Azure RTOS Samples for Microchip SAM E54 Xplained Pro using MPLAB X IDE

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.

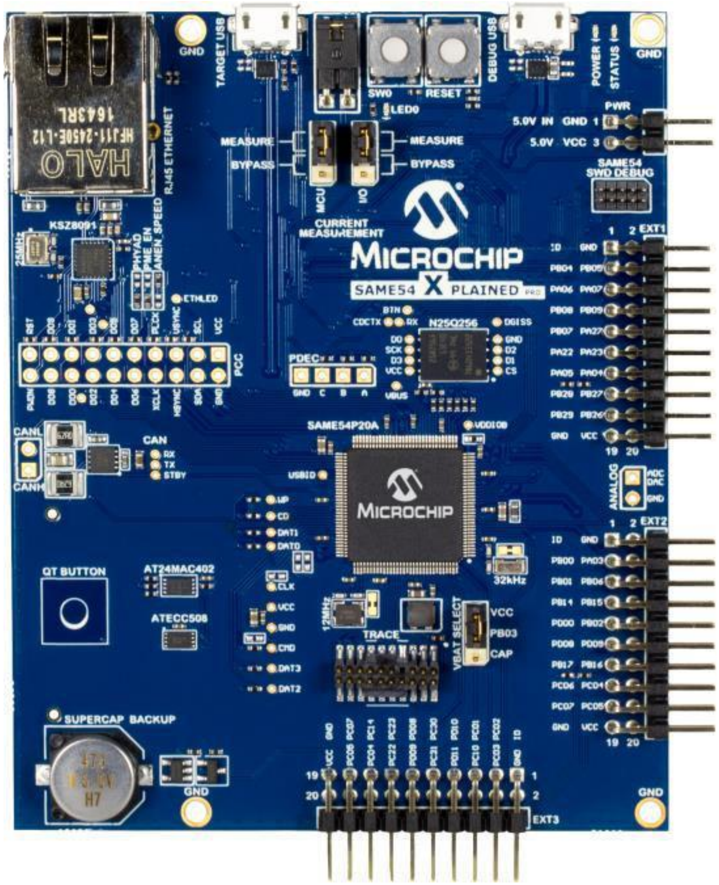
FileX supports the Microsoft exFAT file system format. Your use of exFAT technology in your products requires a separate license from Microsoft. Please see the following link for further details on exFAT licensing:

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

Revision 6.1

Table of Contents

Overview



Microchip SAM E54 Xplained Pro board

Azure RTOS Samples for each component (Azure connectivity, ThreadX, FileX, GUIX, NetX Duo, and USBX) are designed to run on the ATSAME54-XPRO “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 MPLAB X IDE 5.3.5, with the on-board Atmel's Embedded Debugger (EDBG) (Debug USB port). Note that some documentation points to different versions - in particular 5.4.0 version other than 5.3.5 have Dummy_Handler issues in our experience. The default factory jumper selections are assumed. The MPLAB X IDE development tools and MPLAB XC32/32++ Compiler can be downloaded from this page:

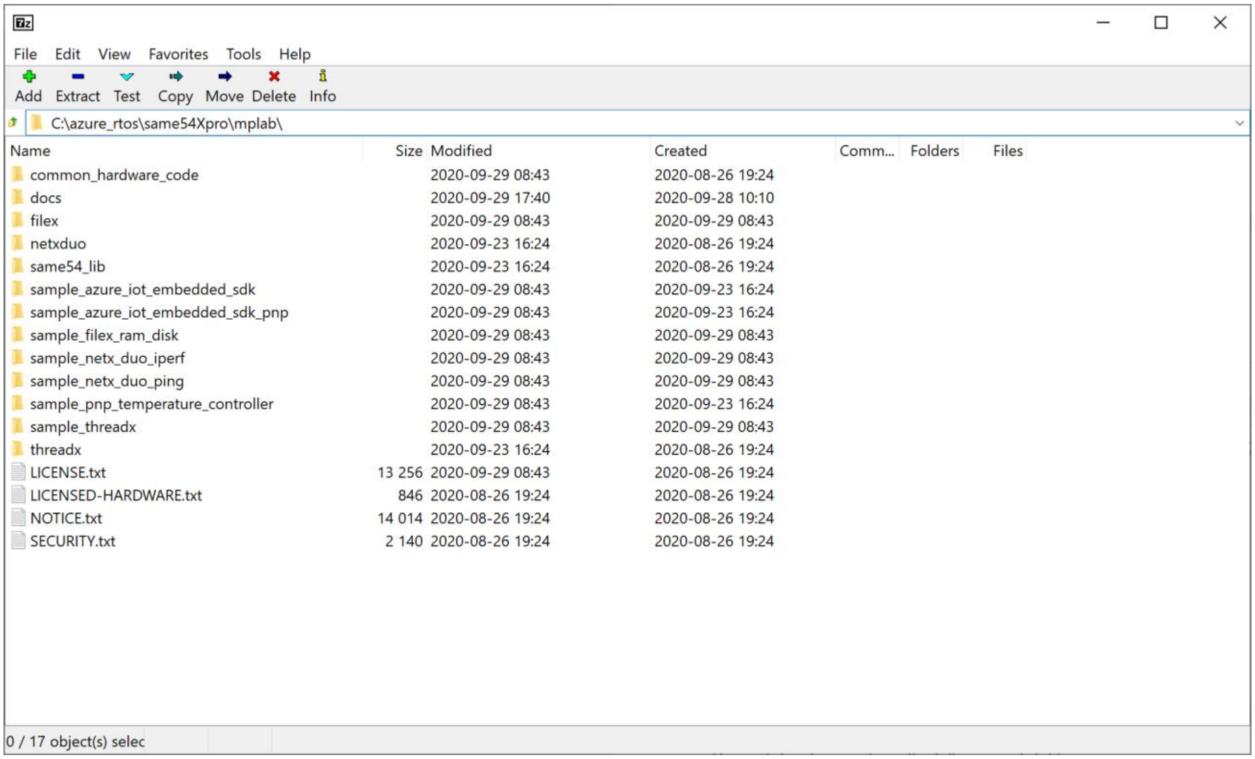
https://www.microchip.com/mplab/mplab-x-ide

You will also need MPLAB XC32/32++ Compiler 2.4.0 or later to build the sample projects. It can be downloaded from this page under compiler download section:

https://www.microchip.com/mplab/compilers

NOTE: On a high DPI monitor or screen, you may find the MPLAB X IDE overall font size is pretty small, follow this FAQ to solve it.

The sample distribution zip file has following organization:



C:\azure_rtos\same54Xpro\mplab\

Name	Size	Modified	Created	Comm...	Folders	Files
common_hardware_code		2020-09-29 08:43	2020-08-26 19:24			
docs		2020-09-29 17:40	2020-09-28 10:10			
filex		2020-09-29 08:43	2020-09-29 08:43			
netxduo		2020-09-23 16:24	2020-08-26 19:24			
same54_lib		2020-09-23 16:24	2020-08-26 19:24			
sample_azure_iot_embedded_sdk		2020-09-29 08:43	2020-09-23 16:24			
sample_azure_iot_embedded_sdk_pnp		2020-09-29 08:43	2020-09-23 16:24			
sample_filex_ram_disk		2020-09-29 08:43	2020-09-29 08:43			
sample_netx_duo_iperf		2020-09-29 08:43	2020-09-29 08:43			
sample_netx_duo_ping		2020-09-29 08:43	2020-09-29 08:43			
sample_pnp_temperature_controller		2020-09-29 08:43	2020-09-23 16:24			
sample_threadx		2020-09-29 08:43	2020-09-29 08:43			
threadx		2020-09-23 16:24	2020-08-26 19:24			
LICENSE.txt	13 256	2020-09-29 08:43	2020-08-26 19:24			
LICENSED-HARDWARE.txt	846	2020-08-26 19:24	2020-08-26 19:24			
NOTICE.txt	14 014	2020-08-26 19:24	2020-08-26 19:24			
SECURITY.txt	2 140	2020-08-26 19:24	2020-08-26 19:24			

0 / 17 object(s) selec

The root directory contains the following sub-folders:

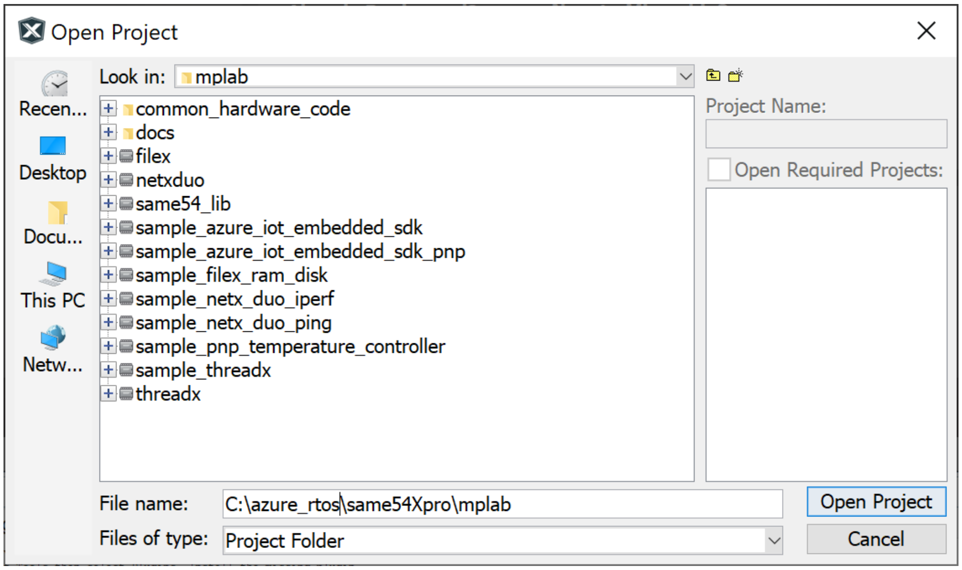
Folder	**Contents**
mplab	Contains the following sub-folders
common_hardware_code	Contains common code for ATSAME54-XPRO board
docs	Contains user guides and supporting documentation
filex	Contains FileX source code
netxduo	Contains NetX Duo and NetX Secure source code
sample_azure_iot_embedded_sdk	Sample project to connect to Azure IoT Hub using Azure IoT Middleware for Azure RTOS
sample_azure_iot_embedded_sdk_pnp	Sample project to connect to Azure IoT Hub using Azure IoT Middleware for Azure RTOS via IoT Plug and Play
sample_pnp_temperature_controller	Sample project with IoT Plug and Play using multiple components
sample_netx_duo_iperf	Contains NetX Duo iPerf sample project
sample_netx_duo_ping	Contains NetX Duo ping sample project
sample_threadx	Contains ThreadX sample project
sample_filex_ram_disk	Contains sample for FileX RAM disk
same54_lib	Contains ATSAME54 drivers
threadx	Contains ThreadX source code

Getting Started

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

   C:\azure_rtos\same54Xpro\mplab

2) Open the project folder using MPLAB X IDE 5.3.5 or later. From ***File > Open Project***. Select all projects, you can use ***Ctrl+A***. Then select ***Open Project***.



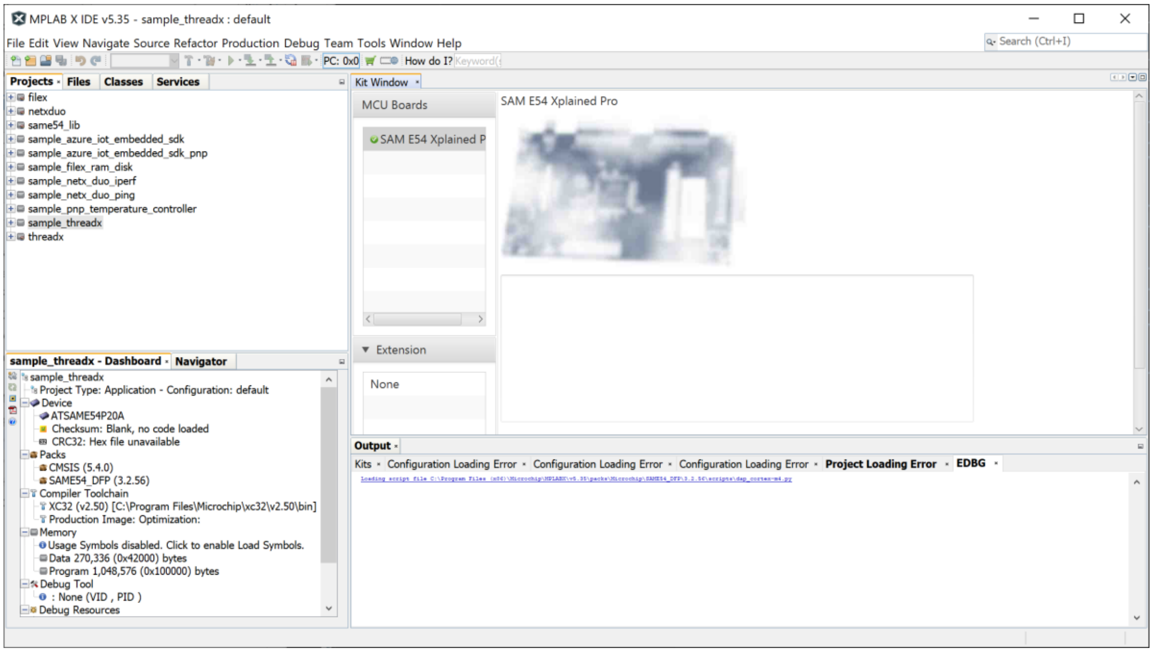
File name: C:\azure_rtos\same54Xpro\mplab

Files of type: Project Folder

3) As sample project has dependency on the Azure RTOS and board libraries. You need to build these libraries first. Select ***threadx*** project and choose ***Build Project (F11)*** from toolbar.

4) Do the same to build the ***filex***, ***netxduo*** and ***same54_lib*** projects.

5) Select the desired sample project from Projects view.


MPLAB X IDE v5.35 - sample_threadx : default
File Edit View Navigate Source Refactor Production Debug Team Tools Window Help
Search (Ctrl+I)
PC: 0x0
How do I?
Keyword(s
Projects
Files
Classes
Services
Kit Window
MCU Boards
SAM E54 Xplained P
SAM E54 Xplained Pro
Extension
None
Output
Kits
Configuration Loading Error
Configuration Loading Error
Configuration Loading Error
Project Loading Error
EDBG

- filex
- netxduo
- same54_lib
- sample_azure_iot_embedded_sdk
- sample_azure_iot_embedded_sdk_pnp
- sample_filex_ram_disk
- sample_netx_duo_iperf
- sample_netx_duo_ping
- sample_pnp_temperature_controller
- sample_threadx
- threadx

sample_threadx - Dashboard | Navigator

```
sample_threadx
  Project Type: Application - Configuration: default
  Device
    ATSAME54P20A
    Checksum: Blank, no code loaded
    CRC32: Hex file unavailable
  Packs
    CMSIS (5.4.0)
    SAME54_DFP (3.2.56)
  Compiler Toolchain
    XC32 (v2.50) [C:\Program Files\Microchip\xc32\v2.50\bin]
    Production Image: Optimization:
  Memory
    Usage Symbols disabled. Click to enable Load Symbols.
    Data 270,336 (0x42000) bytes
    Program 1,048,576 (0x100000) bytes
  Debug Tool
    : None (VID , PID )
  Debug Resources
```

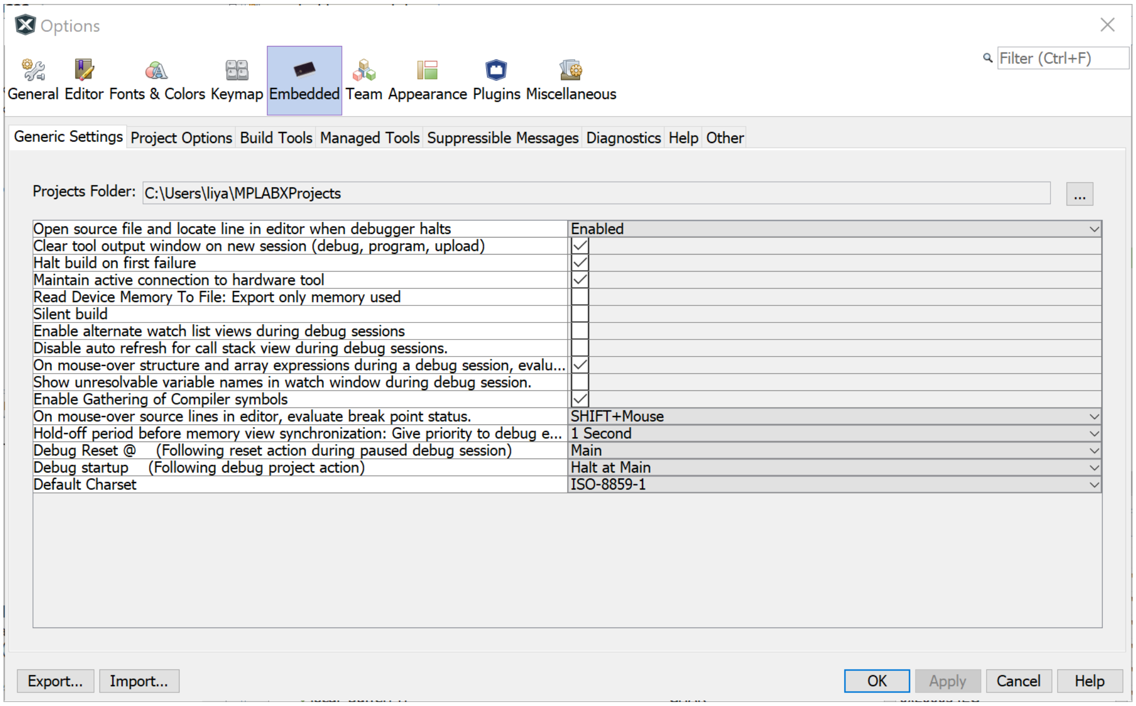
```
Loading script file C:\Program Files (x86)\Microchip\MPLABX\v5.35\packs\Microchip\SAME54_DFP\3.2.56\scripts\dap_cortex-m4.py
```

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

6) Select ***Build Project*** to build the active project selected. You will observe compilation and linking of the selected sample project.

7) Select ***Debug Project*** to download and start execution of the project. Please review the sample descriptions later in this guide for additional setup and expected behavior.

   Note: Make sure you set the ***Tools > Options > Embedded > Generic Settings > Debug startup*** as ***Halt at Main***. The application can be stopped at main then.



Projects Folder: C:\Users\liya\MPLABXProjects

Open source file and locate line in editor when debugger halts	Enabled
Clear tool output window on new session (debug, program, upload)	☑
Halt build on first failure	☑
Maintain active connection to hardware tool	☑
Read Device Memory To File: Export only memory used	☐
Silent build	☐
Enable alternate watch list views during debug sessions	☐
Disable auto refresh for call stack view during debug sessions.	☐
On mouse-over structure and array expressions during a debug session, evalu...	☑
Show unresolvable variable names in watch window during debug session.	☐
Enable Gathering of Compiler symbols	☑
On mouse-over source lines in editor, evaluate break point status.	SHIFT+Mouse
Hold-off period before memory view synchronization: Give priority to debug e...	1 Second
Debug Reset @ (Following reset action during paused debug session)	Main
Debug startup (Following debug project action)	Halt at Main
Default Charset	ISO-8859-1

Sample Descriptions

Azure RTOS IoT Embedded SDK Samples

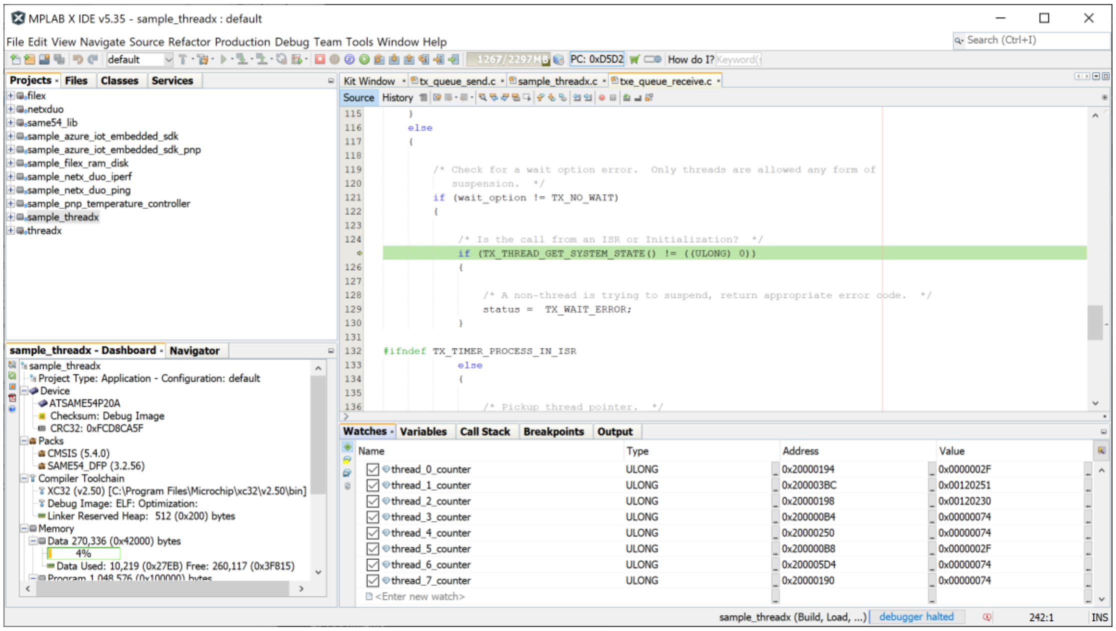
Please see the ***Azure_RTOS_ATSAME54-XPRO_Azure_IoT_Embedded_SDK_For_MPLAB.pdf*** for a detailed description of the sample as well as a step-by-step set of instructions on how to connect to Azure IoT via the Azure IoT Embedded C SDK and Azure IoT Middleware for Azure RTOS.

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 workspace is already open):

1. Click on the ***sample_threadx*** project to make the project.
2. Select ***Build Project (F11)*** button to build the project selected. You will observe compilation and linking of the selected sample project.
3. Select ***Debug Project*** to download and start execution of the demonstration.



```
        }
        else
        {

            /* Check for a wait option error.  Only threads are allowed any form of
               suspension.  */
            if (wait_option != TX_NO_WAIT)
            {

                /* Is the call from an ISR or Initialization?  */
                if (TX_THREAD_GET_SYSTEM_STATE() != ((ULONG) 0))
                {

                    /* A non-thread is trying to suspend, return appropriate error code.  */
                    status =  TX_WAIT_ERROR;
                }

#ifndef TX_TIMER_PROCESS_IN_ISR
                else
                {

                    /* Pickup thread pointer.  */
```

- sample_threadx
  - Project Type: Application - Configuration: default
  - Device
    - ATSAME54P20A
    - Checksum: Debug Image
    - CRC32: 0xFCD8CA5F
  - Packs
    - CMSIS (5.4.0)
    - SAME54_DFP (3.2.56)
  - Compiler Toolchain
    - XC32 (v2.50) [C:\Program Files\Microchip\xc32\v2.50\bin]
    - Debug Image: ELF: Optimization:
    - Linker Reserved Heap: 512 (0x200) bytes
  - Memory
    - Data 270,336 (0x42000) bytes
      - 4%
      - Data Used: 10,219 (0x27EB) Free: 260,117 (0x3F815)
    - Program 1,048,576 (0x100000) bytes

Name	Type	Address	Value
☑ thread_0_counter	ULONG	0x20000194	0x0000002F
☑ thread_1_counter	ULONG	0x200003BC	0x00120251
☑ thread_2_counter	ULONG	0x20000198	0x00120230
☑ thread_3_counter	ULONG	0x200000B4	0x00000074
☑ thread_4_counter	ULONG	0x20000250	0x00000074
☑ thread_5_counter	ULONG	0x200000B8	0x0000002F
☑ thread_6_counter	ULONG	0x200005D4	0x00000074
☑ thread_7_counter	ULONG	0x20000190	0x00000074
<Enter new watch>			

After hitting ***Pause*** ⏸ the ***Watches*** window (Window > Debugging > Watches) in screen shot above shows various counters incremented by the ThreadX sample as each of the main components of the ThreadX are exercised. You can further add expression such as "***thread_x_counter***" in the ***Watches*** window.

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

FileX RAM Disk Sample

This sample illustrates the use of the FileX embedded FAT file system. The example creates a small RAM-disk with a sample file and data, and reads the file data back into memory. The debugger is able to show the data being read.

To run the FileX RAM Disk Sample project, simply follow these steps (assuming the workspace is already open):

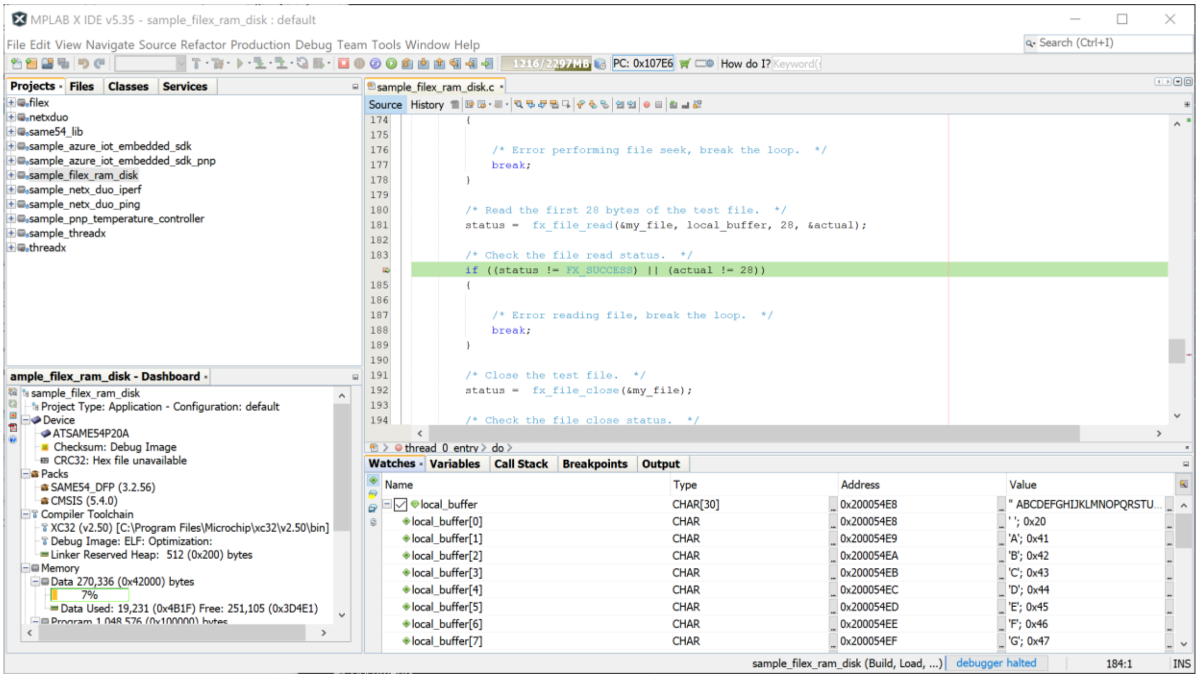
1. Click on the ***sample_filex_ram_disk*** project and make the project.

2. Open ***sample_filex_ram_disk.c*** and set a breakpoint around Line 194 at *if (status != FX_SUCCESS)*

3. Select ***Build Project*** button to build the project selected. You will observe compilation and linking of the selected sample project.

4. Select ***Debug Project*** to download and start execution of the demonstration to execute the RAM disk sample to the point after the file read is complete.

5. In the Watch window, ensure you watch the ***local_buffer*** variable as expression.



- filex
- netxduo
- same54_lib
- sample_azure_iot_embedded_sdk
- sample_azure_iot_embedded_sdk_pnp
- sample_filex_ram_disk
- sample_netx_duo_iperf
- sample_netx_duo_ping
- sample_pnp_temperature_controller
- sample_threadx
- threadx

```
174             {
175
176                 /* Error performing file seek, break the loop.  */
177                 break;
178             }
179
180             /* Read the first 28 bytes of the test file.  */
181             status =  fx_file_read(&my_file, local_buffer, 28, &actual);
182
183             /* Check the file read status.  */
                if ((status != FX_SUCCESS) || (actual != 28))
185             {
186
187                 /* Error reading file, break the loop.  */
188                 break;
189             }
190
191             /* Close the test file.  */
192             status =  fx_file_close(&my_file);
193
194             /* Check the file close status.  */
```

ample_filex_ram_disk - Dashboard

```
sample_filex_ram_disk
  Project Type: Application - Configuration: default
  Device
    ATSAME54P20A
    Checksum: Debug Image
    CRC32: Hex file unavailable
  Packs
    SAME54_DFP (3.2.56)
    CMSIS (5.4.0)
  Compiler Toolchain
    XC32 (v2.50) [C:\Program Files\Microchip\xc32\v2.50\bin]
    Debug Image: ELF: Optimization:
    Linker Reserved Heap:  512 (0x200) bytes
  Memory
    Data 270,336 (0x42000) bytes
      7%
      Data Used: 19,231 (0x4B1F) Free: 251,105 (0x3D4E1)
    Program 1,048,576 (0x100000) bytes
```

Name	Type	Address	Value
☑ local_buffer	CHAR[30]	0x200054E8	" ABCDEFGHIJKLMNOPQRSTU...
local_buffer[0]	CHAR	0x200054E8	' '; 0x20
local_buffer[1]	CHAR	0x200054E9	'A'; 0x41
local_buffer[2]	CHAR	0x200054EA	'B'; 0x42
local_buffer[3]	CHAR	0x200054EB	'C'; 0x43
local_buffer[4]	CHAR	0x200054EC	'D'; 0x44
local_buffer[5]	CHAR	0x200054ED	'E'; 0x45
local_buffer[6]	CHAR	0x200054EE	'F'; 0x46
local_buffer[7]	CHAR	0x200054EF	'G'; 0x47

The ***Watches*** window in the screen shot above shows the file data read back in the RAM disk sample.

To learn more about Azure RTOS FileX, view https://docs.microsoft.com/azure/rtos/filex/.

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 output.

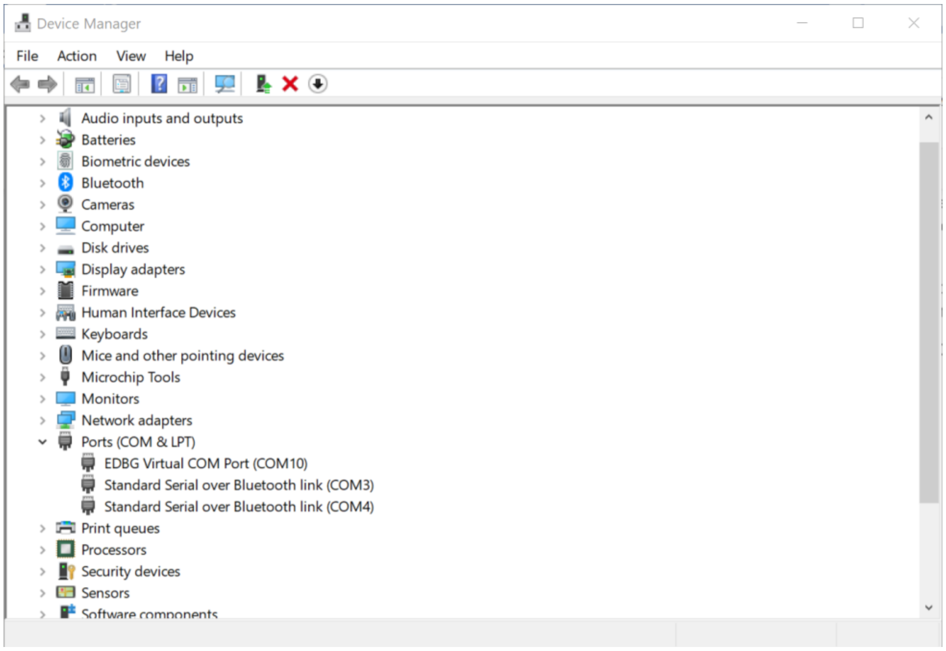
To run the NetX Duo Ping Sample project, simply follow these steps (assuming the workspace is already open):

1. Click on the ***sample_netx_duo_ping*** project and make the project active.

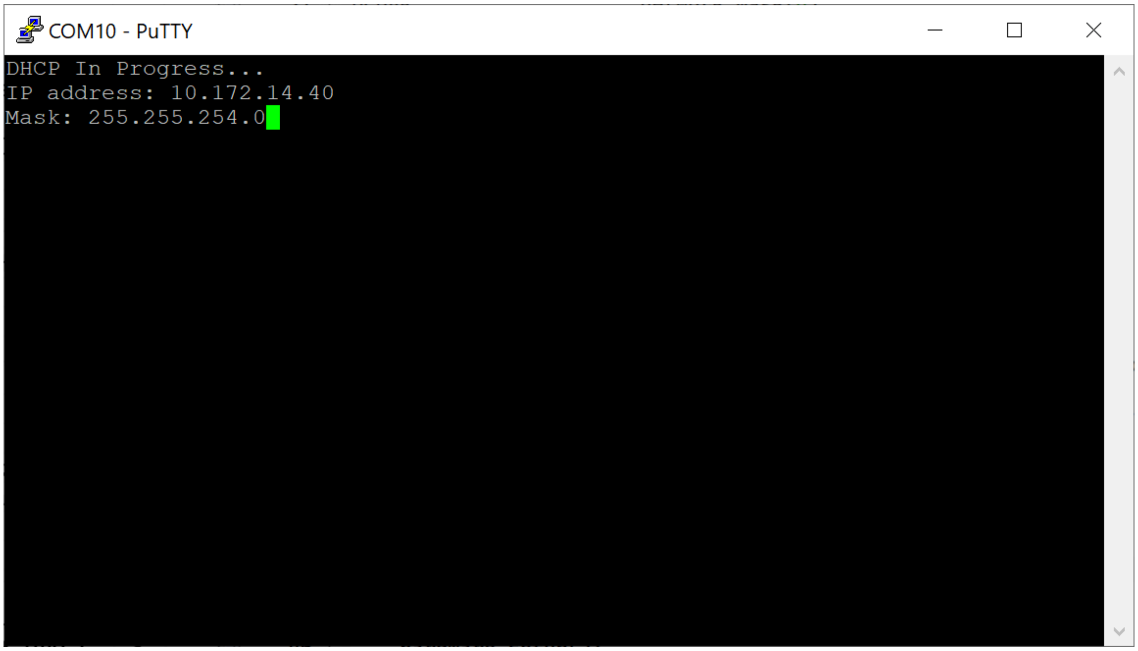
2. Select ***Build Project*** button to build the project selected. You will observe compilation and linking of the selected sample project. *Note: This sample is Ethernet based and therefore assumes an Ethernet cable is connected to the Ethernet connector on the board.*

3. Select ***Debug Project*** to download and start execution of the demonstration.

4. Verify the serial port in your OS's device manager. It should show up as a COM port.

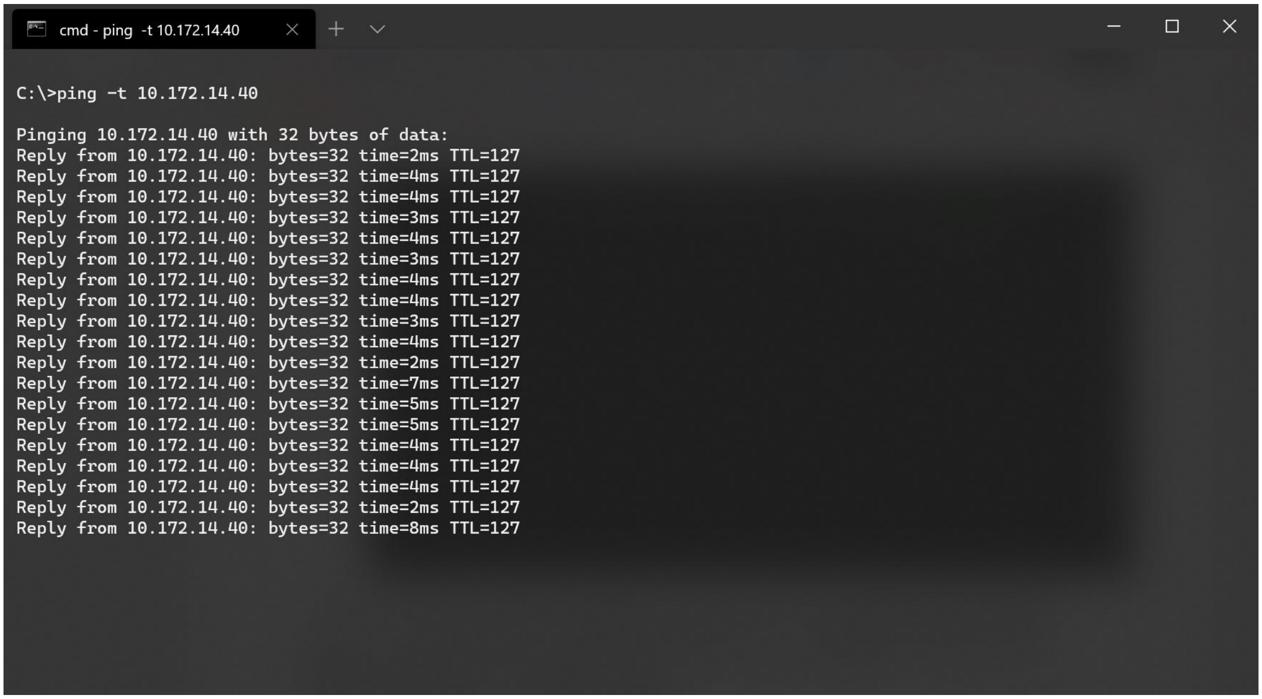


5. Open your favorite serial terminal program such as Putty and connect to the COM port discovered above. As the project runs you should observe the IP address assigned via DHCP in the output window.



```
DHCP In Progress...
IP address: 10.172.14.40
Mask: 255.255.254.0
```

The example above shows that the assigned IP address of the ATSAME54-XPRO board is 10.172.14.40. 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 netword (using the DOS command window):



```
C:\>ping -t 10.172.14.40

Pinging 10.172.14.40 with 32 bytes of data:
Reply from 10.172.14.40: bytes=32 time=2ms TTL=127
Reply from 10.172.14.40: bytes=32 time=4ms TTL=127
Reply from 10.172.14.40: bytes=32 time=4ms TTL=127
Reply from 10.172.14.40: bytes=32 time=3ms TTL=127
Reply from 10.172.14.40: bytes=32 time=4ms TTL=127
Reply from 10.172.14.40: bytes=32 time=3ms TTL=127
Reply from 10.172.14.40: bytes=32 time=4ms TTL=127
Reply from 10.172.14.40: bytes=32 time=4ms TTL=127
Reply from 10.172.14.40: bytes=32 time=3ms TTL=127
Reply from 10.172.14.40: bytes=32 time=4ms TTL=127
Reply from 10.172.14.40: bytes=32 time=2ms TTL=127
Reply from 10.172.14.40: bytes=32 time=7ms TTL=127
Reply from 10.172.14.40: bytes=32 time=5ms TTL=127
Reply from 10.172.14.40: bytes=32 time=5ms TTL=127
Reply from 10.172.14.40: bytes=32 time=4ms TTL=127
Reply from 10.172.14.40: bytes=32 time=4ms TTL=127
Reply from 10.172.14.40: bytes=32 time=4ms TTL=127
Reply from 10.172.14.40: bytes=32 time=2ms TTL=127
Reply from 10.172.14.40: bytes=32 time=8ms TTL=127
```

Note: Static IP address assignment is also possible by disabling ***NX_ENABLE_DHCP*** *in the project settings and modifying the default static IP address of* ***192.2.2.149*** *in the source file* ***demo_netx_duo_ping.c.***

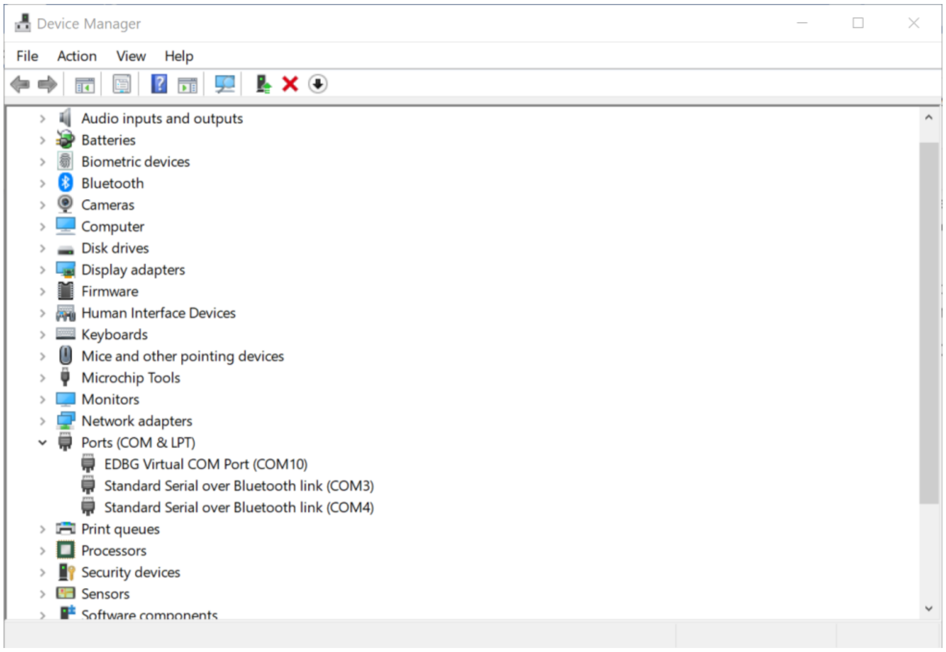
To learn more about Azure RTOS NetX Duo, view https://docs.microsoft.com/azure/rtos/netx/.

NetX Duo Iperf Throughput Sample

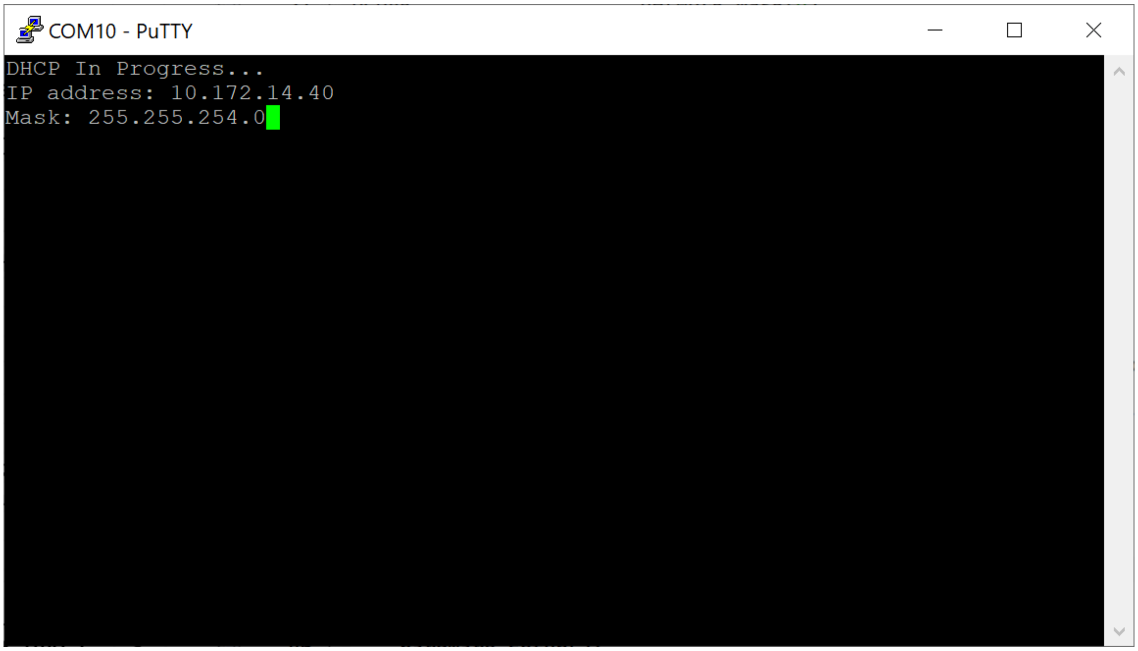
This demonstration illustrates TCP and UDP network throughput, using Express Logic's NetX Duo IPv4/IPv6 TCP/IP stack, and the industry-standard Iperf network throughput benchmark, with Jperf GUI. By default, this demonstration requests an IP Address via DHCP, and displays the status and assigned IP Address via Terminal output.

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

1. Click on the ***sample_netx_duo_iperf*** project and make the project active.
2. Select ***Build Project*** button to build the 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 ***Debug Project*** to download and start execution of the demonstration.
4. Verify the serial port in your OS's device manager. It should show up as a COM port.

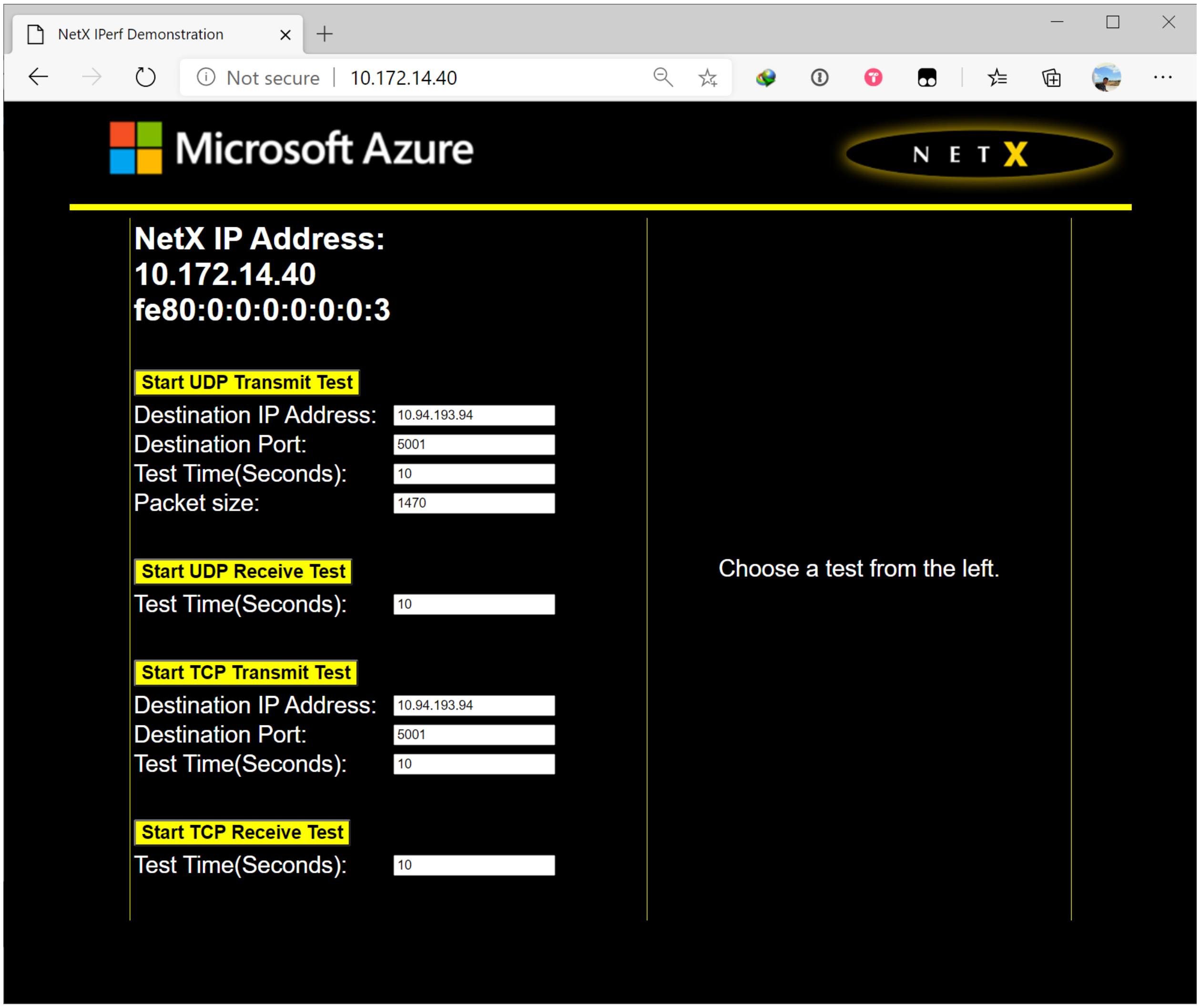


5. Open your favorite serial terminal program such as Putty and connect to the COM port discovered above. As the project runs you should observe the IP address assigned via DHCP in the output window.



```
DHCP In Progress...
IP address: 10.172.14.40
Mask: 255.255.254.0
```

Once running, simply browse to target IP address (in the screen shot above it is 10.172.14.40) to view the NetX Duo Iperf server page, which provides options for running each Iperf test as well as displays the results of each test. Here is as sample view after browsing 10.172.14.40:



NetX IP Address:
10.172.14.40
fe80:0:0:0:0:0:0:3

Destination IP Address: 10.94.193.94
Destination Port: 5001
Test Time(Seconds): 10
Packet size: 1470

Test Time(Seconds): 10

Destination IP Address: 10.94.193.94
Destination Port: 5001
Test Time(Seconds): 10

Test Time(Seconds): 10

Choose a test from the left.

You will now need to setup and run Jperf on a Windows host on the same local network. To learn how to use the Jperf with Iperf sample, view ***Azure_RTOS_NetX_Duo_Iperf_User_Guide.pdf***.

Note: Static IP address assignment is also possible by disabling ***NX_ENABLE_DHCP*** *in the project settings and modifying the default static IP address of* ***192.2.2.149*** *in the source file* ***sample_netx_duo_iperf.c.***

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