Microsoft

Device Update Sample i.MX RT1060 EVK using MCUXpresso IDE

User Guide

Published: March 2021

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

Revision 6.1

Table of Contents

Overview

The following steps detail how to configure, build and execute the Device Update for IoT Hub (public preview) example on the i.MX RT1060 EVK. For this sample, we will build a new firmware that will be uploaded to the Device Update for IoT Hub and deployed to the device.

The tool uses in the sample is MCUXpresso IDE 11.1.1 or later development tools. It can be downloaded free from this page: https://www.nxp.com/design/software/development-software/mcuxpresso-software-and-tools/mcuxpresso-integrated-development-environment-ide:MCUXpresso-IDE

You will also need MIMXRT1060-EVK SDK 2.7.0 or later to build the sample projects. It can be downloaded from this page: https://mcuxpresso.nxp.com/en/builder

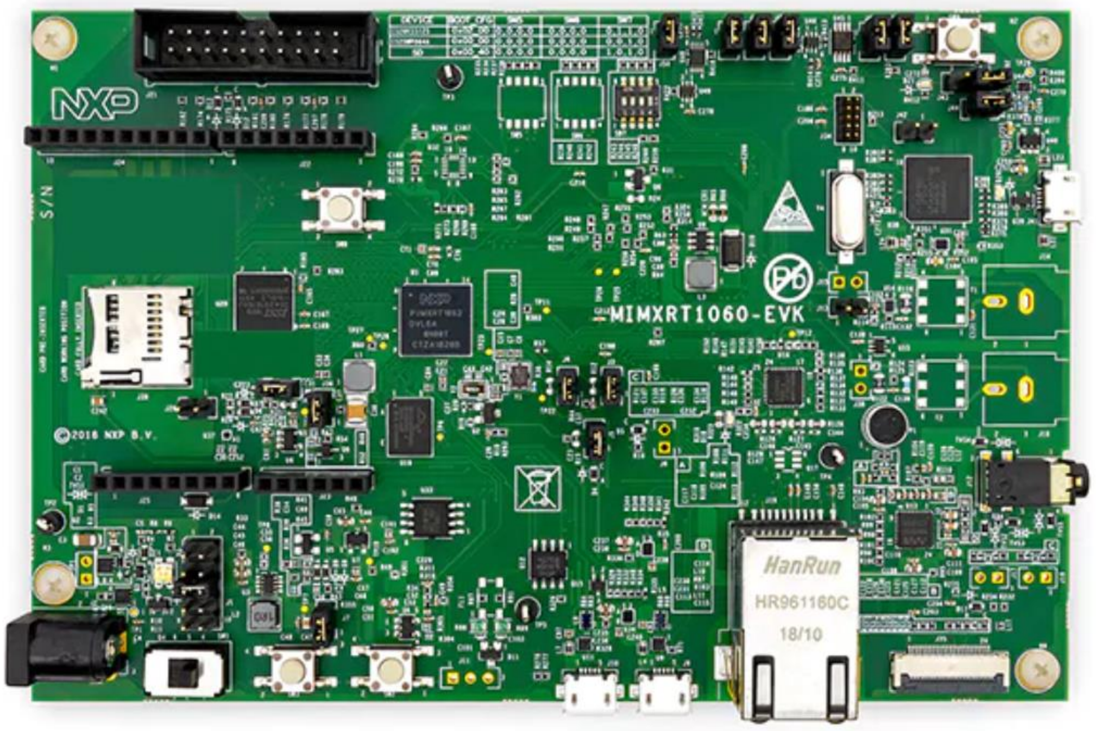


Figure 1 i.MX RT1060 EVK

The sample distribution zip file contains the following sub-folders:

Folder	**Contents**
bootloader	Bootloader binary
common_hardware_code	Common code for i.MX RT1060 EVK board
docs	User guides
netxduo	NetX Duo source code
sample_azure_iot_embedded_sdk_adu	Sample project to connect to Azure IoT Hub using Azure IoT Middleware for Azure RTOS
mimxrt1060_library	i.MX RT1060 drivers
threadx	ThreadX source code
filex	FileX source code

Prepare Azure Resources

To prepare Azure cloud resources and connect a device to Azure, you can use Azure CLI. There are two ways to access the Azure CLI: by using the Azure Cloud Shell, or by installing Azure CLI locally. Azure Cloud Shell lets you run the CLI in a browser, so you don't have to install anything.

Use one of the following options to run Azure CLI.

If you prefer to run Azure CLI locally:

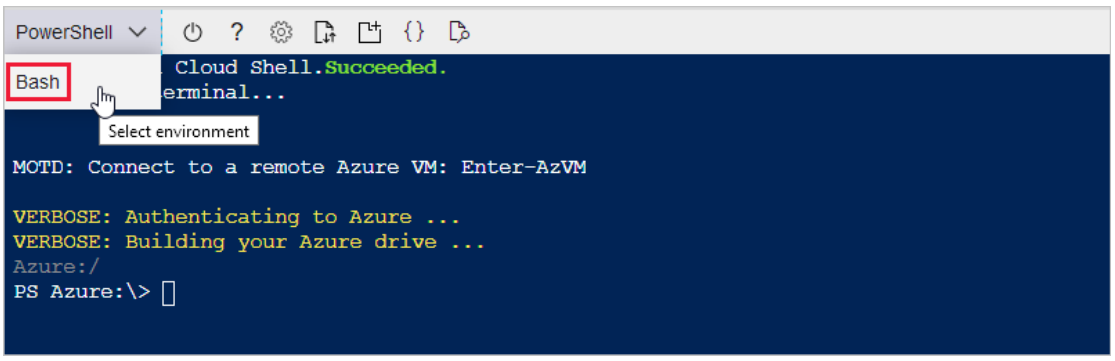
1. If you already have Azure CLI installed locally, run az --version to check the version. This tutorial requires Azure CLI 2.5.1 or later.
2. To install or upgrade, see Install Azure CLI. If you install Azure CLI locally, you can run CLI commands in the GCC Command Prompt, Git Bash for Windows, or Powershell.

If you prefer to run Azure CLI in the browser-based Azure Cloud Shell:

1. Use your Azure account credentials to sign into the Azure Cloud shell at https://shell.azure.com/.

   Note: If this is the first time you've used the Cloud Shell, it prompts you to create storage, which is required to use the Cloud Shell. Select a subscription to create a storage account and Microsoft Azure Files share.

2. Select Bash or Powershell as your preferred CLI environment in the Select environment dropdown. If you plan to use Azure Cloud Shell, keep your browser open to run the Azure CLI commands in this tutorial.



```
 Cloud Shell.Succeeded.
erminal...

MOTD: Connect to a remote Azure VM: Enter-AzVM

VERBOSE: Authenticating to Azure ...
VERBOSE: Building your Azure drive ...
Azure:/
PS Azure:\>
```

Create an IoT Hub

You can use Azure CLI to create an IoT hub that handles events and messaging for your device.

To create an IoT hub:

1. In your CLI console, run the `az extension add` command to add the Microsoft Azure IoT Extension for Azure CLI to your CLI shell. The IOT Extension adds IoT Hub, IoT Edge, and IoT Device Provisioning Service (DPS) specific commands to Azure CLI.

   ```
   az extension add --name azure-iot
   ```

2. Run the `az group create` command to create a resource group. The following command creates a resource group named ***MyResourceGroup*** in the ***eastus*** region.

   Note: Optionally, to set an alternate ***location***, run `az account list-locations` to see available locations. Then specify the alternate location in the following command in place of eastus.

   ```
   az group create --name MyResourceGroup --location eastus
   ```

3. Run the `az iot hub create` command to create an IoT hub. It might take a few minutes to create an IoT hub.

   YourIotHubName. Replace this placeholder below with the name you chose for your IoT hub. An IoT hub name must be globally unique in Azure. This placeholder is used in the rest of this tutorial to represent your unique IoT hub name.

   ```
   az iot hub create --resource-group MyResourceGroup --name
   {YourIoTHubName}
   ```

4. After the IoT hub is created, view the JSON output in the console, and copy the ***hostName*** value to a safe place. You use this value in a later step. The ***hostName*** value looks like the following example:

   ```
   {Your IoT hub name}.azure-devices.net
   ```

Register an IoT Hub device

In this section, you create a new device instance and register it with the Iot hub you created. You will use the connection information for the newly registered device to securely connect your physical device in a later section.

To register a device:

1. In your console, run the `az iot hub device-identity create` command. This creates the simulated device identity.

   YourIotHubName. Replace this placeholder below with the name you chose for your IoT hub.

   MyDevKit. You can use this name directly for the device in CLI commands in this tutorial. Optionally, use a different name.

   ```
   az iot hub device-identity create --device-id MyDevKit --hub-name {YourIoTHubName}
   ```

2. After the device is created, view the JSON output in the console, and copy the ***deviceId*** and ***primaryKey*** values to use in a later step.

Confirm that you have the copied the following values from the JSON output to use in the next section:

- ***hostName***
- ***deviceId***
- ***primaryKey***

Create Device Update account

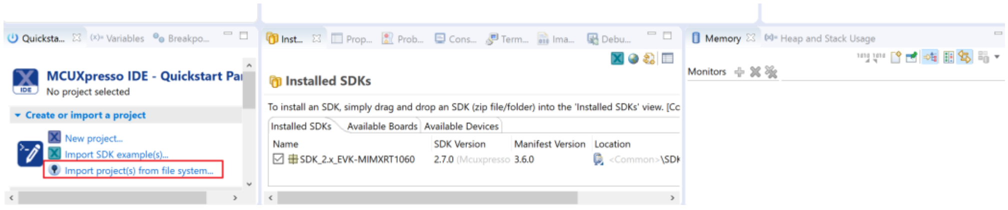
Follow this guide to create a device update account using Azure portal: https://docs.microsoft.com/azure/iot-hub-device-update/create-device-update-account

Prepare the new firmware

To connect the device to Azure, you'll modify a configuration file for Azure IoT settings, build and flash the image to the device.

Build new firmware

1. Launch MCUXpresso IDE, select ***Import project(s) from file system*** from ***Quickstart*** pane and select the ***MCUXpresso*** folder from the extracted zip file.



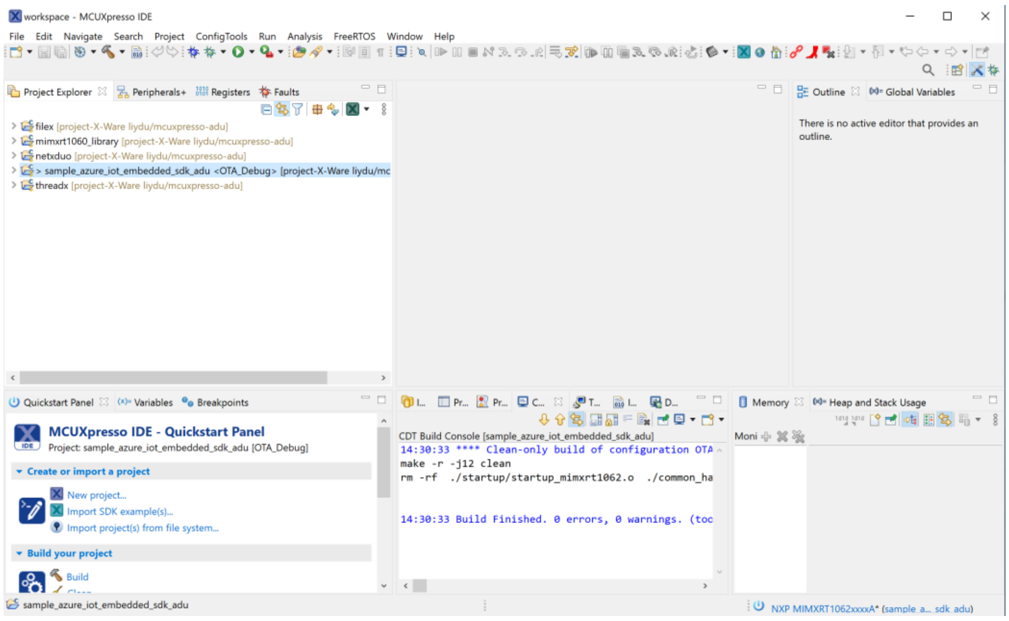
Installed SDKs

To install an SDK, simply drag and drop an SDK (zip file/folder) into the 'Installed SDKs' view. [Cc

Name	SDK Version	Manifest Version	Location
☑ SDK_2.x_EVK-MIMXRT1060	2.7.0 (Mcuxpresso	3.6.0	<Common>\SDK

MCUXpresso IDE - Quickstart Panel
No project selected
Create or import a project
New project...
Import SDK example(s)...
Import project(s) from file system...

2. Import all projects, uncheck the ***Copy projects into workspace*** and select ***Finish***.



```
filex [project-X-Ware liydu/mcuxpresso-adu]
mimxrt1060_library [project-X-Ware liydu/mcuxpresso-adu]
netxduo [project-X-Ware liydu/mcuxpresso-adu]
> sample_azure_iot_embedded_sdk_adu <OTA_Debug> [project-X-Ware liydu/mc
threadx [project-X-Ware liydu/mcuxpresso-adu]
```

There is no active editor that provides an outline.

MCUXpresso IDE - Quickstart Panel
Project: sample_azure_iot_embedded_sdk_adu [OTA_Debug]

```
CDT Build Console [sample_azure_iot_embedded_sdk_adu]
14:30:33 **** Clean-only build of configuration OTA
make -r -j12 clean
rm -rf  ./startup/startup_mimxrt1062.o  ./common_ha


14:30:33 Build Finished. 0 errors, 0 warnings. (toc
```

workspace - MCUXpresso IDE
File Edit Navigate Search Project ConfigTools Run Analysis FreeRTOS Window Help
Project Explorer
Peripherals+
Registers
Faults
Outline
Global Variables
Quickstart Panel
Variables
Breakpoints
Create or import a project
New project...
Import SDK example(s)...
Import project(s) from file system...
Build your project
Build
Memory
Heap and Stack Usage
Moni
sample_azure_iot_embedded_sdk_adu
NXP MIMXRT1062xxxxA* (sample_a... sdk_adu)

1. Select the ***sample_azure_iot_embedded_sdk_adu*** sample.
2. Expand the sample folder to open ***sample_config.h*** to set the Azure IoT device information constants to the values that you saved after you created Azure resources.

Constant name	Value
`HOST_NAME`	{Your IoT hub hostName value}
`DEVICE_ID`	{Your deviceID value}
`DEVICE_SYMMETRIC_KEY`	{Your primaryKey value}

3. In the same folder, open ***sample_azure_iot_embedded_sdk_adu.c,*** modify the firmware version to mimic it is a new firmware that will be deployed from Device Update.

```
#define SAMPLE_UPDATE_ID_VERSION
"7.0.0"
```

4. Select ***Project > Build*** Project or ***Build*** on the toolbar. You will observe compilation and linking of sample project.
5. Create a folder for putting the update firmware and manifest that you will later upload to ADU. For example, `%USERPROFILE%\Documents\new_firmware`
6. Copy the generated binary file from `sample_azure_iot_embedded_sdk_adu\Debug\Exe\sample.bin` to the folder you just created. And rename it to ***firmware_7.0.0.bin***.

Now you have prepared the update firmware that will be uploaded to Device Update for IoT Hub.

Generate import manifest

An import manifest is a JSON file that defines important information about the update that you are importing that is required by the Device Update for IoT Hub. You can learn the detailed steps about importing new update from here. For this sample:

1. Ensure you have installed PowerShell v7.0 or above.
2. Clone Device Update for IoT Hub repository, or download it as a .zip file to a location accessible from PowerShell.
3. In PowerShell, navigate to ***tools/AduCmdlets*** directory and run:

   ```
   Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope
   Process

   Import-Module .\AduUpdate.psm1
   ```

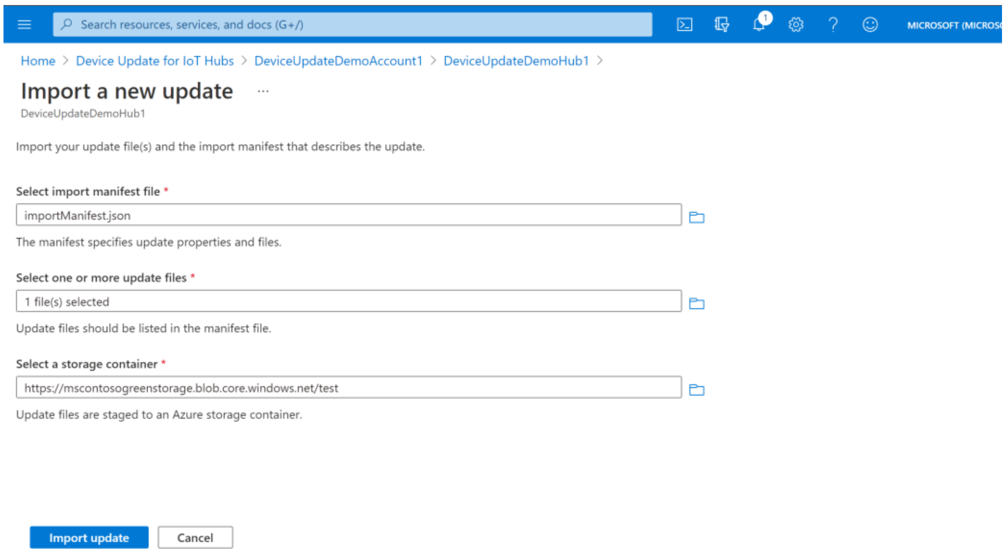
4. Run the following commands to generate an import manifest, a JSON file that describes the update:

   ```
   $compat = New-AduUpdateCompatibility -DeviceManufacturer
   'NXP' -DeviceModel 'RT1060'

   $importManifest = New-AduImportManifest -Provider 'NXP' -
   Name 'RT1060' -Version '7.0.0' `
   -UpdateType 'microsoft/swupdate:1' -InstalledCriteria '5'
   `
   ```

```
-Compatibility $compat -Files
'%USERPROFILE%\Documents\new_firmware\firmware_7.0.0.bin'

$importManifest | Out-File '.\importManifest.json' -
Encoding UTF8
```

Publish firmware and manifest

1. Open the IoT Hub you created before with Device Update enabled from Azure portal.
2. From the left-hand navigation menu, select ***Device Update*** under Automatic Device Management. Then select the ***Updates*** tab and select ***Import a new update***.
3. Follow the instruction. For the storage container, you can create a new or use existing storage container to host the firmware file. Select ***Submit*** to import the files.


Search resources, services, and docs (G+/)
MICROSOFT (MICROS
Home > Device Update for IoT Hubs > DeviceUpdateDemoAccount1 > DeviceUpdateDemoHub1 >
Import update
Cancel

Import a new update
DeviceUpdateDemoHub1

Import your update file(s) and the import manifest that describes the update.

Select import manifest file *
importManifest.json
The manifest specifies update properties and files.

Select one or more update files *
1 file(s) selected
Update files should be listed in the manifest file.

Select a storage container *
https://mscontosogreenstorage.blob.core.windows.net/test
Update files are staged to an Azure storage container.

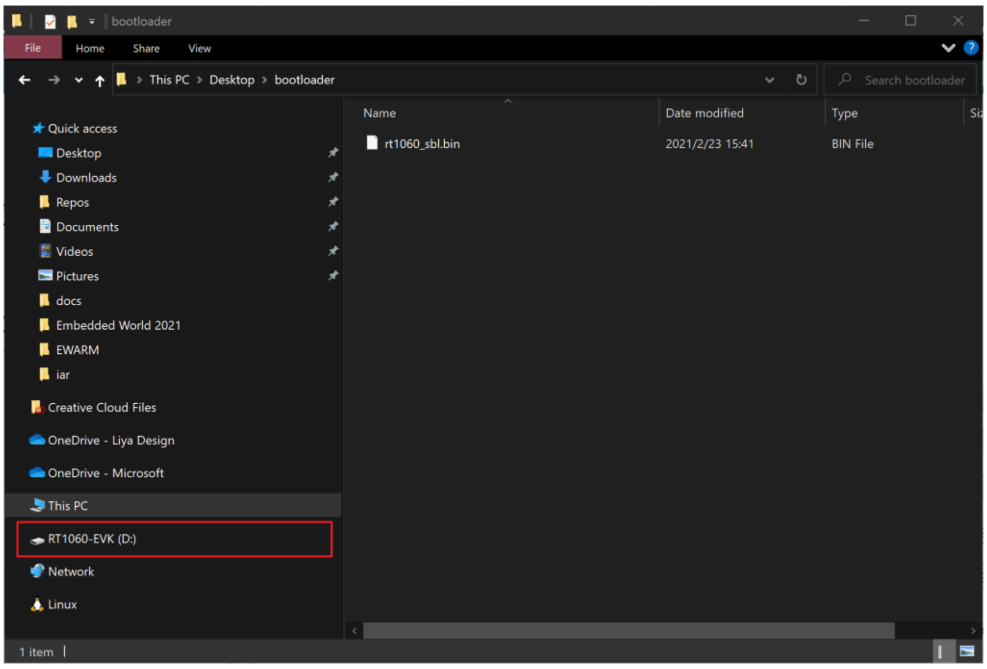
4. You can go to ***Import history*** to see the progress of publishing the files. And once done, it will show in the ***Ready to deploy*** tab.

Prepare the device

We will build and run the same device application but on an older version, so later we can observe the new firmware deployed to it.

Update the bootloader

1. Use the Micro USB cable to connect the ***Debug USB*** port on the i.MX RT1060 EVK, and then connect it to your computer.
2. Open File Explorer, find ***bootloader/rt1060_sbl.bin*** and drag and drop it to ***RT1060-EVK*** mass storage to update the bootloader.



Name	Date modified	Type
rt1060_sbl.bin	2021/2/23 15:41	BIN File

After update, it will reconnect the device again.

Modify the configuration

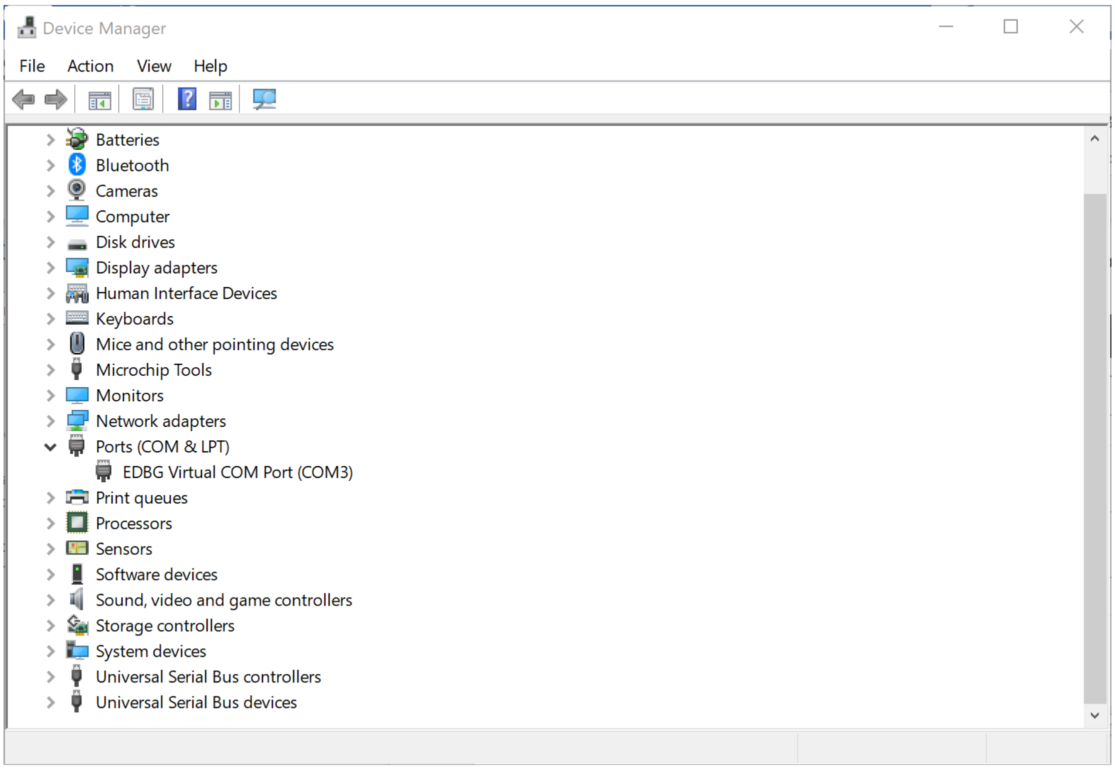
1. In MCUXpresso IDE, open ***sample_azure_iot_embedded_sdk_adu.c***, modify version to an older one to mimic current firmware:

```
#define SAMPLE_UPDATE_ID_VERSION    "6.0.0"
```

2. Now rebuild the ***sample_azure_iot_embedded_sdk_adu*** project.

Download and run the project

1. Use the Micro USB cable to connect the ***Debug USB*** port on the i.MX RT1060 EVK, and then connect it to your computer.
2. Use the Ethernet cable to connect the i.MX RT1060 EVK to an Ethernet port.
3. In MCUXpresso IDE, select ***Start debugging project*** on the toolbar to download the program and run it. Then select ***Resume.***
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] or [Tera Term] and connect to the COM port discovered above. Configure the following values for the serial ports:

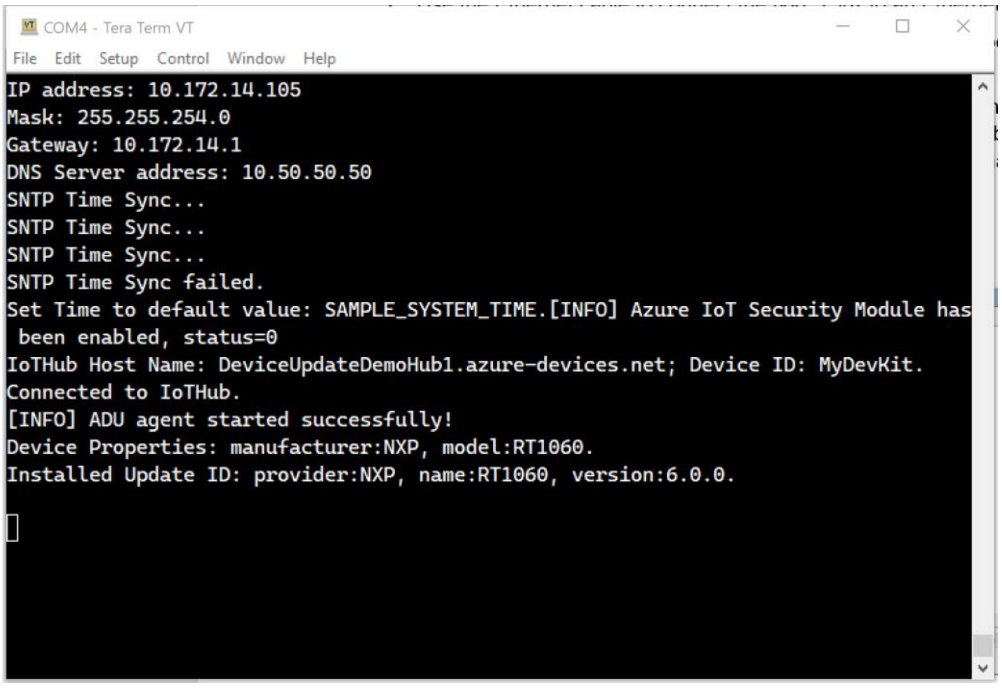
   Baud rate: ***115200***

   Data bits: ***8***

   Stop bits: ***1***

6. As the project runs, the demo prints out status information to the terminal output window. Check the terminal output to verify that messages have been successfully sent to the Azure IoT hub and the version number of the firmware is running.

   NOTE: The terminal output content varies depending on which sample you choose to build and run.



```
IP address: 10.172.14.105
Mask: 255.255.254.0
Gateway: 10.172.14.1
DNS Server address: 10.50.50.50
SNTP Time Sync...
SNTP Time Sync...
SNTP Time Sync...
SNTP Time Sync failed.
Set Time to default value: SAMPLE_SYSTEM_TIME.[INFO] Azure IoT Security Module has
 been enabled, status=0
IoTHub Host Name: DeviceUpdateDemoHub1.azure-devices.net; Device ID: MyDevKit.
Connected to IoTHub.
[INFO] ADU agent started successfully!
Device Properties: manufacturer:NXP, model:RT1060.
Installed Update ID: provider:NXP, name:RT1060, version:6.0.0.
```

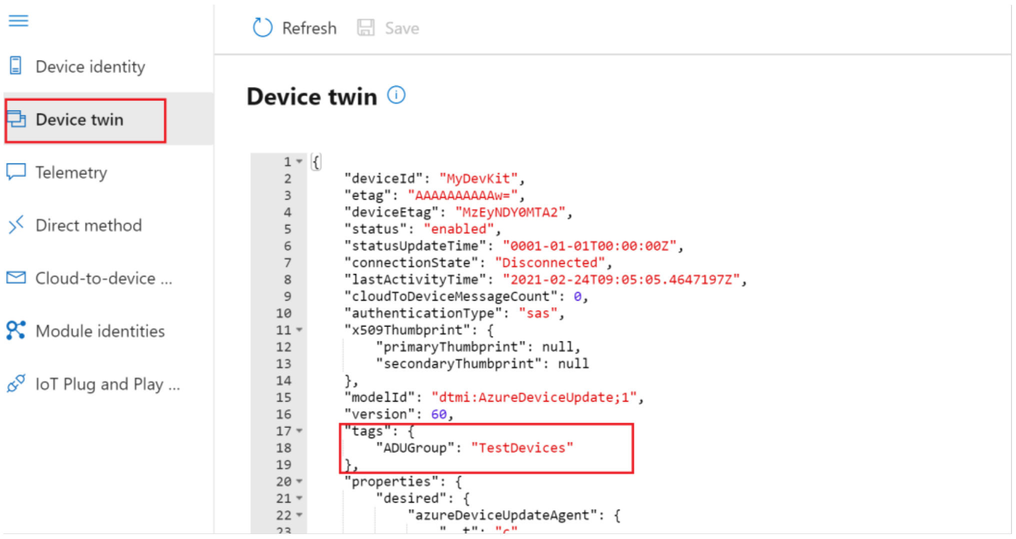
Keep terminal window open to monitor device output in subsequent steps.

Deploy new firmware

Add a tag to your device

1. Keep the device application running from the previous step.
2. Log into Azure portal and navigate to the IoT Hub.
3. From ***IoT Devices***, select the IoT device you use and navigate to ***Device Twin*** tab.
4. Delete any existing Device Update tag value by setting them to null.
5. Add a new Device Update tag value:

```
"tags": {
    "ADUGroup": "<CustomTagValue>"
}
```



Device twin

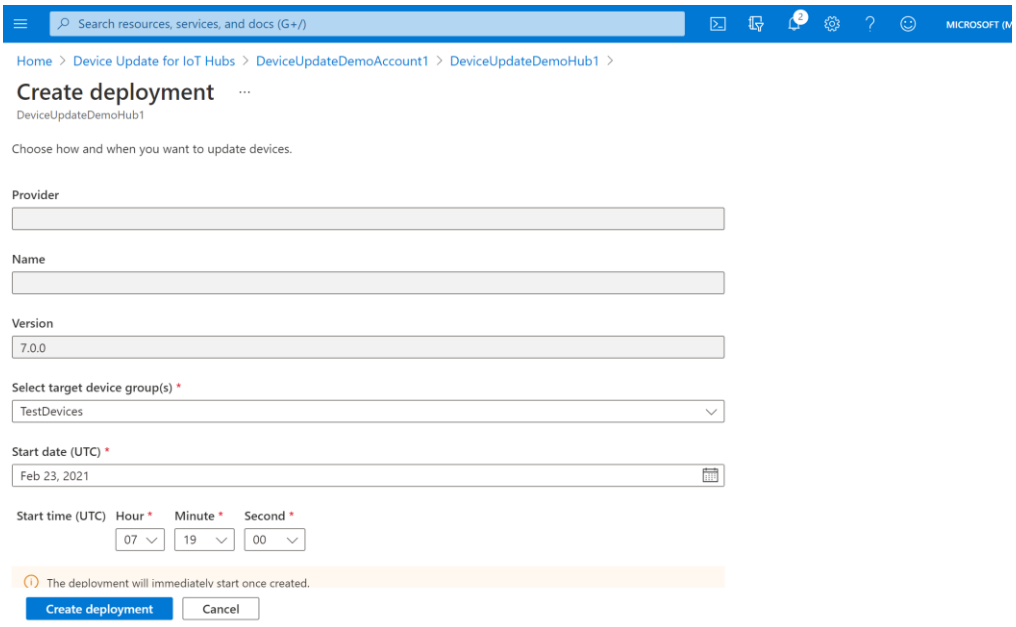
```
{
    "deviceId": "MyDevKit",
    "etag": "AAAAAAAAAAw=",
    "deviceEtag": "MzEyNDY0MTA2",
    "status": "enabled",
    "statusUpdateTime": "0001-01-01T00:00:00Z",
    "connectionState": "Disconnected",
    "lastActivityTime": "2021-02-24T09:05:05.4647197Z",
    "cloudToDeviceMessageCount": 0,
    "authenticationType": "sas",
    "x509Thumbprint": {
        "primaryThumbprint": null,
        "secondaryThumbprint": null
    },
    "modelId": "dtmi:AzureDeviceUpdate;1",
    "version": 60,
    "tags": {
        "ADUGroup": "TestDevices"
    },
    "properties": {
        "desired": {
            "azureDeviceUpdateAgent": {
```

Create update group

1. Go to the IoT Hub you previously connected to your Device Update instance.
2. Select the ***Device Updates***, then select the ***Groups*** tab.
3. Select the ***Add*** to create a new group.
4. Select the IoT Hub tag you created in the previous step from the list. Select ***Create group***.

Deploy update

1. Go to ***Updates*** tab, select the firmware you uploaded and select ***Create a new deployment***.
2. Select the device group you just created and make sure you set the ***Start date (UTC)*** ahead of now to immediately deploy the firmware. Select ***Create deployment*** to roll out the firmware deployment.


Search resources, services, and docs (G+/)
MICROSOFT (M
Home > Device Update for IoT Hubs > DeviceUpdateDemoAccount1 > DeviceUpdateDemoHub1 >

Create deployment

DeviceUpdateDemoHub1

Choose how and when you want to update devices.

Provider

Name

Version: 7.0.0

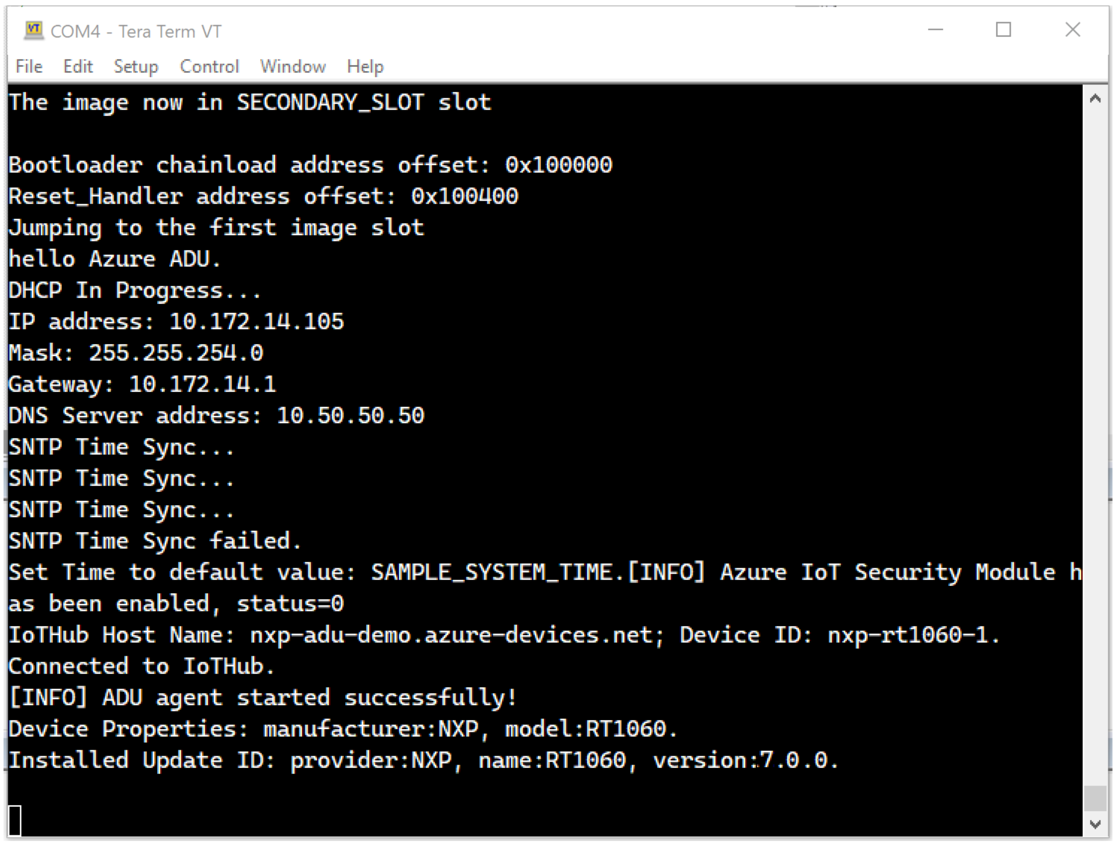
Select target device group(s) *: TestDevices

Start date (UTC) *: Feb 23, 2021

Start time (UTC) Hour *: 07 Minute *: 19 Second *: 00

The deployment will immediately start once created.

3. Go back to Putty or Tera Term, you can see the update firmware is pushed from ADU to the device and after downloading it, the device will reboot with the new firmware:



```
The image now in SECONDARY_SLOT slot

Bootloader chainload address offset: 0x100000
Reset_Handler address offset: 0x100400
Jumping to the first image slot
hello Azure ADU.
DHCP In Progress...
IP address: 10.172.14.105
Mask: 255.255.254.0
Gateway: 10.172.14.1
DNS Server address: 10.50.50.50
SNTP Time Sync...
SNTP Time Sync...
SNTP Time Sync...
SNTP Time Sync failed.
Set Time to default value: SAMPLE_SYSTEM_TIME.[INFO] Azure IoT Security Module h
as been enabled, status=0
IoTHub Host Name: nxp-adu-demo.azure-devices.net; Device ID: nxp-rt1060-1.
Connected to IoTHub.
[INFO] ADU agent started successfully!
Device Properties: manufacturer:NXP, model:RT1060.
Installed Update ID: provider:NXP, name:RT1060, version:7.0.0.
```

Clean up resources

If you no longer need the Azure resources created in this tutorial, you can use the Azure CLI to delete the resource group and all the resources you created for this tutorial. Optionally, you can use Azure IoT Explorer to delete individual resources including devices and IoT hubs.

If you continue to another tutorial in this getting started guide, you can keep the resources you've already created and reuse them.

Important: Deleting a resource group is irreversible. The resource group and all the resources contained in it are permanently deleted. Make sure that you do not accidentally delete the wrong resource group or resources.

To delete a resource group by name:

1. Run the `az group delete` command. This removes the resource group, the IoT Hub, and the device registration you created.

   ```
   az group delete --name MyResourceGroup
   ```

2. Run the `az group list` command to confirm the resource group is deleted.

   ```
   az group list
   ```

Next steps

In this tutorial you created an IoT Hub and added the Device Update resource to it. Then you prepared and deployed a new firmware image.

To learn more about the APIs of the Device Update agent for Azure RTOS, or the Device Update for IoT Hub service, view https://aka.ms/azrtos-device-update-preview.

To learn more about Azure RTOS and how it works with Azure IoT, view https://azure.com/rtos.