



AN1389: Running Zigbee Host Applications in a Docker Container

The updated application structure in Zigbee EmberZNet 7.0 and higher no longer supports compiling host applications in MinGW for Windows. This document offers an alternative solution by using a Docker container to run the NCP Host Application.

KEY POINTS

- Example Configuration for NCP Host Application in Docker
- Host configuration solution for Windows users.

1 Introduction

Beginning with Zigbee EmberZNet 7.0, Silicon Labs introduced a new callback framework that automatically supplies *weak* callback functions to the stack, such as `global-callback.c` and `zap-event.c`. These *weak* functions ensure that the project builds when the callbacks are not explicitly implemented. This new framework replaces the 'callback-stub.c' file generated in EmberZNet 6.0 by AppBuilder. Overall, the new callback structure simplifies the user experience in application development but has been seen to cause errors in the linking stage in environments that do not support *weak* functions. More information on the new EmberZNet 7.0 callback framework can be found in *UG491: Zigbee Application Framework Developer's Guide for SDK 7.x and Higher*.

Specifically, the *weak* functions in EmberZNet 7.0 cause issues for Windows users when running applications in MinGW on Cygwin. MinGW treats *weak* function as "Null", causing undefined reference errors in the linking stages of the project build. This application note offers an alternative solution for Windows users that uses Docker to configure the host application. This solution is not required for Linux and MacOS operating systems, as Clang and GCC support *weak* functions. However, if you want to follow a similar procedure for Linux and MacOS, configuring a Docker host application will differ slightly from the procedure below. For more information contact Silicon Labs support through the support portal <https://www.silabs.com/support>.

This application note will walk through the configuration of the host application in a Docker container on a local Windows PC connected to an NCP-configured radio board via USB. To follow the procedure, you should have both Docker Desktop and Silink installed on your local PC. Silink is a Simplicity Studio tool that maps USB Ports to IP addresses, which allows Docker Containers running on Windows PCs to connect to the radio board.

Currently, this configuration does not allow Network Analyzer's network capture to run while the host and NCP applications are running. This is due to conflicts with Silink's port mapping. It is recommended to configure another radio board as a sniffer node and complete the network capture that way.

Requirements:

Docker (Version 20.10.0 or Higher) installed from <https://www.docker.com/products/docker-desktop>. This will install all dependencies needed for this procedure.

Simplicity Studio 5 installed from <https://www.silabs.com/developers/simplicity-studio>

Silink installed as part of Simplicity Studio 5 adapter pack tools (see section [2.3 Step 3: Install and Config Silink](#) for instructions)

EmberZNet SDK 7.x or higher, installed as part of the Gecko SDK Suite (GSDK). See the [Simplicity Studio v5 User's Guide](#) for more information about installing the GSDK.

Radio Board installed on a development mainboard (BRD4162A is used in this example)

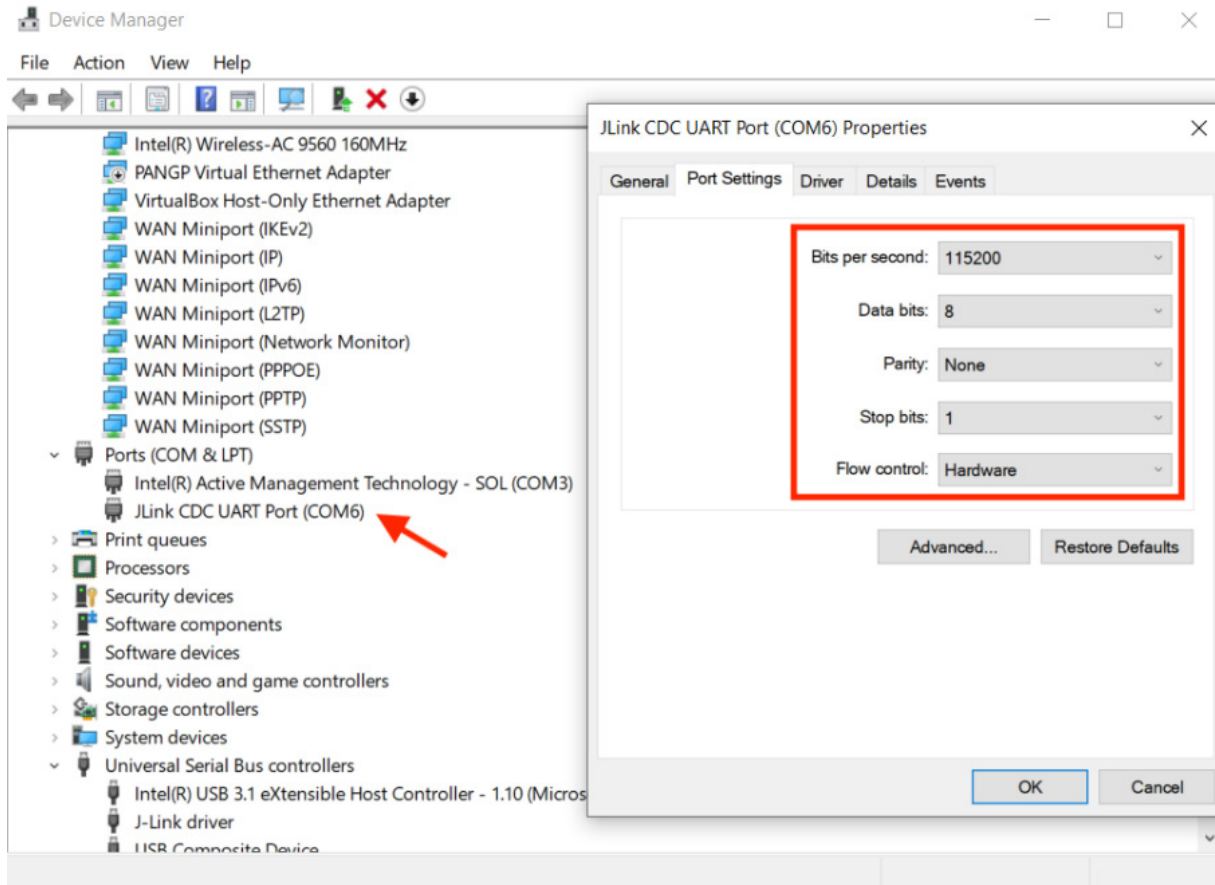
2 Running a Host Application in a Docker Container

2.1 Step 1: Create and Flash NCP-UART-HW Example Project onto BRD4162A

Create a new Simplicity Studio project for the target radio board (BRD4162A). Select the **ncp-uart-hw** example project. Build and flash the project onto BRD4162A. This will be the NCP-configured device. For more information on how to create projects in Simplicity Studio, refer to *QSG180: Zigbee EmberZNet Quick-Start Guide for SDK v7.0 and Higher*.

2.2 Step 2: Config COM Port

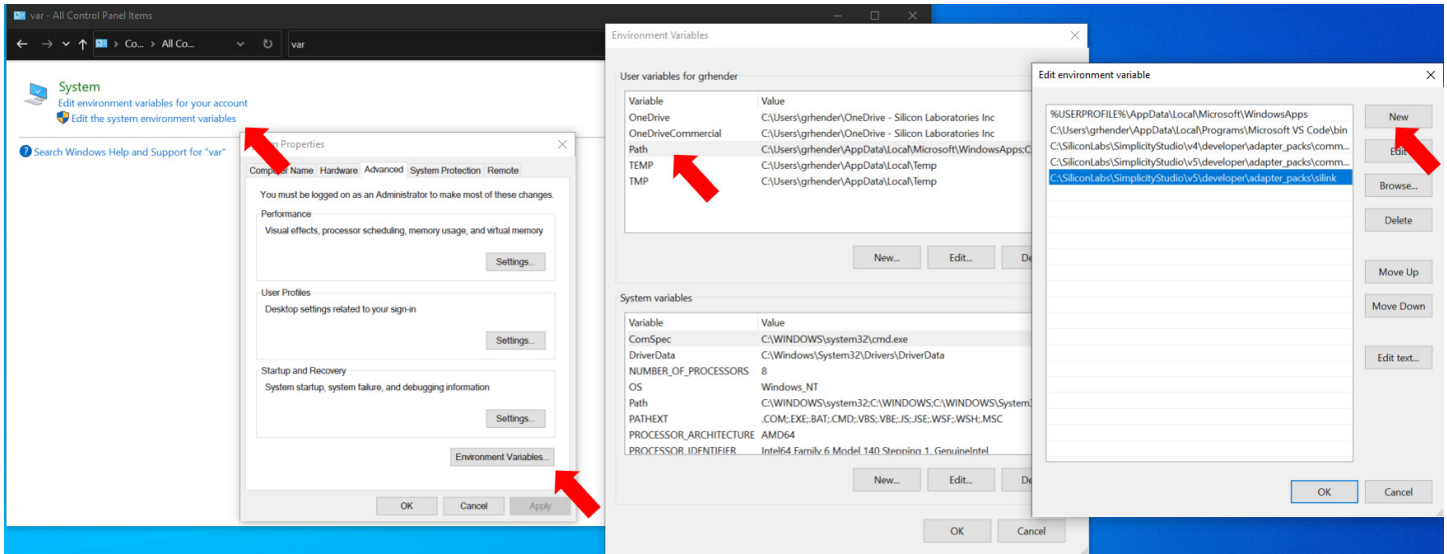
Make sure the radio board (BRD4162A) is connected to the host PC via USB. Open the Windows Device Manager and configure the "Port Settings" to match those used by the NCP application (115200 Baud rate with hardware flow control), as shown in the following figure.



2.3 Step 3: Install and Config Silink

To find Silink's installation path, navigate through the installation path of Simplicity Studios until you reach the adapter pack directory. If you used the default installation path, the file path should look like: C:/SiliconLabs/SimplicityStudio/v5/developer/adapter_packs/Silink.

Add the Silink file path to the Windows "PATH" environment variables. The following figure shows the steps required.



Run "Windows PowerShell" as Administrator and execute the following commands to map the USB ports to IP address ports.

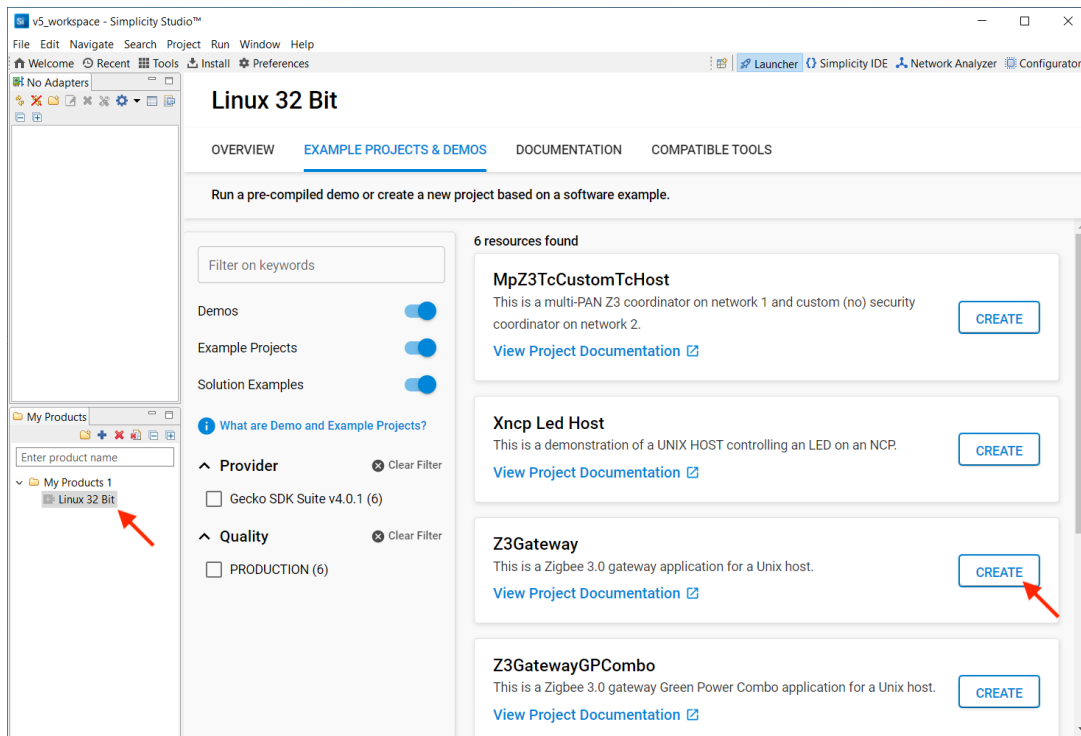
```
silink.exe -automap 4900
```

The following figure shows the desired output from the console.

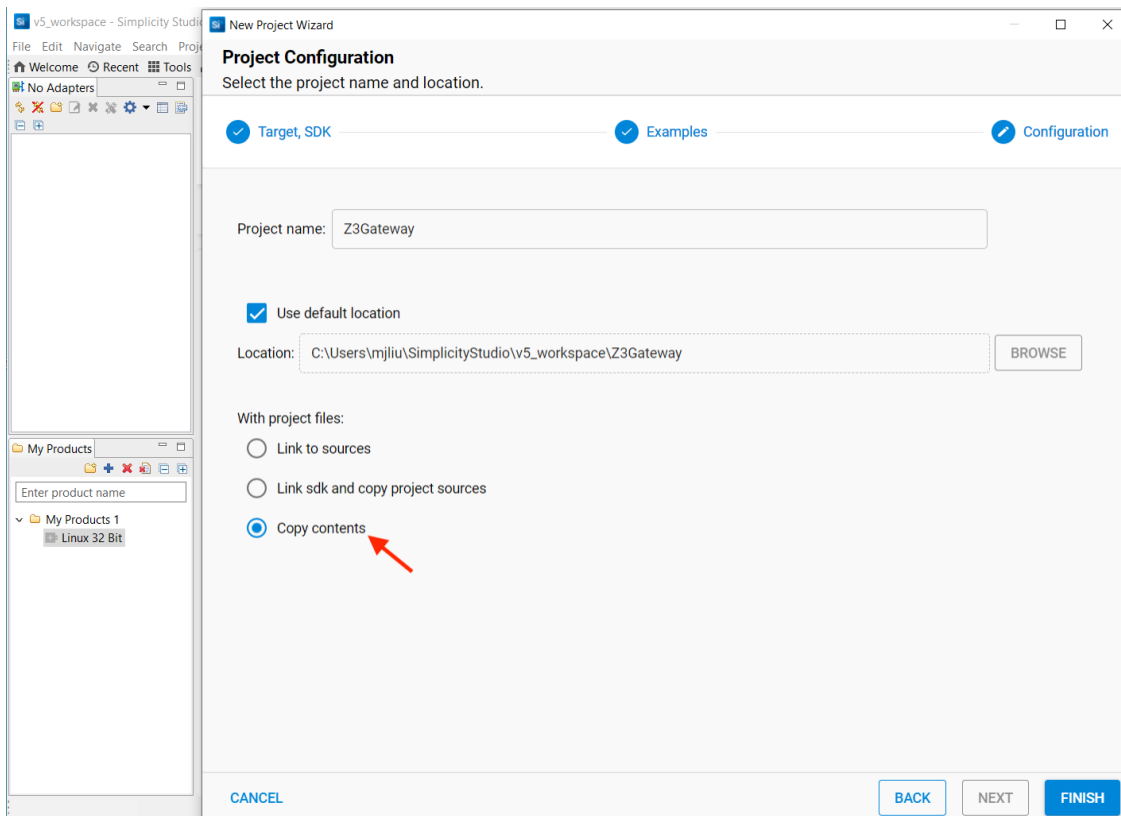
```
PS C:\WINDOWS\system32> silink.exe -automap 4900
Silink version: 0.5.2b238
UART Target connection allowed.
USB Protocol: USBT
Mapping channel VUART0 to port 4900
Mapping channel VCOM0 to port 4901
Mapping channel ADM_CONSOLE to port 4902
Mapping channel DCH to port 4905
OEM:
Device family: Unknown
Serial number: 440187227
silink>
```

2.4 Step 4: Create a Host App for Linux 32 Bit

In the Simplicity Studio My Products view, select **Linux 32 Bit** as the target part and select the Z3 Gateway project, as shown in the following figure.



In the Project Configuration dialog, select **Copy Contents**, as shown in the following figure. This copies necessary files from the GSDK to the project folder, so the Docker Container does not need a complete GSDK.

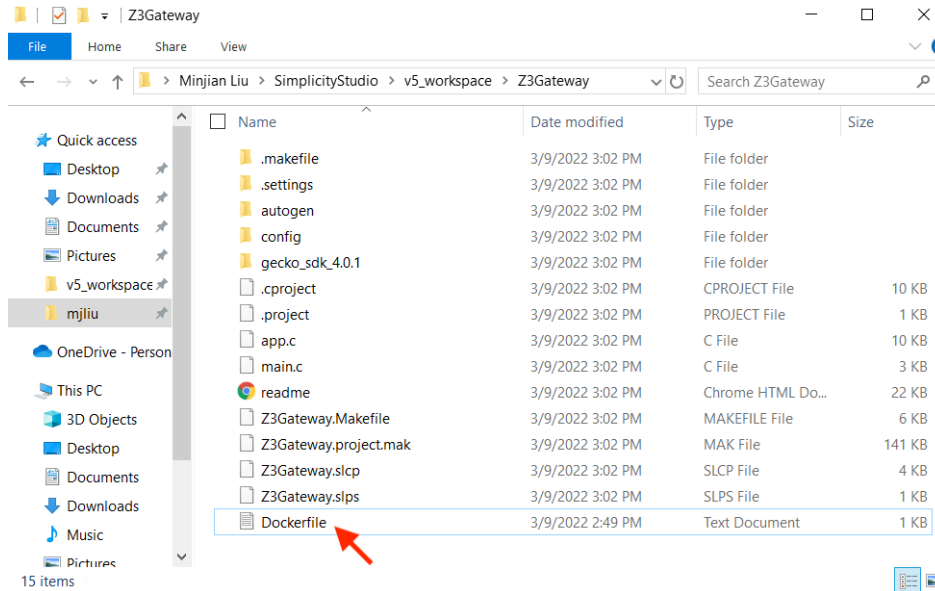


2.5 Step 5: Create a Dockerfile

Create a new file in the root project directory and name it "Dockerfile" without any file extension or suffix. Add the following commands:

```
FROM gcc
RUN apt-get update && apt-get -y install gcc-multilib socat
RUN echo "socat -d TCP:host.docker.internal:4901 pty,raw,echo=0,link=/dev/ttySilink &" >> ~/.bashrc
COPY . /usr/src/Z3Gateway
WORKDIR /usr/src/Z3Gateway
RUN make -f Z3Gateway.Makefile
```

This file configures the Docker container with a GCC environment and downloads the required packages to run the application. The following figure shows the Dockerfile location in the project structure of Z3 Gateway.



2.6 Step 6: Build and Run the Host App in the Container

Run the following commands on the console inside the project folder to build and start the container.

```
docker build . -t z3gateway
docker run -it z3gateway
./build/debug/Z3Gateway -n 0 -p /dev/ttySilink
```

The following figure shows the desired output of these commands.

```
C:\Users\grhender\SimplicityStudio\v5_workspace_cases\Z3Gateway>docker build . -t z3gateway
[+] Building 15.3s (11/11) FINISHED
-> [internal] load build definition from Dockerfile
-> -> transferring dockerfile: 292B
-> [internal] load .dockerignore
-> -> transferring context: 2B
-> [internal] load metadata for docker.io/library/gcc:latest
-> [internal] load build context
-> -> transferring context: 7.87MB
-> [1/6] FROM docker.io/library/gcc@sha256:4b9ff60142b51981fc6727398c39a4ed1214f0a95e95a9a68f11df1ed93a7e8e
-> -> resolve docker.io/library/gcc@sha256:4b9ff60142b51981fc6727398c39a4ed1214f0a95e95a9a68f11df1ed93a7e8e
-> CACHED [2/6] RUN apt-get update && apt-get -y install gcc-multilib socat
-> CACHED [3/6] RUN echo "socat -d TCP:host.docker.internal:4901 pty,raw,echo=0,link=/dev/ttySilink &" >> ~/.bashrc
-> [4/6] COPY . /usr/src/Z3Gateway
-> [5/6] WORKDIR /usr/src/Z3Gateway
-> [6/6] RUN make -f Z3Gateway.Makefile
-> exporting to image
-> -> exporting layers
-> -> writing image sha256:fcd72e2a8cca2df9dd5960dac16316f17f765e0367462f1be1753461ab9923c3
-> -> naming to docker.io/library/z3gateway

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

C:\Users\grhender>docker run -it z3gateway
root@5b6e5c73eaf8:/usr/src/Z3Gateway# ./build/debug/Z3Gateway -n 0 -p /dev/ttySilink
Reset info: 11 (SOFTWARE)
ezsp ver 0x09 stack type 0x02 stack ver. [7.1.0 GA build 191]
Ezsp Config: set address table size to 0x0002:Success: set
Ezsp Config: set TC addr cache to 0x0002:Success: set
Ezsp Config: set MAC indirect TX timeout to 0x1E00:Success: set
Ezsp Config: set max hops to 0x001E:Success: set
Ezsp Config: set tx power mode to 0x8000:Success: set
Ezsp Config: set supported networks to 0x0001:Success: set
Ezsp Config: set stack profile to 0x0002:Success: set
Ezsp Config: set security level to 0x0005:Success: set
Ezsp Value : set end device keep alive support mode to 0x00000003:Success: set
Ezsp Policy: set binding modify to "allow for valid endpoints & clusters only":Success: set
Ezsp Policy: set message content in msgSent to "return":Success: set
Ezsp Value : set maximum incoming transfer size to 0x00000052:Success: set
Ezsp Value : set maximum outgoing transfer size to 0x00000052:Success: set
Ezsp Config: set binding table size to 0x0002:Success: set
Ezsp Config: set key table size to 0x0004:Success: set
Ezsp Config: set max end device children to 0x0006:Success: set
Ezsp Config: set aps unicast message count to 0x000A:Success: set
Ezsp Config: set broadcast table size to 0x000F:Success: set
Ezsp Config: set neighbor table size to 0x0010:Success: set
NCP supports maxing out packet buffers
Ezsp Config: set packet buffers to 255
Ezsp Config: set end device poll timeout to 0x0008:Success: set
Ezsp Config: set zll group addresses to 0x0000:Success: set
Ezsp Config: set zll rssi threshold to 0xFFD8:Success: set
Ezsp Config: set transient key timeout to 0x012C:Success: set
Ezsp Endpoint 1 added, profile 0x0104, in clusters: 8, out clusters 17
Ezsp Endpoint 242 added, profile 0xA1E0, in clusters: 0, out clusters 1
Starting identifying on endpoint 0x01, identify time is 0 sec
Stopping identifying on endpoint 0x01
No endpoints identifying; stopping identification feedback.
Found 0 files

Z3Gateway>
```

After this step, the host application is up and running.

In order to stop the Docker container, enter the following commands into a new console. Docker ps lists all running containers on the local PC. Docker stop then stops whatever container is explicitly specified.

```
Docker ps
Docker stop <container_id>
```



```
C:\Users\grhender>docker ps
CONTAINER ID   IMAGE             COMMAND          CREATED        STATUS          PORTS          NAMES
e520d1a02776  z3gateway:latest "bash"          About a minute ago Up About a minute          flamboyant_satoshi

C:\Users\grhender>docker stop e520d1a02776
e520d1a02776

C:\Users\grhender>docker ps
CONTAINER ID   IMAGE             COMMAND          CREATED        STATUS          PORTS          NAMES
```

Alternatively, the Docker desktop application can run, stop, and pause the Docker containers on the local machine.

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

Disclaimer

Silicon Labs intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Labs products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Labs reserves the right to make changes without further notice to the product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Without prior notification, Silicon Labs may update product firmware during the manufacturing process for security or reliability reasons. Such changes will not alter the specifications or the performance of the product. Silicon Labs shall have no liability for the consequences of use of the information supplied in this document. This document does not imply or expressly grant any license to design or fabricate any integrated circuits. The products are not designed or authorized to be used within any FDA Class III devices, applications for which FDA premarket approval is required or Life Support Systems without the specific written consent of Silicon Labs. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Labs products are not designed or authorized for military applications. Silicon Labs products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons. Silicon Labs disclaims all express and implied warranties and shall not be responsible or liable for any injuries or damages related to use of a Silicon Labs product in such unauthorized applications.

Note: This content may contain offensive terminology that is now obsolete. Silicon Labs is replacing these terms with inclusive language wherever possible. For more information, visit www.silabs.com/about-us/inclusive-lexicon-project

Trademark Information

Silicon Laboratories Inc.[®], Silicon Laboratories[®], Silicon Labs[®], SiLabs[®] and the Silicon Labs logo[®], Bluegiga[®], Bluegiga Logo[®], EFM[®], EFM32[®], EFR, Ember[®], Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Redpine Signals[®], WiSeConnect, n-Link, ThreadArch[®], EZLink[®], EZRadio[®], EZRadioPRO[®], Gecko[®], Gecko OS, Gecko OS Studio, Precision32[®], Simplicity Studio[®], Telegesis, the Telegesis Logo[®], USBXpress[®], Zentri, the Zentri logo and Zentri DMS, Z-Wave[®], and others are trademarks or registered trademarks of Silicon Labs. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. Wi-Fi is a registered trademark of the Wi-Fi Alliance. All other products or brand names mentioned herein are trademarks of their respective holders.



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

www.silabs.com