



Azure RTOS Samples for STM32L4+-DISCO IoT Node using IAR User Guide

Published: September 2020

For the latest information, please see
azure.com/rtos

This document is provided “as-is”. Information and views expressed in this document, including URL and other Internet Web site references, may change without notice.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

Azure RTOS provides OEMs with components to secure communication and to create code and data isolation using underlying MCU/MPU hardware protection mechanisms. It is ultimately the responsibility of the device builder to ensure the device fully meets the evolving security requirements associated with its specific use case.

FileX supports the Microsoft exFAT file system format. Your use of exFAT technology in your products requires a separate license from Microsoft. Please see the following link for further details on exFAT licensing:

<https://www.microsoft.com/en-us/legal/intellectualproperty/mtl/exfat-licensing.aspx>

© 2020 Microsoft. All rights reserved.

Microsoft Azure RTOS, Azure RTOS FileX, Azure RTOS GUIX, Azure RTOS GUIX Studio, Azure RTOS NetX, Azure RTOS NetX Duo, Azure RTOS ThreadX, Azure RTOS TraceX, Azure RTOS Trace, event-chaining, picokernel, and preemption-threshold are trademarks of the Microsoft group of companies. All other trademarks are property of their respective owners.

Revision 6.1

Table of Contents

Overview 1

Getting Started 4

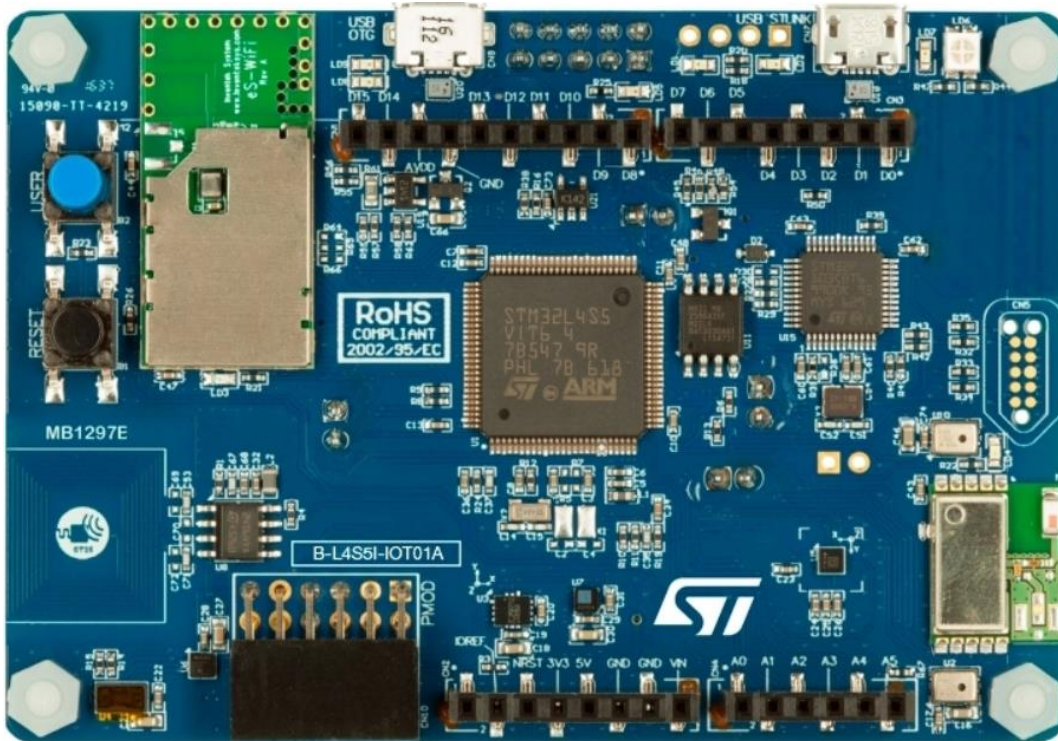
Sample Descriptions 6

 Azure IoT HUB Connectivity Sample **Error! Bookmark not defined.**

 ThreadX Sample 7

 NetX Duo Simple Ping Sample 8

Overview



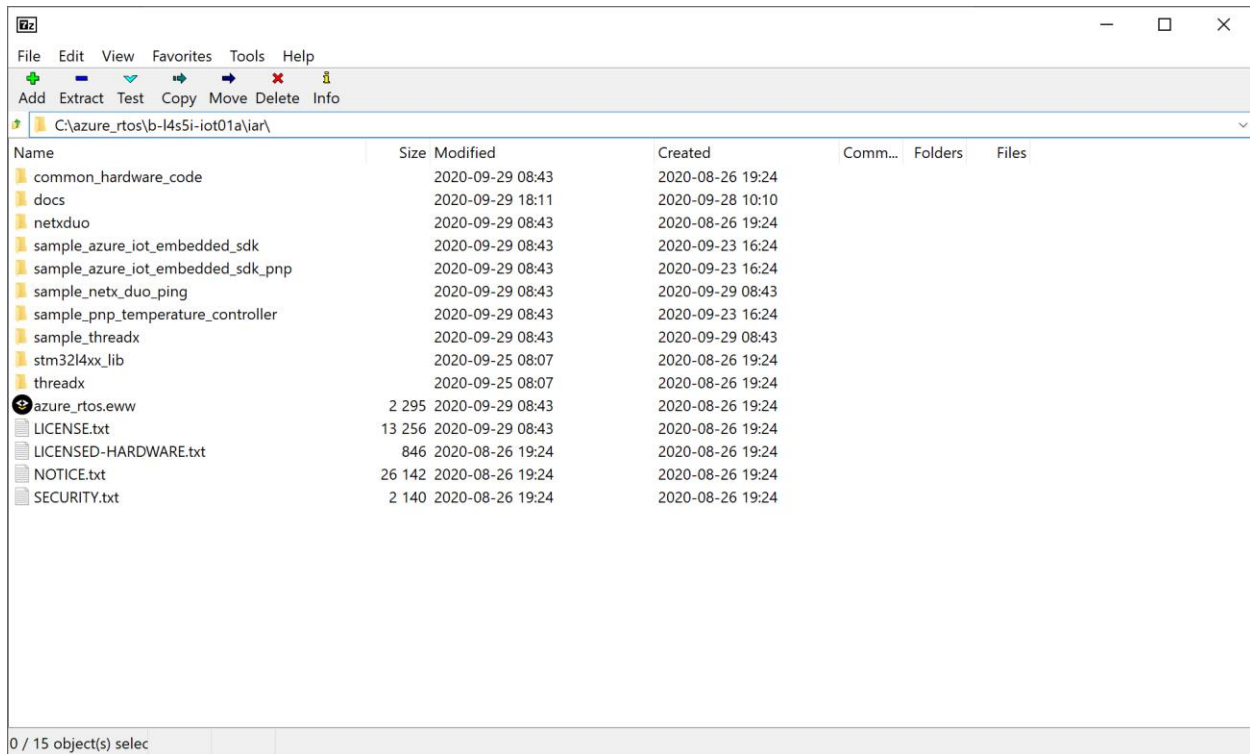
STM32L4S5I-DISCO IoT Node board

Azure RTOS Samples for each component (Azure connectivity, ThreadX and NetX Duo) are designed to run on the STM32L4S5I-DISCO IoT Node “out-of-the-box.” Each sample project is described later in this document along with links to further information as necessary.

All samples are designed to run using the IAR Embedded Workbench for ARM (EWARM) 8.50.1 or later, with the on-board ST-LINK debugger (Debug USB port). The default factory jumper selections are assumed. A 30-day free trial of IAR’s EWARM development tools can be downloaded from this page:

<https://www.iar.com/iar-embedded-workbench/#!?architecture=Arm>

The sample distribution zip file has following organization:



The root directory contains the **azure_rtos.eww** workspace as well as following sub-folders:

Folder	Contents
<i>iar</i>	Contains the following sub-folders as well as the azure_rtos.eww workspace
<i>common_hardware_code</i>	Contains common code for STM32L4S5I-DISCO board
<i>docs</i>	Contains user guides and supporting documentation
<i>netxduo</i>	Contains NetX Duo and NetX Secure source code
<i>sample_azure_iot_embedded_sdk</i>	Sample project to connect to Azure IoT Hub using Azure IoT Middleware for Azure RTOS
<i>sample_azure_iot_embedded_sdk_pnp</i>	Sample project to connect to Azure IoT Hub using Azure IoT Middleware for Azure RTOS via IoT Plug and Play
<i>sample_pnp_temperature_controller</i>	Sample project with IoT Plug and Play using multiple components
<i>sample_netx_duo_ping</i>	Contains NetX Duo ping sample project
<i>sample_threadx</i>	Contains ThreadX sample project
<i>stm32l4xx_lib</i>	Contains STM32L4+ drivers
<i>threadx</i>	Contains ThreadX source code

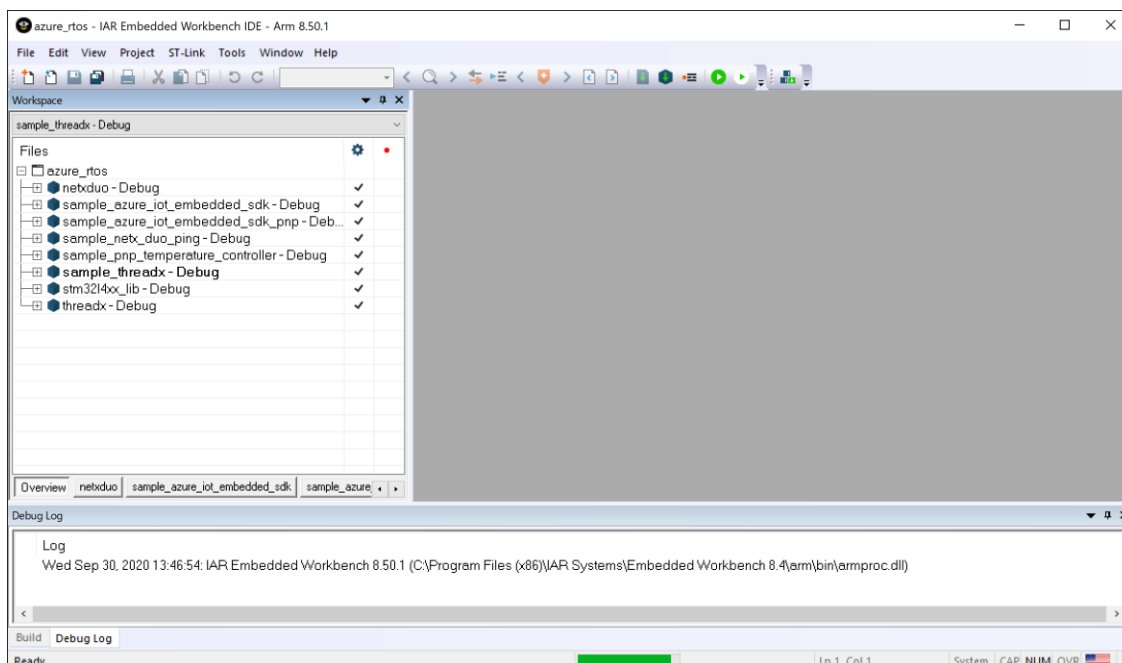
Getting Started

- 1) Unpack the sample zip file into a folder of your choice, we recommend:

C:\azure_rtos\l4s5i-iot01a\iar

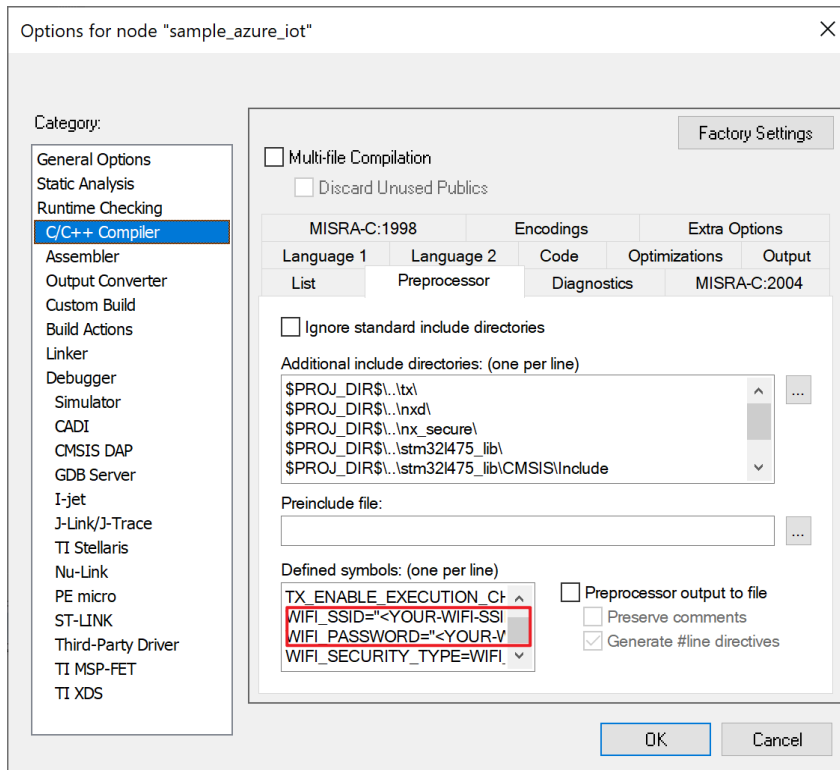
NOTE: If you unzip the IAR project in a very long path, it might not open properly.

- 2) Open the ***azure_rtos.eww*** EWARM Workspace, using IAR EWARM.
- 3) Select the desired sample project in the IAR workspace. There is a menu of available reference project tabs in the Project Pane of the EWARM IDE (as shown below). Click on the desired sample project and make the project active by selecting ***Active Project*** via a right-click.



The ***sample_threadx*** project is shown above as the currently active project.

- 4) Configure the ***WIFI_SSID*** and ***WIFI_PASSWORD*** by right click on active project, select ***Options > C/C++ Compiler > Preprocessor***. Replace the values for your WiFi to be used.



- 5) Select **Project > Batch Build** and choose **build_all** and select **Make** to build all projects. You will observe compilation and linking of all sample projects.
- 6) Select **Download and Debug** to download and start execution of the project. By default, execution stops at a breakpoint set at **main**.
- 7) Select **Go** to start execution of the demonstration. Please review the sample descriptions later in this guide for additional setup and expected behavior.

The sample project can also run on STM32L475-DISCO IoT Node with just a few configurations:

- 1) Right click on active project, select **Options > General Options**. In **Target** tab, change the device to **ST STM32L475VG**.
- 2) In **Options > Linker**, replace the configuration file with **common_hardware_code/stm32l475xx_flash.icf**

Sample Descriptions

Azure RTOS IoT Embedded SDK Samples

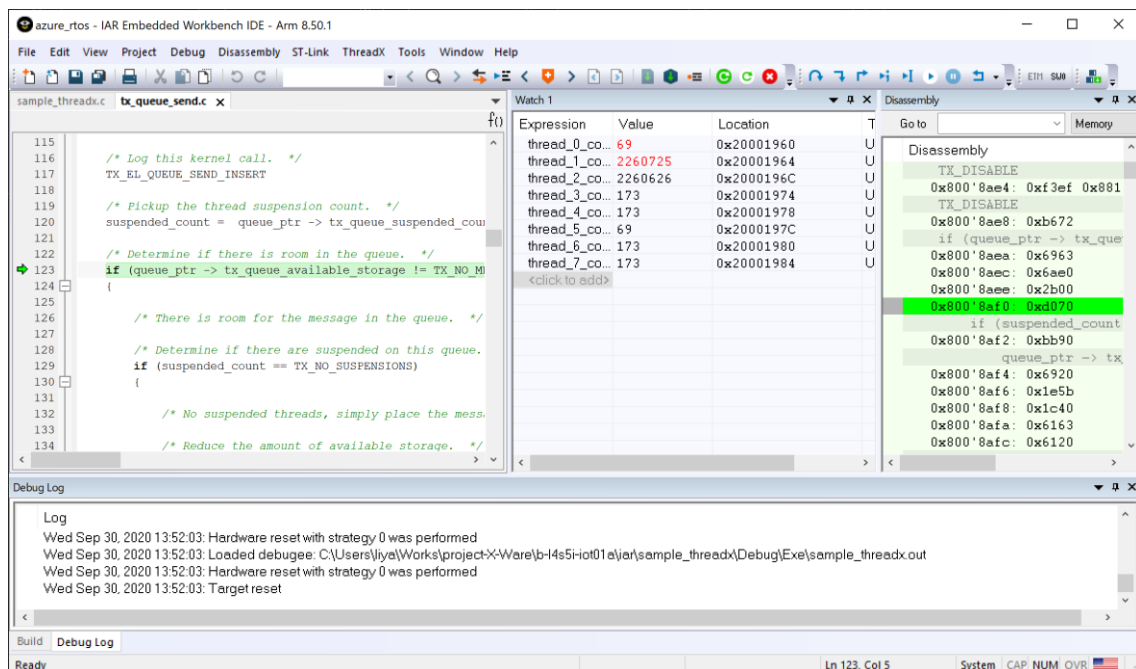
Please see the ***Azure_RTOS_STM32L4+-DISCO_Azure_IoT_Embedded_SDK_For_IAR.pdf*** for a detailed description of the sample as well as a step-by-step set of instructions on how to connect to Azure IoT via the Azure IoT Embedded C SDK and Azure IoT Middleware for Azure RTOS.

ThreadX Sample

This sample is the standard 8-thread ThreadX example, that illustrates the use of the main ThreadX services, including threads, message queues, timers, semaphores, byte memory pools, block memory pools, event flag groups, and mutexes. This demonstration is fully described, including a source code listing, in Chapter 6 of the **Azure_RTOS_ThreadX_User_Guide.pdf**.

To run the ThreadX Sample project, simply follow these steps (assuming the **azure_rtos.eww** workspace is already open):

1. Click on the **sample_threadx** project and make the project active by selecting **Active Project** via a right-click.
2. Select **Make** button to build the **Active Project** selected. You will observe compilation and linking of the selected sample project.
3. Select **Download and Debug** to download and start execution of the demonstration. The sample will initially stop at **main**. Select another **Go** to execute the sample.



After hitting **Break** the IAR debugger screen shot above shows various counters incremented by the ThreadX sample as each of the main components of the ThreadX are exercised.

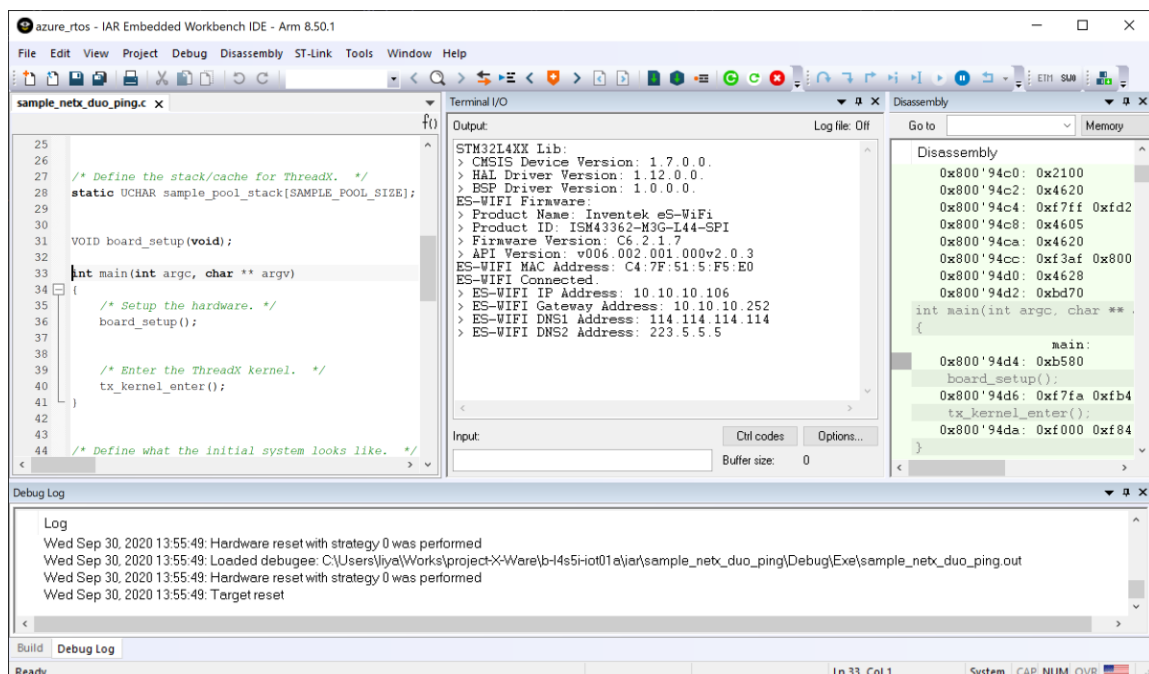
To learn more about Azure RTOS ThreadX, view <https://docs.microsoft.com/azure/rtos/threadx/>.

NetX Duo Simple Ping Sample

This sample project illustrates the setup and use of NetX Duo IPv4/IPv6 TCP/IP stack via ping from another node on the local network. By default, this demonstration requests an IP Address via DHCP, and displays the status and assigned IP Address via Terminal I/O in the IAR debugger. In addition, the retrieved IP address is displayed in the watch window of the IAR debugger.

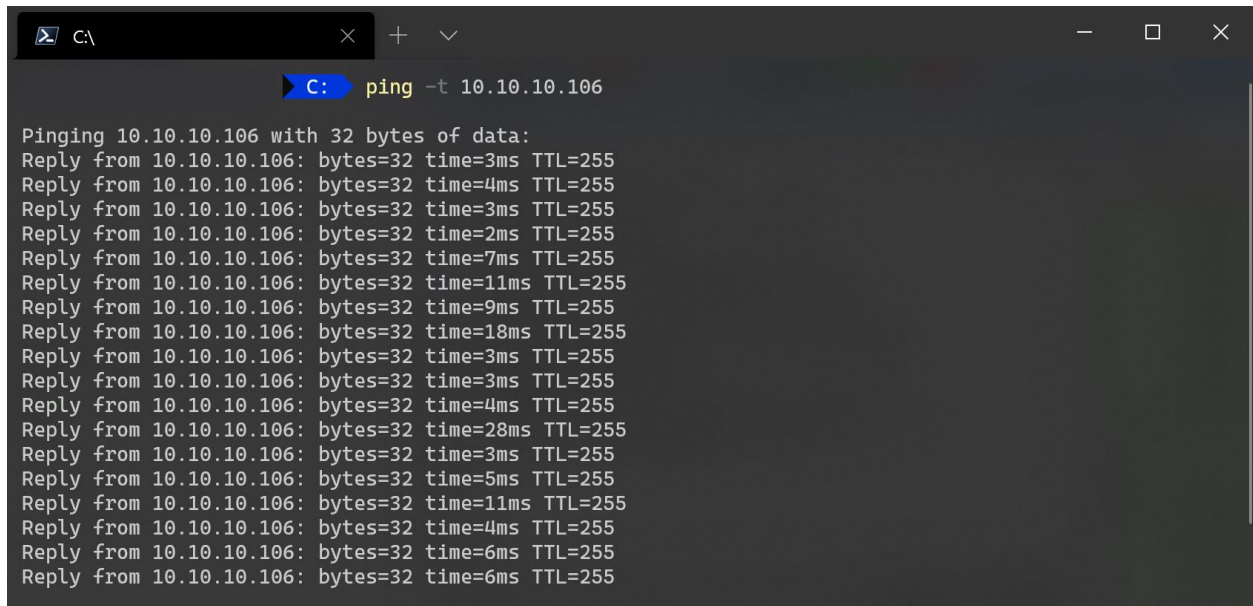
To run the NetX Duo Ping Sample project, simply follow these steps (assuming the **azure_rtos.eww** workspace is already open):

1. Click on the **sample_netx_duo_ping** project and make the project active by selecting **Active Project** via a right-click.
2. Select **Make** button to build the **Active Project** selected. You will observe compilation and linking of the selected sample project. *Note: This sample is Ethernet based and therefore assumes an Ethernet cable is connected to the Ethernet connector on the board.*
3. Select **Download and Debug** to download and start execution of the demonstration. The sample will initially stop at **main**. Select another **Go** to execute the sample. At this point, you should observe the IP address assigned via DHCP in the Terminal I/O window.



The example above shows that the assigned IP address of the STM32L4S5I-DISCO IoT Node board is 10.10.10.106. When the demonstration is running it can be pinged by any machine on

the network. The following is an example of a ping from a Windows machine on the same local network (using the DOS command window):



```
C:\> ping -t 10.10.10.106

Pinging 10.10.10.106 with 32 bytes of data:
Reply from 10.10.10.106: bytes=32 time=3ms TTL=255
Reply from 10.10.10.106: bytes=32 time=4ms TTL=255
Reply from 10.10.10.106: bytes=32 time=3ms TTL=255
Reply from 10.10.10.106: bytes=32 time=2ms TTL=255
Reply from 10.10.10.106: bytes=32 time=7ms TTL=255
Reply from 10.10.10.106: bytes=32 time=11ms TTL=255
Reply from 10.10.10.106: bytes=32 time=9ms TTL=255
Reply from 10.10.10.106: bytes=32 time=18ms TTL=255
Reply from 10.10.10.106: bytes=32 time=3ms TTL=255
Reply from 10.10.10.106: bytes=32 time=3ms TTL=255
Reply from 10.10.10.106: bytes=32 time=4ms TTL=255
Reply from 10.10.10.106: bytes=32 time=28ms TTL=255
Reply from 10.10.10.106: bytes=32 time=3ms TTL=255
Reply from 10.10.10.106: bytes=32 time=5ms TTL=255
Reply from 10.10.10.106: bytes=32 time=11ms TTL=255
Reply from 10.10.10.106: bytes=32 time=4ms TTL=255
Reply from 10.10.10.106: bytes=32 time=6ms TTL=255
Reply from 10.10.10.106: bytes=32 time=6ms TTL=255
```

To learn more about Azure RTOS NetX Duo, view <https://docs.microsoft.com/azure/rtos/netx/>.