-RTOSv2 API. This allows support of any RTOS implementation (or kernel) that provides the CMSIS-RTOSv2 API. However, Silicon Labs recommends using the implementations that are available in Simplicity Studio as components in the RTOS group, because these were validated against Silicon Labs stacks, including Connect. As of this writing, Micrium OS and FreeRTOS are supported in Flex SDK 3.1. For more details on CMSIS-RTOSv2, see the CMSIS-RTOS2 Documentation.

Silicon Labs does not recommend using the CMSIS-RTOSv2 API from the application unless the developed application needs to support multiple kernels. If the application needs to support a single RTOS, using the API provided by the kernel is simpler and cleaner.

For details on how to use the RTOS implementation of your choice, refer to the documentation for the given RTOS. For documentation on the Micrium OS, see http://doc.micrium.com.

2. Getting Started

Any Connect application can be easily turned into an application running on a RTOS. To complete this task, first install the RTOS component of your choice—for example, **Micrium OS kernel** or **FreeRTOS**. Next, install the **CMSIS Stack IPC** component as shown in the following figure.

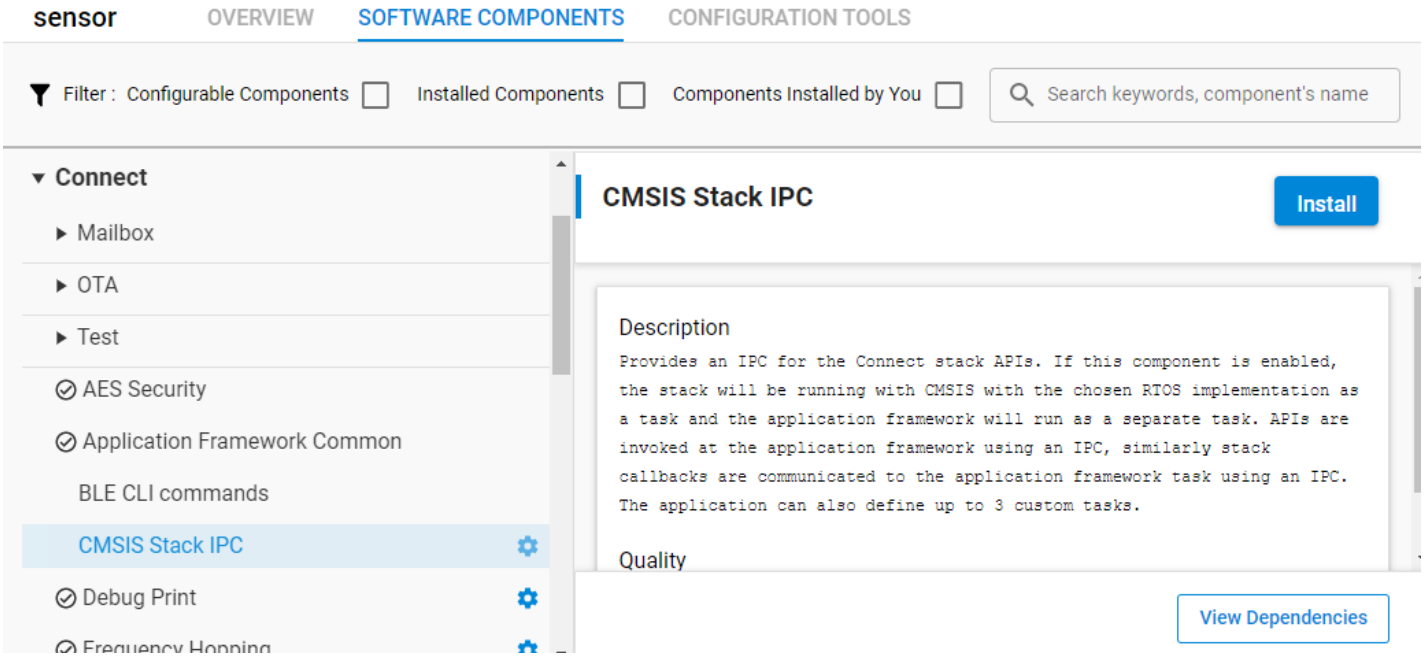


Figure 2.1. CMSIS Stack IPC Component

The CMSIS Stack IPC component has the following configuration options that allow application developers to customize certain settings:

- **CPU usage tracking:** If enabled, the Application Framework compiles in some additional code that allows certain debug tools to have extended debug information from the OS.
- **Connect task priority:** The priority of the task running the stack. Note that this is defined in CMSIS-RTOSv2 priority.
- **Connect task call stack size:** The size in bytes of the call stack used by the stack task.
- **Application Framework task priority:** The priority of the task running the application. Note that this is defined in CMSIS-RTOSv2 priority.
- **Application Framework task call stack size:** The size in bytes of the call stack used by the Application Framework task.
- **Max callback queue size:** Defines the maximum supported number of simultaneous callback messages from the stack task to the application tasks.

Installing any RTOS can change the behavior of other components as well. For example, the CLI Core component will also create an RTOS task for itself, while normally it is called from the main while loop.

3. Virtual NCP Architecture Details

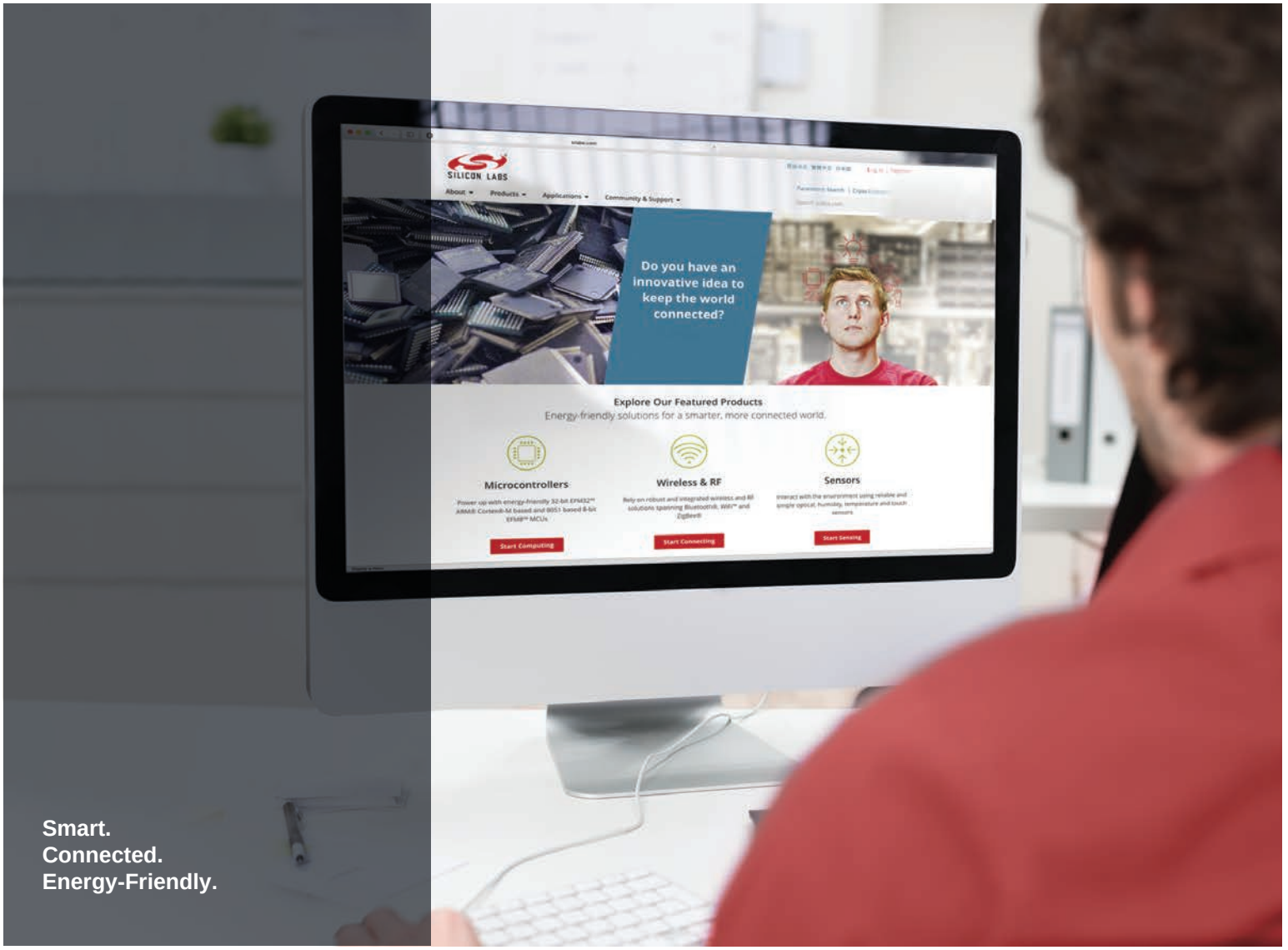
With the CMSIS Stack IPC component enabled, the Connect stack runs within an RTOS task while the Application Framework code runs within a separate RTOS task. This two-task model is also known as the **virtual Network Co-Processor** architecture (**vNCP**) because the processing and communication between the two tasks is the same as for the host/NCP architecture. The difference is that instead of running on separate processors and passing messages to each other through the serial port, the application and NCP run on the same processor but in different Micrium tasks. Message passing is handled using RTOS message queues or protected global data structures and is transparent to the application.

By enabling an RTOS Kernel component, the requested kernel code is added to the project. The CMSIS Stack IPC component will automatically start the two required main tasks for the stack to operate:

- A **Connect stack task** (higher priority task, 39 by default) responsible for running the Connect stack. This task is responsible for the following operations:
  - Periodically tick the Connect stack by invoking the `emberTick()` API.
  - Process incoming IPC (inter-process communication) commands from application tasks (if any) and send out a response.
  - Send callback IPC commands (if any) to the application tasks.
  - Attempt whenever possible to suspend itself to allow lower priority application tasks to run.
- An **Application Framework task** (lower priority, 38 by default) responsible for running the Application Framework code. This task is responsible for the following operations:
  - Periodically calls `emberAfTickCallback()` and `emberAfTick()`, ticking the application and connect components respectively.
  - Run application events.
  - Process incoming callback IPC commands (if any) from the stack task.
  - Attempt whenever possible to suspend itself to allow lower priority custom application tasks to run.

Of course, you can create additional RTOS tasks, but Silicon Labs recommends that you keep the Connect stack task priority as high as possible.

Custom application tasks can call stack APIs freely because the underlying IPC protocol ensures that all stack APIs are thread-safe. Keep in mind that only stack APIs are channeled through the above described vNCP. HAL functions and any MCU-related code might not be thread-safe. You should rely on the documentation for these to verify whether or not they are thread-safe. For non-volatile storage, only NVM3 is supported, because neither simEEv1 nor simEEv2 are thread-safe.



Products
www.silabs.com/products

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS

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

http://www.silabs.com