NXP Semiconductors
User's Guide

Document Number: SLN-LOCAL-IOT-UG
Rev. 1.1, 2/2020

MCU Local Voice Control Solution User's Guide

CONTENTS

System Requirements and Prerequisites

To run the demos and development tools required for the SLN-LOCAL-IOT kit, and up-to-date computer is required. The following are the system requirements for running the out-of-box demo:

Computer Type	OS Version	Serial Terminal Application
PC	Windows 10	TeraTerm, PuTTY
Mac	macOS Catalina	Serial, CoolTerm, goSerial
PC	Linux	PuTTY

Usage Condition

The following information is provided per Article 10.8 of the Radio Equipment Directive 2014/53/EU:

(a) Frequency bands in which the equipment operates.
(b) The maximum RF power transmitted.

PN	RF Technology	(a) Frequency Range	(b) Max Transmitted Power
SLN-LOCAL-IOT	WiFi	2412MHz – 2472MHz	17.9dBm

EUROPEAN DECLARATION OF CONFORMITY (Simplified DoC per Article 10.9 of the Radio Equipment Directive 2014/53/EU)

This apparatus, namely SLN-LOCAL-IOT, conforms to the Radio Equipment Directive 2014/53/EU. The full EU Declaration of Conformity for this apparatus can be found at this location: https://www.nxp.com/

The product is expected to be used laying flat on a table, microphone output pointing up.

The data mode of the USB bus is not covered by the CE certification as this mode is used exceptionally to reprogram the device.

1. Introduction

NXP's MCU Local Commands Voice Solution is a comprehensive, secure and cost-optimized voice control solution that enables customers to quickly get to market. Based on the high performance **i.MX RT106L** Microcontroller (MCU), the solution enables single chip voice control with plenty of power left over for user-specific tasks.

SLN-LOCAL-IOT hardware highlights:

- Up to 600 MHz (528 MHz default) Cortex-M7 MCU core
- 1 MB of on-chip RAM (512 KB TCM)
- 32 MB HyperFlash memory for fast XiP (e**X**ecute **I**n **P**lace)
- Three PDM MEMS microphones
- TFA9894 Class-D amplifier
- WiFi/Bluetooth combo chip
- Integrated Speaker
- GPIO expansion headers

This solution is supported by a comprehensive and free suite of enablement from NXP and its partners including:

- MCUXpresso Suite of tools (IDE, SDK and corresponding tools)
- Hardware design files
- Software audio tuning tools
- Documentation

2. Getting Started with the SLN-LOCAL-IOT

This section describes the getting start guide for first time users. We show how to run the out-of-box demo, how to get the solution software and tools, and how to build the example projects and run a built project with the kit.

If you already went through 1. Plug It In, 2. Get Software, and 3. Build / Run in the getting start guide on the NXP website (http://www.nxp.com/pages/:GS-SLN-LOCAL-IOT), you can skip this section.

2.1 Plug It In

2.1.1 Unboxing

The **SLN-LOCAL-IOT kit** arrives in a box as shown below. Inside the box, in addition to the kit, you will find a Quick Start Card, "**Let's Get Started**!," and a **USB Type-C to dual**

Type-A cable, as shown in Figure 1. The kit is pre-programmed with an audio playback control demo that is ready to run, as soon as it is powered via the USB cable.

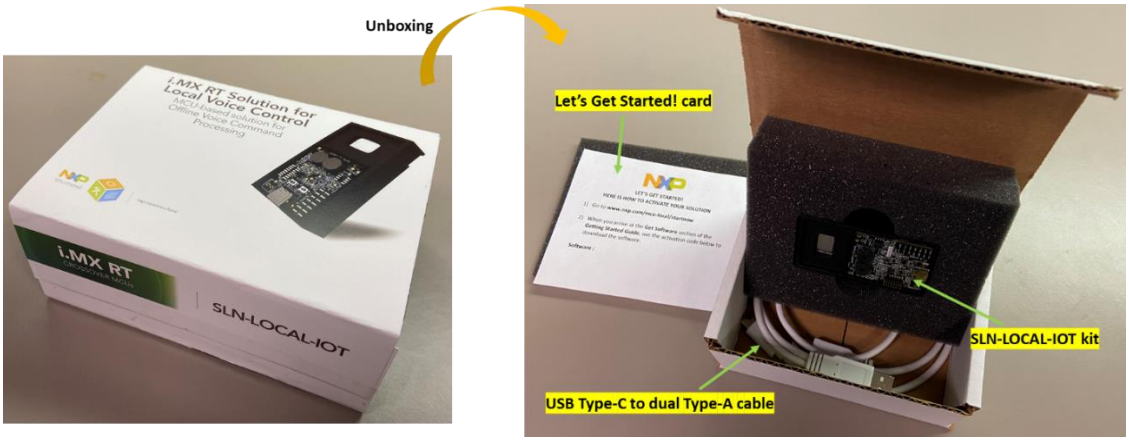

Figure 1. SLN-LOCAL-IOT Kit Contents.

2.1.1.1 Package and Collateral Content

In addition to the contents listed above, owners of the SLN-LOCAL-IOT development kit can also access a software package containing the SLN-LOCAL-IOT MCUXpresso SDK, precompiled release binaries, and manufacturing tools.

Section 2.2.3 details how to download these files for yourself using the download code on the Quick Start Card.

2.1.2 Power On

Plug the USB Type-C connector into the SLN-LOCAL-IOT kit and the dual Type-A connectors into your PC. Figure 2 illustrates how to connect the kit with the USB cable.

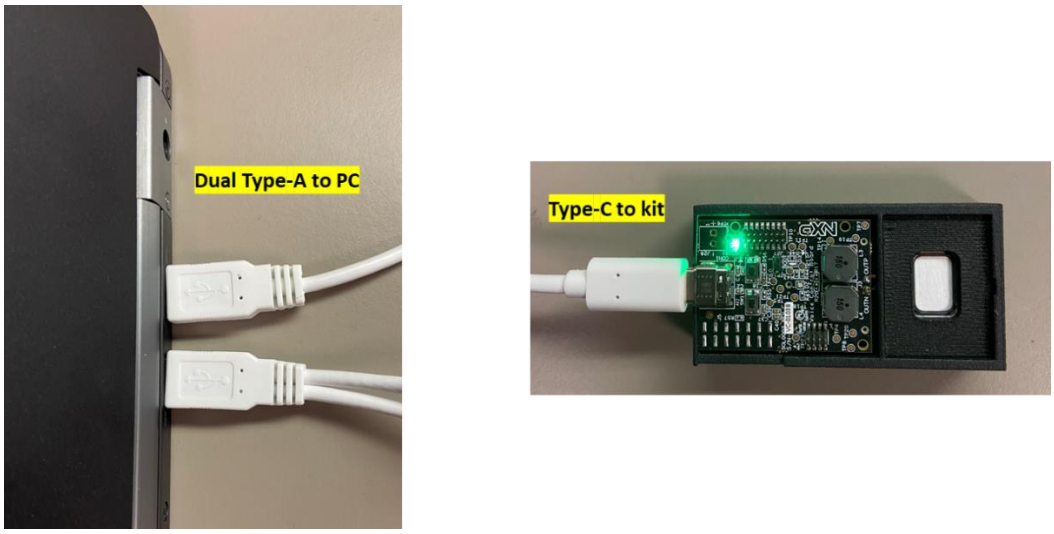

Figure 2. Plugging the USB cable in PC and SLN-LOCAL-IOT Kit.

When you power on the kit for the first time, you will see the LED cycle through various color patterns, as depicted in Figure 3. These are status indicators for the various stages of the boot process as the chip cycles from internal ROM, to bootstrap, to bootloader, to running the application.

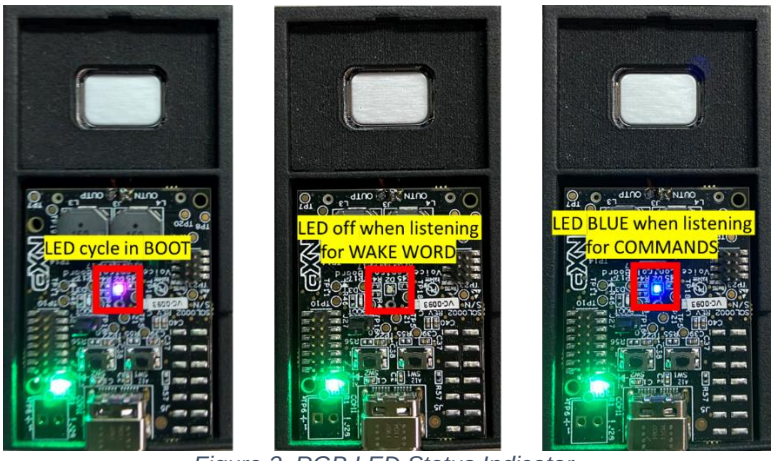

Figure 3. RGB LED Status Indicator.

The final step before the device starts listening will turn the LED red for about 2 seconds before turning off. Once the LED is off, the demo is running and listening for the wake word.

To test the demo, say the "**Hey, NXP**" wake word, followed by a command from the audio playback set (e.g. "**Next track**"), see Section 2.1.3 for the full list of available commands. The SLN-LOCAL-IOT kit audibly responds to every valid command by playing the phrase, "**Intent Detected**," over the built-in speaker. In Section 2.1.4 and 2.1.5 we will show how each command is uniquely acknowledged over the serial terminal. Developers can also modify the software to create different physical responses to commands, e.g. toggle a GPIO on the processor to control a switch, playback different audio files over the speaker, or flash different colors on the LED.

2.1.3 Command Sets

The kit ships with four command sets for control of audio playback (**audio**), smart home (**iot**), washing machine (**wash**), and some generic commands (**misc**). Only one command set can be active at a time. Audio is the default command set. The active command set can be changed via the serial terminal interface (see Section 2.1.5). The generic commands are intended to be used for proof of concept and prototype developments that need commands that are not available in the other three command sets.

- **audio** (default – Play, Pause, Start, Stop, Turn On, Turn Off, Previous Track, Next Track, Volume Up, Volume Down)

- **iot** (Temperature Up, Temperature Down, Windows Up, Windows Down, Turn On, Turn Off, Brighter, Darker)
- **wash** (Wash Normal, Wash Delicate, Wash Heavy Duty, Wash Whites, Cancel)
- **misc** (Alpha, Bravo, Charlie, Delta, Echo)

Please contact NXP (local-commands@nxp.com) to discuss the cost and process to create speech recognition models for custom command sets and wake words. The i.MX RT106L solution can support commands sets containing up to thirty commands, in over sixty different languages.

2.1.4 Connect a Serial Terminal

Connect a serial terminal application to the USB serial device interface that enumerates (115200-8-N-1). Figure 4 is a snapshot of serial terminal setup.

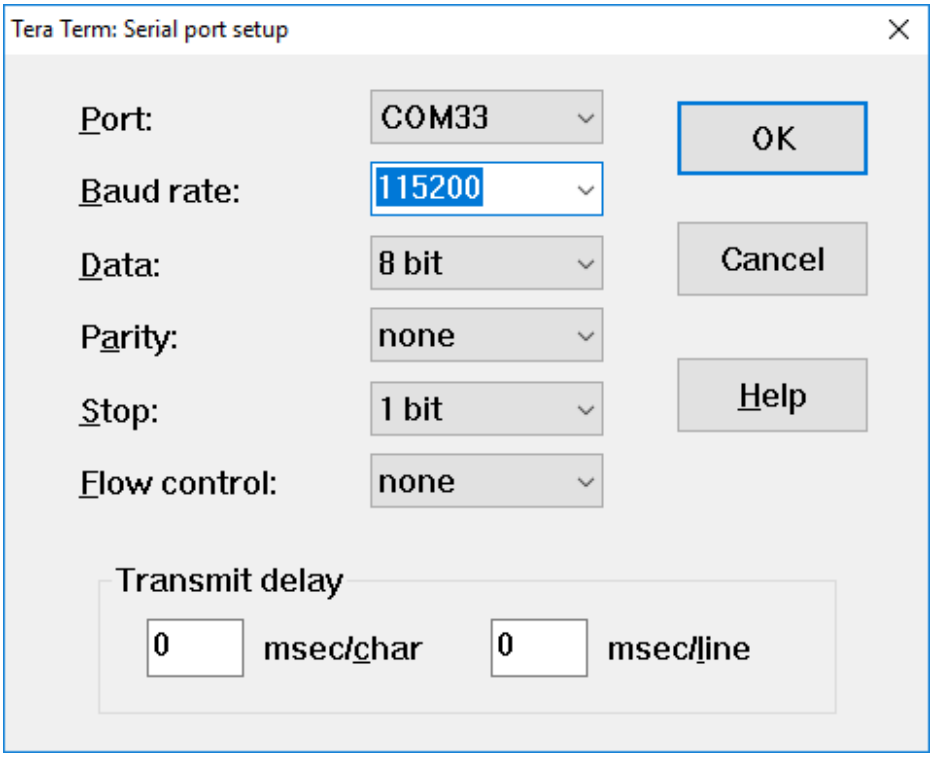

Figure 4. Serial Terminal Port Setup.

Press **Enter** on the keyboard and you will see the **SHELL>>** prompt. Type **help** to show the available commands. Figure 5 shows the available commands, with a description of each.

```
SHELL>> help

"help": List all the registered commands

"exit": Exit program

"reset": Reset the MCU

"volume": Set the volume of the amplifier (0 - 10)
        Usage:
           volume N
        Parameters
           N between 0 and 10

"commands": List available commands

"changeto": Change the command set
        Usage:
           changeto <param>
        Parameters
           audio: Audio Control
           iot: IoT
           misc: Miscellaneous (Alpha, Bravo, etc)
           wash: Washing Machine

"micsoff": Turn microphones off

"micson": Turn microphones on

"updateotw": Restarts the board in the OTW update mode
SHELL>>
```

Figure 5. Shell Interface.

As you use your voice to control the demo, status messages will be displayed in the terminal window showing which command has been recognized. In case of **PLAY** command, you should see the messages in Figure 6.

```
SHELL>>
[ASR] Session started
[ASR] Intent: play
[ASR] Session ended
```

Figure 6. Status Message for "Hey, NXP" Wake Word and "Play" Command.

2.1.5 Listing Commands and Changing Command Sets

Type **commands** to list the active command set and the available commands, as shown in Figure 7.

```
SHELL>> commands
SHELL>>
  Current command set: Audio Control
    - play
    - pause
    - turn_on
    - previous_track
    - next_track
    - stop
    - turn_off
    - start
    - volume_down
    - volume_up
```

Figure 7. Commands for Audio Playback Control.

If you would like to change to a different command set, use the **changeto** command. For example, switching to the IoT command set would use **changeto iot** as shown in Figure 8**.** You can find the other two command sets in Figure 9 and Figure 10.

```
SHELL>> changeto iot
SHELL>>
Switched to IoT command set.

SHELL>> commands
SHELL>>
  Current command set: IoT
    - temperature_up
    - turn_on
    - window_down
    - window_up
    - turn_off
    - brighter
    - temperature_down
    - darker
```

Figure 8. Changing to IoT Command Set and the Commands List.

```
SHELL>> changeto misc
SHELL>>
Switched to Miscellaneous command set.

SHELL>> commands
SHELL>>
  Current command set: Miscellaneous
    - bravo
    - alpha
    - delta
    - charlie
    - echo
```

Figure 9. Changing to Misc Command Set and the Commands List.

```
SHELL>> changeto wash
SHELL>>
Switched to Wash command set.

SHELL>> commands
SHELL>>
  Current command set: Washing Machine
    - cancel
    - wash_delicate
    - wash_normal
    - wash_heavy_duty
    - wash_whites
```

Figure 10. Changing to Washing Machine Command Set and the Commands List.

2.1.6 Changing the Volume

The shell interface provides a mechanism to change the volume and turn the microphones on / off. In addition, if you would like to change the volume without the shell, **SW1** (volume up) and **SW2** (volume down) can be used.

Figure 11 shows where these switches are placed on the kit and Figure 12 is a screenshot taken after changing the volume first via the shell, then using the switches.

Figure 11. SW1 and SW2 on the SLN-LOCAL-IOT Kit.

```
SHELL>> volume 5
SHELL>> Volume set to 5

SHELL>> volume 7
SHELL>> Volume set to 7
Volume set to 8
Volume set to 9
Volume set to 10
Volume set to 10
Volume set to 9
Volume set to 8
Volume set to 7
Volume set to 6
Volume set to 5
Volume set to 4
Volume set to 3
Volume set to 2
Volume set to 1
Volume set to 0
```

Figure 12. Setting the Volume Ranging from 0 to 10.

The default volume of the kit is 50%. Volume is persistent across power cycles.

Use **micsoff** on the serial terminal to mute the microphone as shown in Figure 13. To unmute, use **micson**.

```
SHELL>> micsoff
SHELL>> [ASR] Session ended

SHELL>> micson
SHELL>>
```

Figure 13. Commands for Mute / Unmute.

When the microphone is muted, the LED turns **orange**, as illustrated in Figure 14.

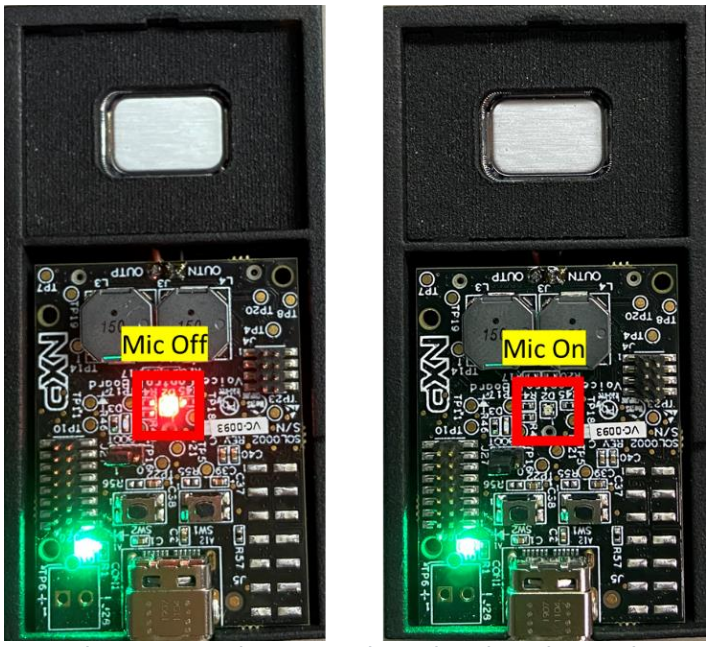

Figure 14. LED in Orange when Microphone is Muted.

2.2 Get Software

We have shown the unboxing and out-of-box demo experience for those who use the kit first time. This section introduces an integrated development environment (IDE) and the solution's software development package. Then, in the later sections, we will describe how to build a simple demo project and introduce deeper development guidelines for the local voice control solution.

2.2.1 MCUXpresso IDE

MCUXpresso IDE brings developers an easy-to-use Eclipse-based development environment for NXP's microcontrollers based on Arm Cortex-M cores. It offers advanced editing, compiling and debugging features with the addition of MCU-specific debugging views, code trace and profiling, multicore debugging, and integrated configuration tools. Its debug connections support every all NXP's MCU based development and evaluation kits, including the SLN-LOCAL-IOT kit, with industry-leading open-source and commercial debug probes from Arm® and SEGGER®

To download NXP's MCUXpresso IDE for free, go to: www.nxp.com/MCUXpresso. Select **MCUXpresso IDE** under the **PRODUCTS** tab. Go to **DOWNLOADS** tab and select the **LATEST VERSION** of the tool. Users will be asked to sign-in/up with a free NXP user-account.

When MCUXpresso installer download completes, double click on the executable, follow the installation instructions and keep the default options. Launch MCUXpresso IDE, define the Workspace location, and press the **OK** button, as shown in Figure 15.

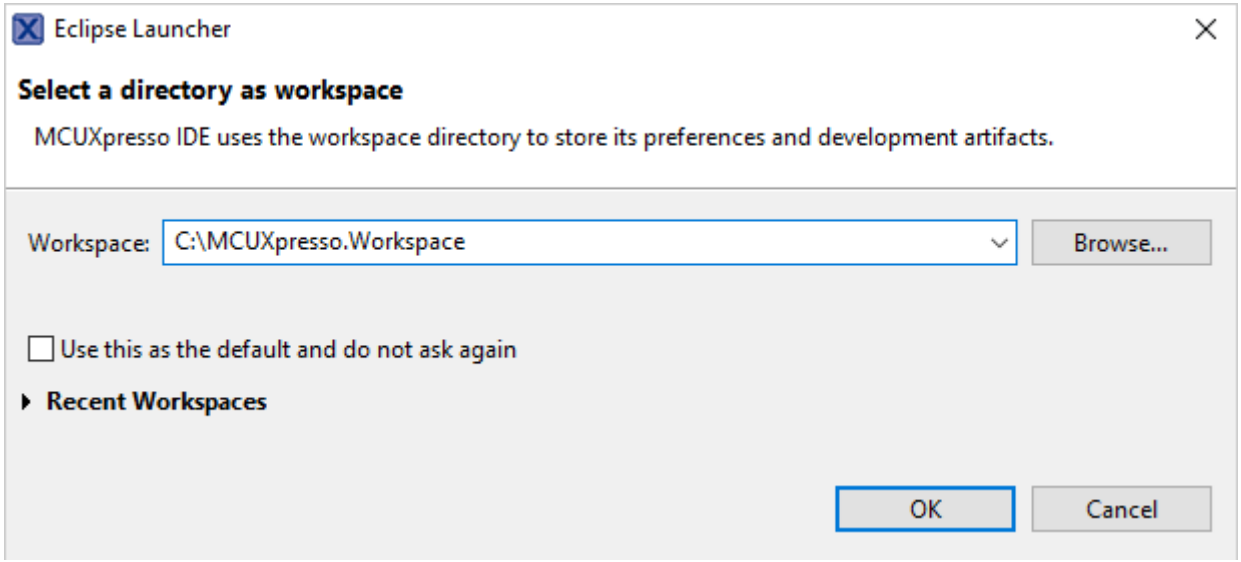

Figure 15. MCUXpresso IDE Workspace.

2.2.2 J-Link Debugger

Segger J-Link is a recommended debugger for the SLN-LOCAL-IOT kit. Users need to make sure that the installed J-Link software version is **later than V6.60**. The J-Link software and documentation pack for various operating systems can be downloaded from www.segger.com/downloads/jlink.

2.2.3 Downloading the Software Package

Follow the steps below to download the software package for the MCU-based Local Voice Control solution.

1) Go to www.nxp.com/activation.
2) Log in with a registered user name and password. If an account is not registered on www.nxp.com, please register at www.nxp.com/webapp-signup/register.
3) After logging in, the user will be sent to the activation page.
4) Enter into the indicated field the download code from the Quick Start Card that comes with the SLN-LOCAL-IOT kit.
5) Follow the instructions on the screen.

The software package consists of the items shown in Figure 16. This package contains everything that users need to run the demo application and develop custom solutions.

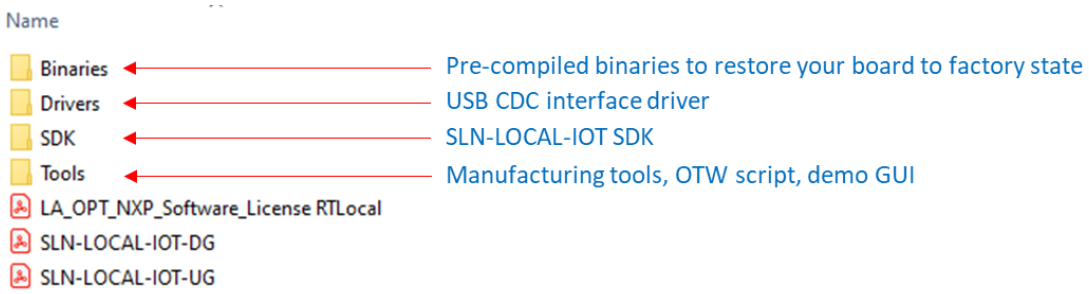

Figure 16. Folder Structure of Software Package.

2.2.4 SLN-LOCAL-IOT SDK

NXP provides the MCUXpresso SDK, a comprehensive software enablement package, designed to simplify and accelerate application developments with NXP's microcontrollers.

SLN-LOCAL-IOT SDK is a customized MCUXpresso SDK for the Local Voice Control solution. Users can find the SDK, as a **.zip** file named **SLN-LOCAL-IOT.zip,** in the SDK

folder of the software package**.** This custom SDK can be used to develop new local voice control solutions with both the i.MX RT106L device and the specific hardware of the SLN-LOCAL-IOT kit. Components of the SDK include an RTOS, middleware (USB, TCP/IP), drivers (MCU and external components), and reference software tools.

Unfortunately, the SLN-LOCAL-IOT SDK is currently **NOT** available in the MCUXpresso SDK Builder on www.nxp.com/mcuxpresso. Instead, the custom SDK is only available in the software package downloaded by following the instructions described in Section 2.2.3.

To install the SLN-LOCAL-IOT SDK into MCUXpresso IDE, simply drag-and-drop the **SLN-LOCAL-IOT.zip** file into the **Installed SDKs** tab at the bottom of the MCUXpresso IDE. Figure 17 shows a screenshot after installing the SDK.

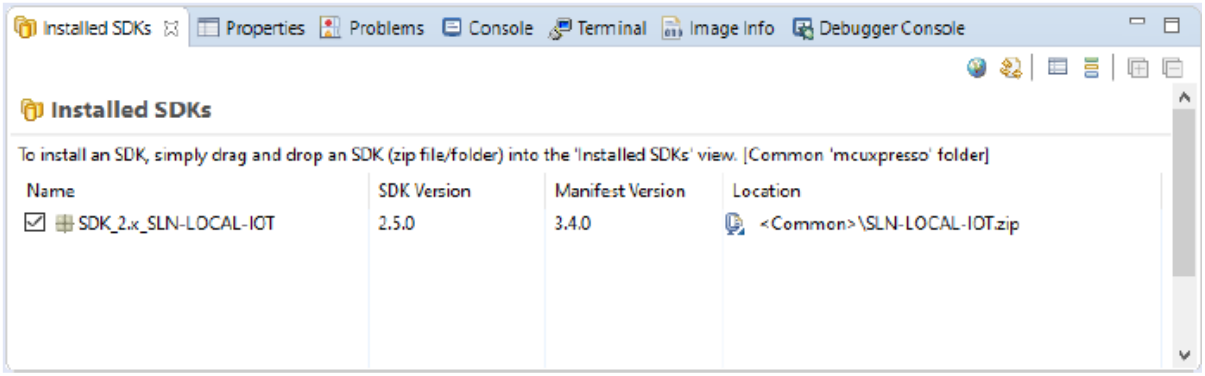

Figure 17. Installed SDKs Tab.

2.3 Build, Run

In this section we show how to import / build / run example projects in MCUXpresso IDE, where the projects are already available in SLN-LOCAL-IOT SDK. Users will experience how to customize the demo application by simply letting the LED blink when a command is detected.

Prerequisite: a J-Link debugger for 2.3.3 Programming the SLN-LOCAL-IOT kit and 2.3.4 Factory Reset in this section. If you do not have such a debugger, then please use the Mass Storage Device (MSD) mode in Section 3 Firmware Update. You can drag-and-drop your application image into the flash without a debugger.

2.3.1 Importing Projects

The installed SDK allows users to import example applications as a development starting point.

To import an example project, select **Import SDK example(s)…** from the **Quickstart** pane in the lower left corner of the IDE as shown in Figure 18.

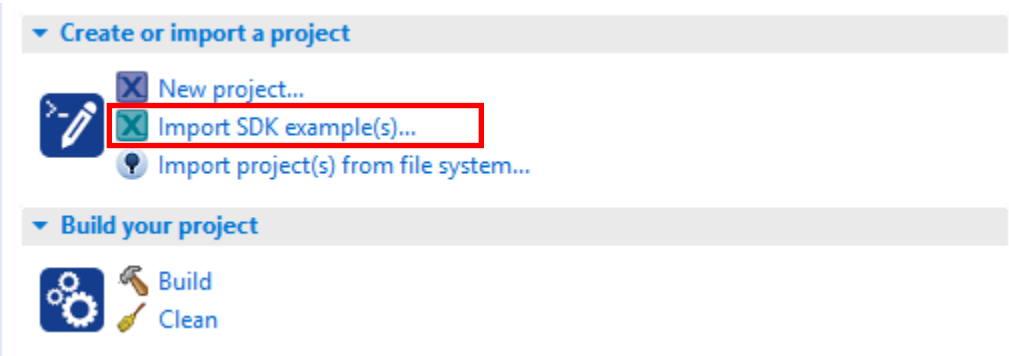

Figure 18. Importing Existing SDK Project.

User should see a pop-up window like Figure 19. Select the **sln_local_iot** option and then proceed by selecting the **Next** button.

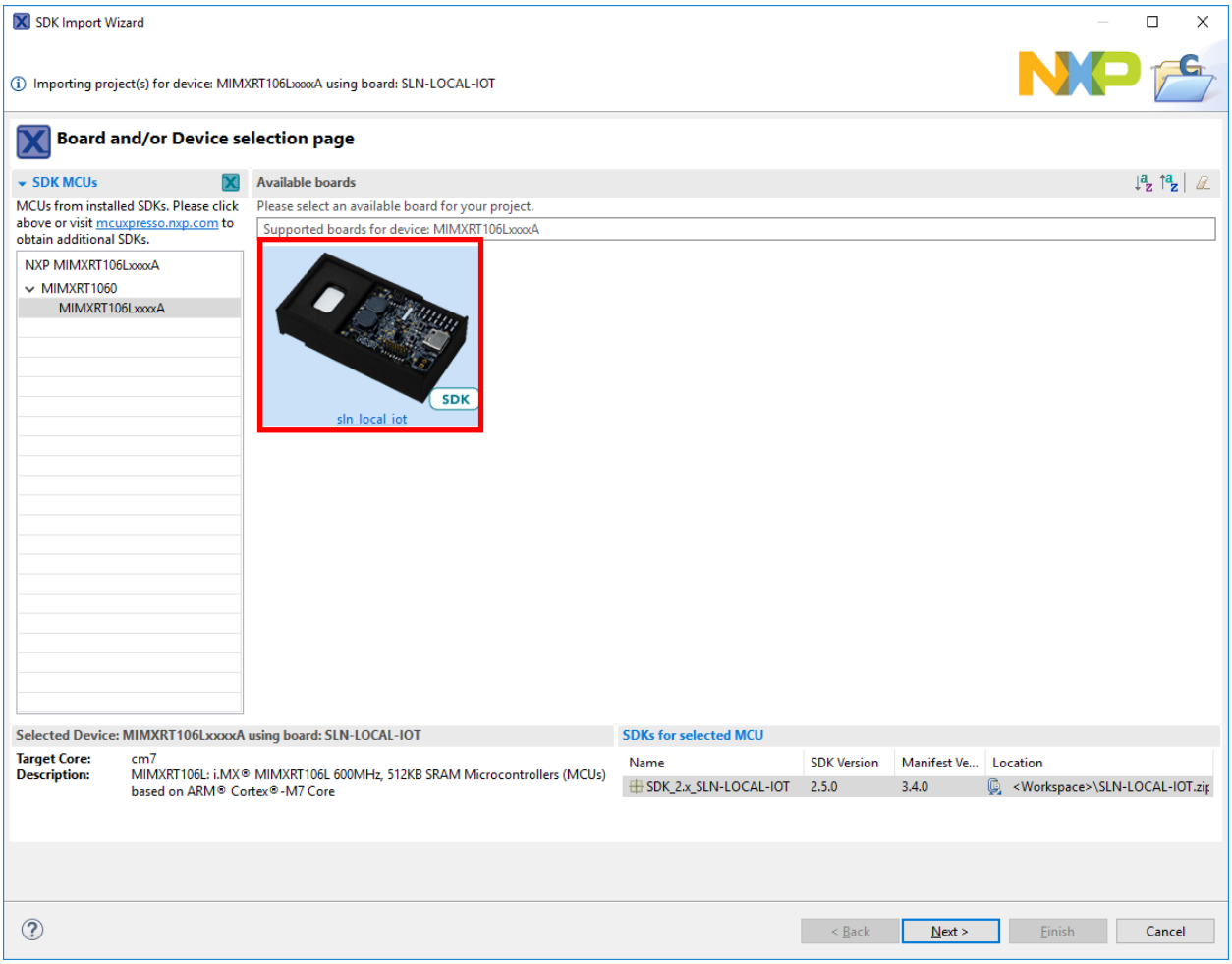

Figure 19. Select SLN-LOCAL-IOT Kit.

The import wizard, as shown in Figure 20, will then display all the projects that can be imported.

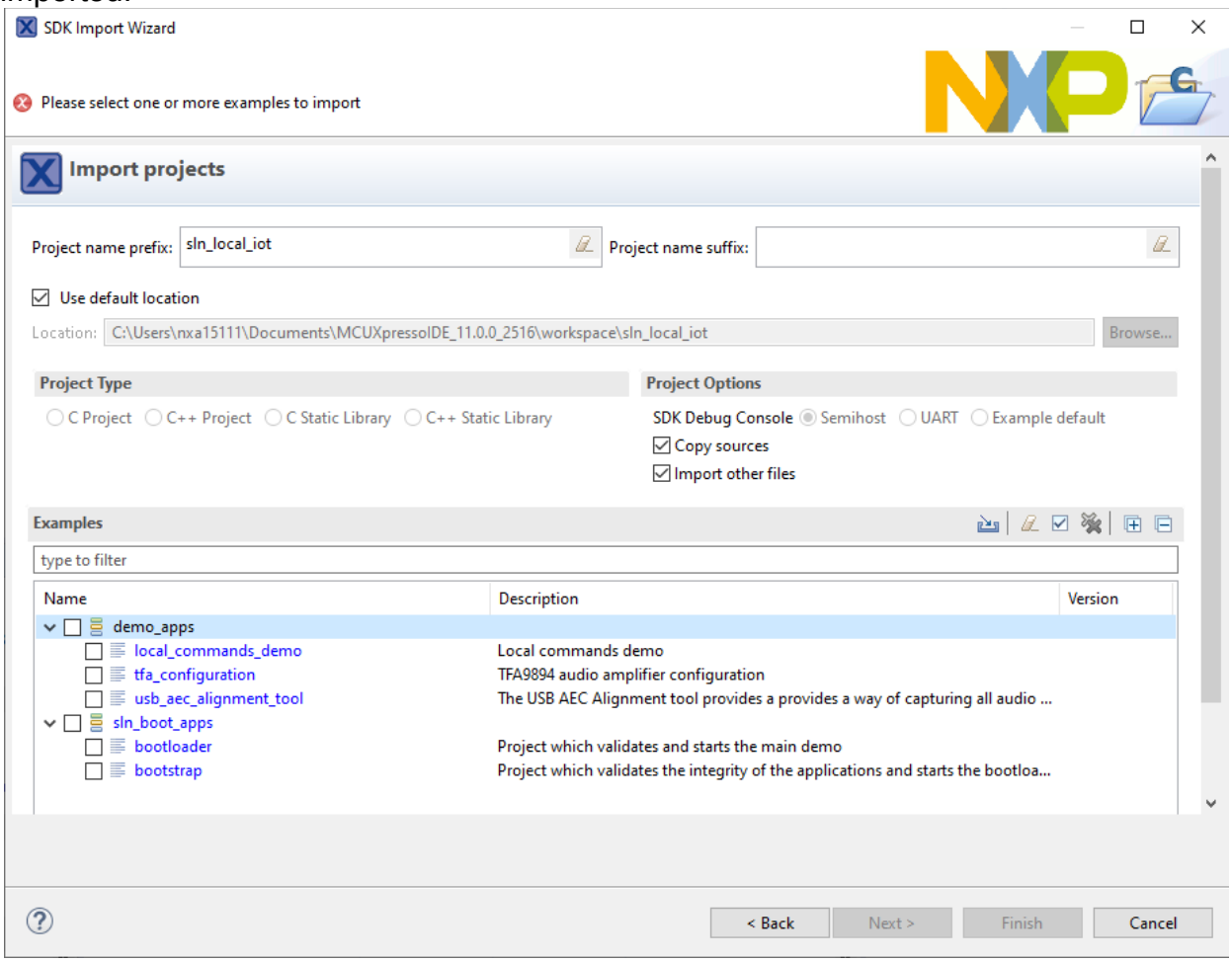

Figure 20. Select Project to Import.

Select **demo_apps > local_commands_demo** to import the demo application project.

Once the project is successfully imported, it will be listed in the project explorer and ready to build and run.

There are other projects available to import, build, and run. However, we briefly introduce bootstrap and bootloader projects in this section. For more details of all the projects, users are recommended to review developer's guide that can be downloaded from www.nxp.com/mcu-local.

The **bootstrap** project is the first flash-resident application that runs. The bootstrap is a minimal FreeRTOS application that is responsible for verification of the bootloader. The bootstrap firmware is designed to be fixed for the product's lifetime, since any corruption of this image will result in an unbootable or bricked device.

The **bootloader** project is the second-stage application that manages signature verification and loading a main application. In addition, the bootloader is responsible for Mass Storage Device (MSD) drag-and-drop and Over-The-Wire (OTW) update of new main application firmware. With the MSD mode, users can simply drag-and-drop new firmware onto the SLN-LOCAL-IOT kit that is connected to the PC via the USB Type-C and dual Type-A cable. With the OTW update mode, users can update a firmware via a wired connection. Currently the SLN-LOCAL-IOT kit supports a UART interface, but it can be extended to support SPI, TCP sockets, I2C, and even a wireless connection. There is more information about the bootloader in Chapter **3.** For those who are interested in the bootloader, please refer to this section.

For more details of **bootstrap** and **bootloader**, please review the developer's guide that can be downloaded from www.nxp.com/mcu-local.

2.3.2 Building Projects

The **local_commands_demo** is the main application and demonstrates the local voice control capabilities of the SLN-LOCAL-IOT kit.

To build the project, click on the project name in project explorer first. Select the **Build** option in the **Quickstart** pane as shown in Figure 21.

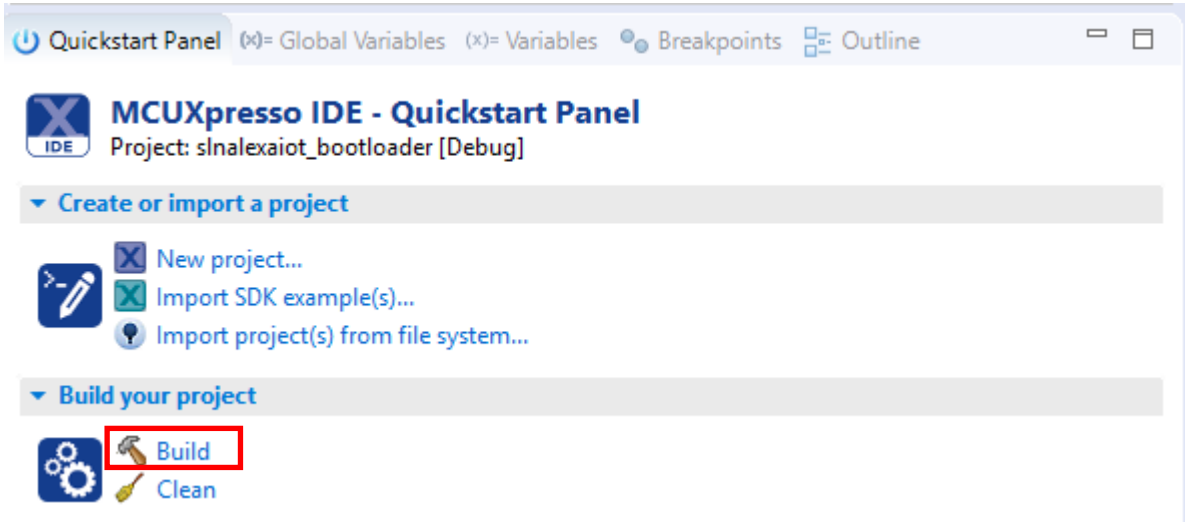

Figure 21. Building an Application.

The other projects, if imported, can be built in the same way.

As a simple demo experience, we will add a LED blink function in the imported demo project. The LED will blink in three cycles with white, red, and blue colors, when a command is detected.

Open **main.c** file under **sln_local_iot_local_commands_demo > source** folder. In the **app_task()**, simply add **RapidBlinkLED();** within `case ASR_COMMAND_DETECTED: {……}` as shown in Figure 22.

```
524                 case ASR_COMMAND_DETECTED:
525                 {
526                     configPRINTF(("[ASR] Inter
527
528                     audio_play_clip(sound_buf
529
530                     RapidBlinkLED();
531
532                     break;
533                 }
```

Figure 22. Adding RapidBlinkLED() in Demo Application.

2.3.3 Programming the SLN-LOCAL-IOT

This section describes how to program the compiled demo project into the SLN-LOCAL-IOT kit.

Make sure the 10-pin J-Link connector is connected to the i.MXRT JTAG connector that is located on the backside of SLN-LOCAL-IOT kit, as shown in Figure 23.

Figure 23. i.MXRT Debug Interface on the Backside of SLN-LOCAL-IOT Kit.

Click on the project name that will be programed into the SLN-LOCAL-IOT kit. Click on the **Debug** button in the **Quickstart** pane as shown in Figure 24.

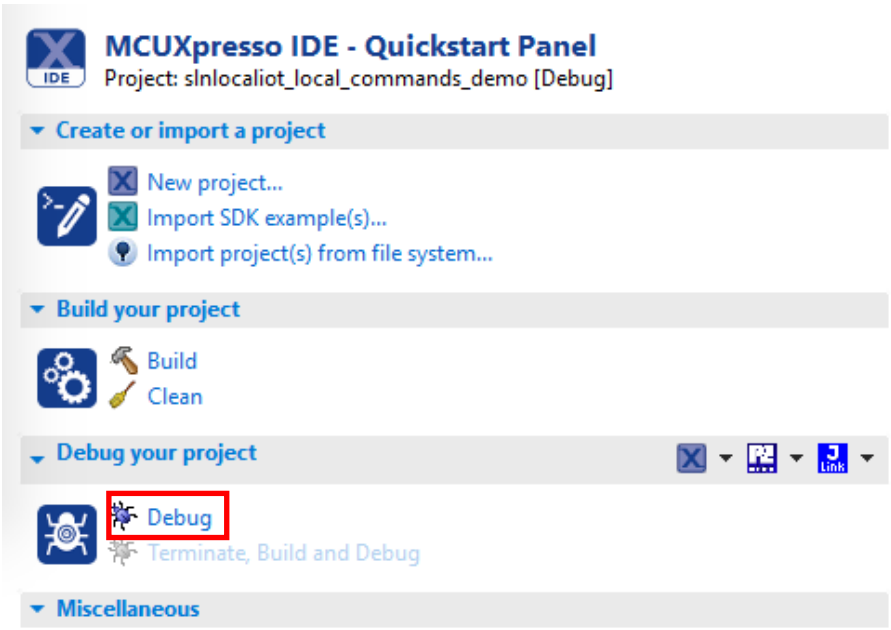

Figure 24. Start a Debug Session.

Users should see a pop-up window like Figure 25. Select the **J-Link probe** that is connected to the kit and press **OK**.

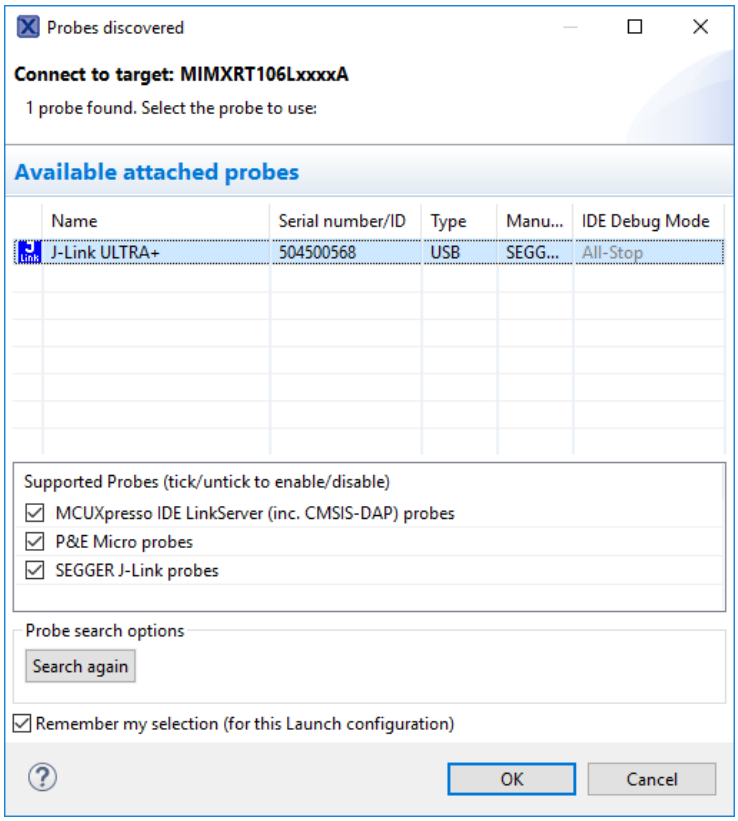

Figure 25. Probe Selection Dialogue.

This will launch the flashing tool and proceed to load the image into the flash. The programming status is displayed on another pop-up window like Figure 26. When this is complete, the debugger will be launched and running. Users can then step, run or set breakpoints.

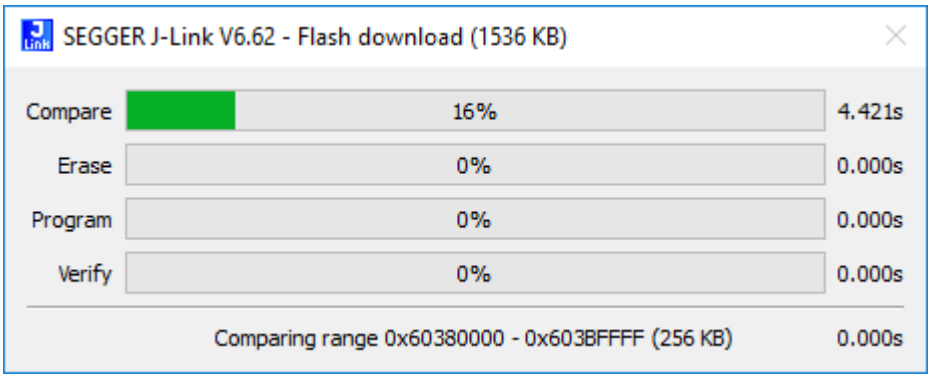

Figure 26. J-Link Programming Window.

2.3.4 Factory Reset

It is possible that something can go wrong during development. NXP provides the full 32 MB application binary for the out-of-box demo as part of the software package for the SLN-LOCAL-IOT. Programming this binary onto the SLN-LOCAL-IOT kit **will erase the entire flash** and replace it with the factory settings.

To restore the kit to the factory state, use the built-in flash programmer in MCUXpresso IDE. The flash tool can be opened by selecting the icon in the image shown in Figure 27.

Figure 27. Opening the GUI Flash Tool.

The Flash GUI Tool will automatically fill in the fields associated with the project that is currently selected.

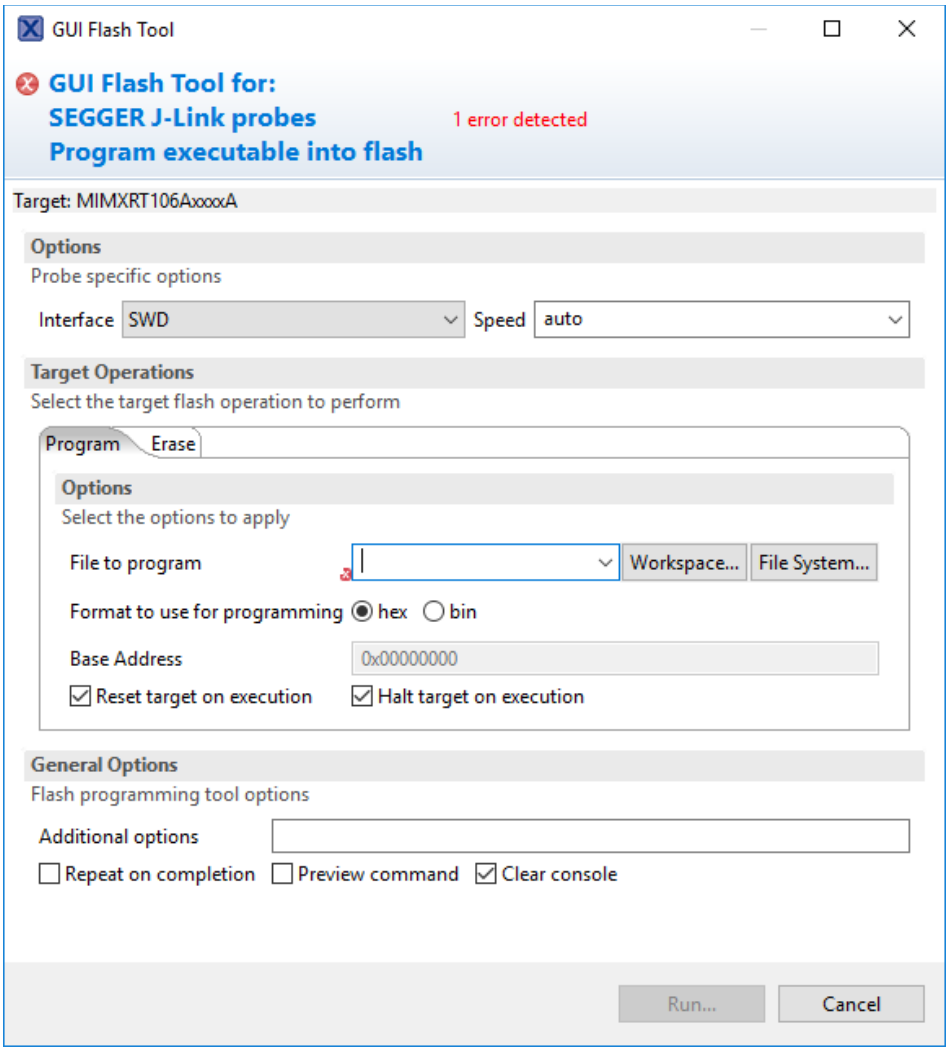

Figure 28. GUI Flash Tool.

After the Flash GUI Tool is opened, users might see an error message as shown in Figure 28. The message will disappear if the file to flash is selected.

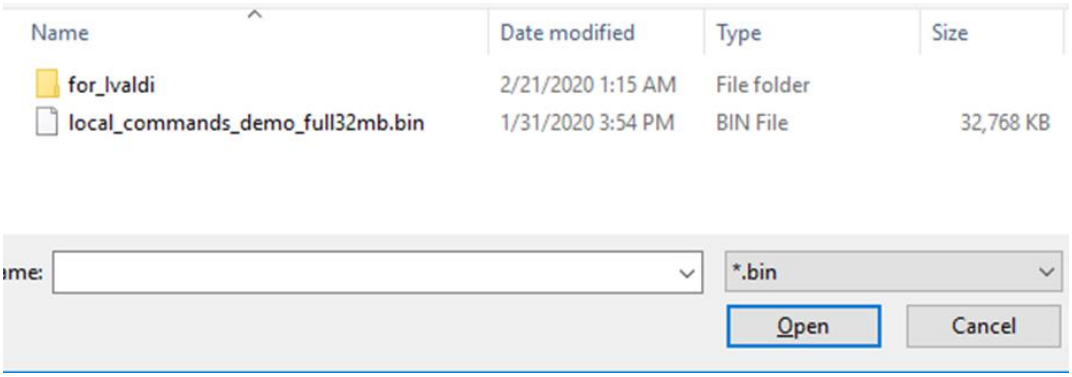

Figure 29. Select the Binary.

Select the **Filesystem** button to open a file selection window and select the **local_commands_demo_full32mb.bin** file from the **Binaries** folder of the SLN-LOCAL-IOT software package. Make sure that the file extension type ***.bin** was selected. Figure 29 shows a screenshot of the binary file selection steps.

In the **Base Address** field, change the address to **0x60000000** to select the base address of the flash. Clicking on **Run…** will kick off the programming operation.

	**PROGRAMMING CAN TAKE UP TO 10 MINUTES DEPENDING ON PC SPECIFICATIONS AND PROBE, SINCE THE BINARY IS 32 MB IN SIZE.**

3. Firmware Update

A main application can be updated in the bootloader of SLN-LOCAL-IOT kit without J-Link probe. **This firmware update is only for the main application, not for the bootstrap and bootloader**. If the bootstrap or bootloader needs to be updated, the users must use J-Link probe. The MCUxpresso projects for boostrap and bootloader are in the SLN-LOCAL-IOT SDK. To modify and update the bootloader or bootstrap, users need to import the project in MCUXpresso IDE, generate the firmware, and program the kit via J-Link probe at appropriate memory addresses. The memory address information can be found in the developer guide SLN-LOCAL-IOT-DG.pdf.

Figure 30 Shows a flowchart to get into the bootloader, where users can enter Mass Storage Device (MSD) mode or Over-The-Wire (OTW) update mode.

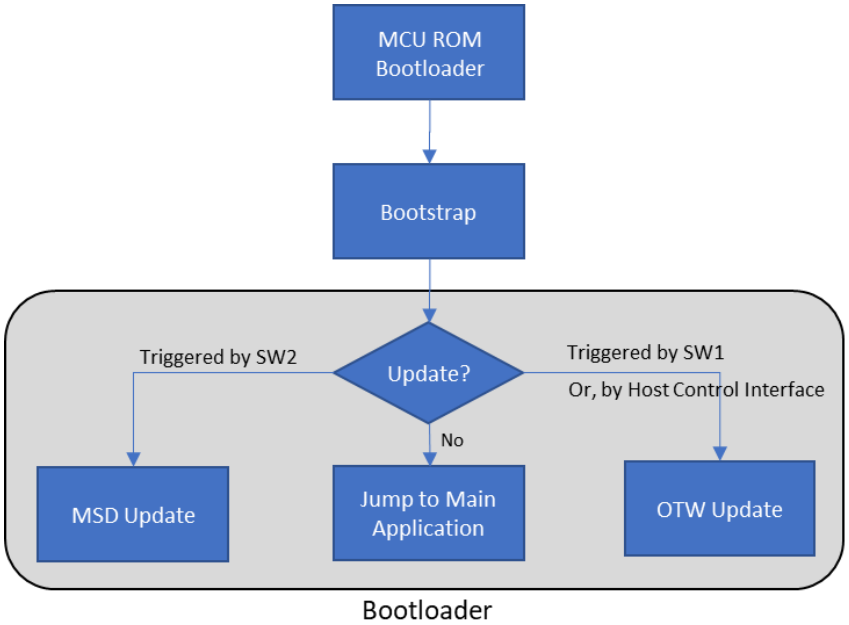

Figure 30. Bootloader Flow.

3.1 Generating a Firmware

This section describes how to generate a main application firmware to update it to SLN-LOCAL-IOT kit via MSD or OTW.

There are two application banks in the flash memory on the SLN-LOCAL-IOT kit.

- Address for Application Bank A: 0x60300000
- Address for Application Bank B: 0x60D00000

The user must configure the bank address properly when the main application is compiled. This ensures that the device is safe to jump into a new application image in one memory location without compromising the other one.

⚠	**IF THE APPLICATION CURRENTLY RUNNING IS IN BANK A, THE NEW IMAGE MUST BE LINKED TO BANK B.**

To change the address from Bank A to Bank B, in the MCUXpresso IDE project explorer, right click on the **sln_local_iot_local_commands_demo** (or, the user's application project name) > **Project Settings** > **Memory**, as shown in Figure 31.

Select **Edit Memory** which will open the **Memory Configuration Editor**.

Change the address of Flash type to **0x6030 0000** for **Application Bank A** and **0x60D0 0000** for **Application Bank B**.

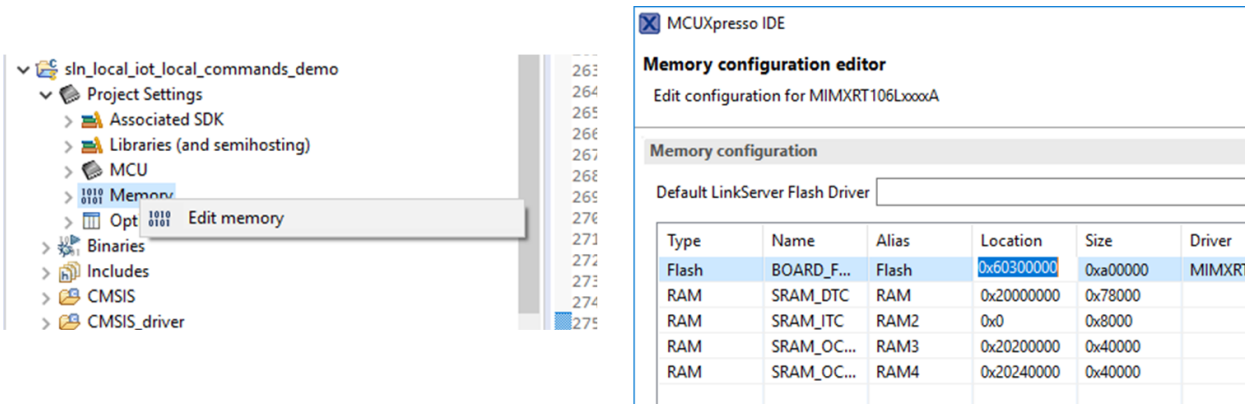

Figure 31. Edit Memory Configuration.

Before building the application, users need to make sure that the MCUXpresso project generates a **.bin** file as an outcome of the build process. Right click on the project name and open **Properties**, as shown in Figure 32.

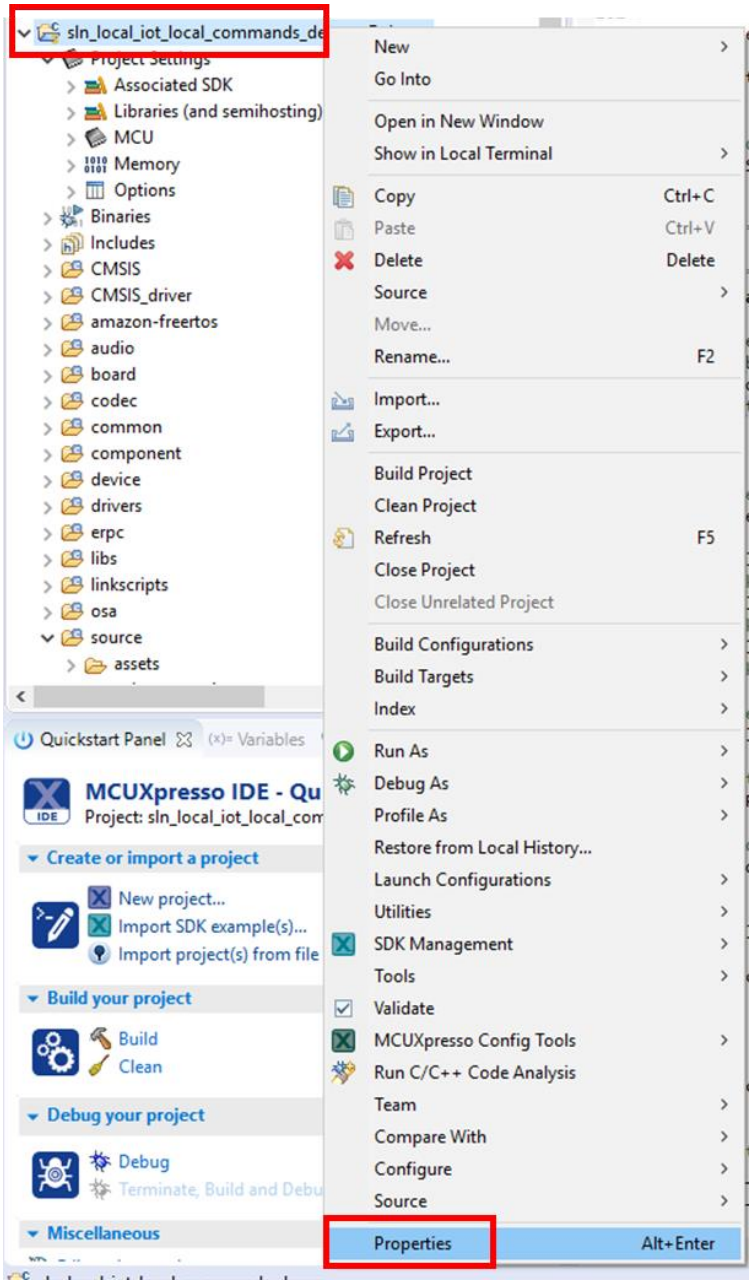

Figure 32. Project Properties.

Expand **C/C++ Build** in the menu and click **Settings**. Select **Build steps** tab where **Post-build steps** can be edited. Click **Edit** and it will show the commands for the Post-build steps. Figure 33 illustrates how to open the Post-build steps window.

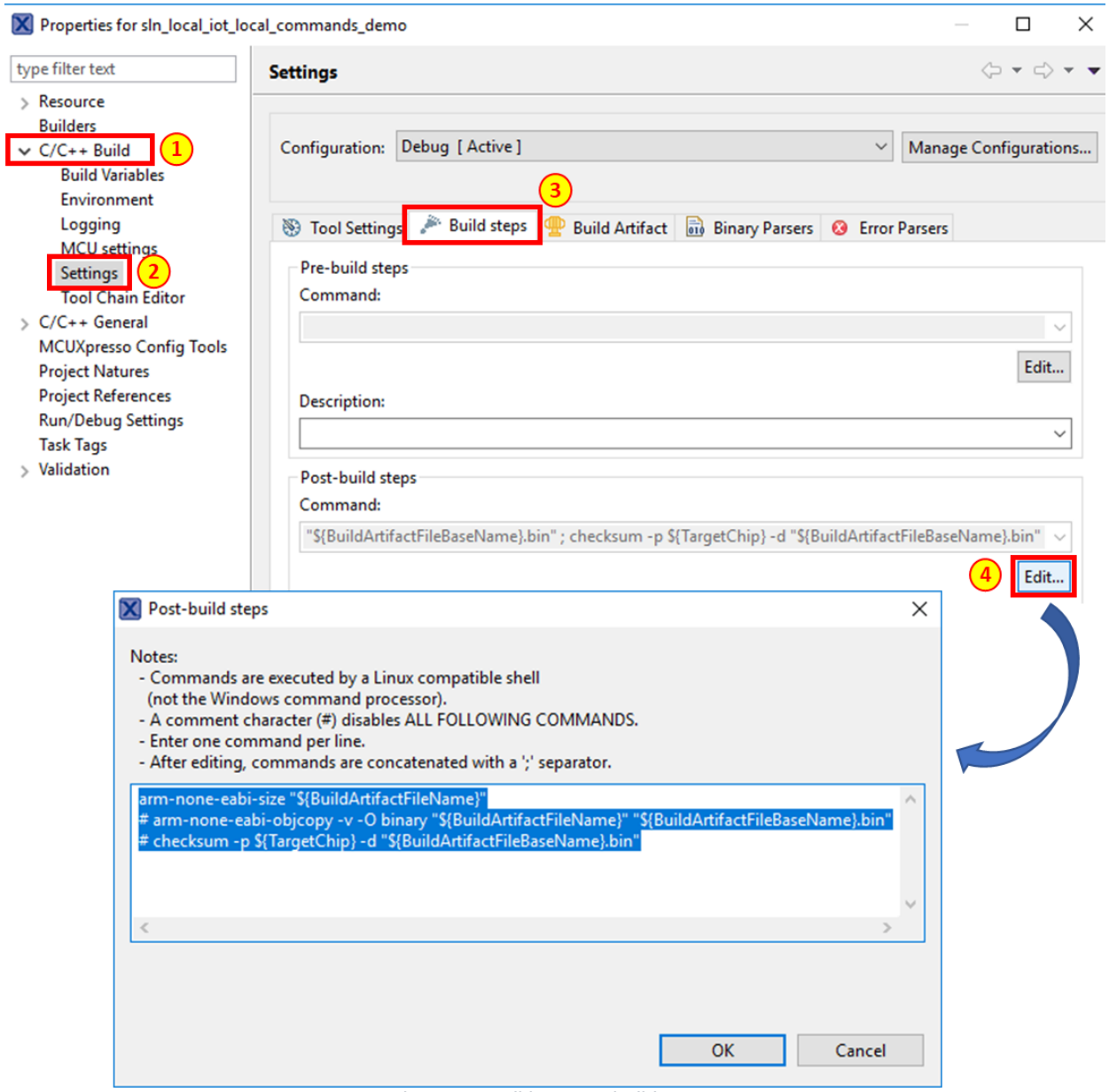

Figure 33. Editing Post-build Steps.

A command character "#" disables all following commands. In order to generate **.bin** file in post-build process, remove the character "#" and click OK. The resulting commands must look like Figure 34 after removing "#":

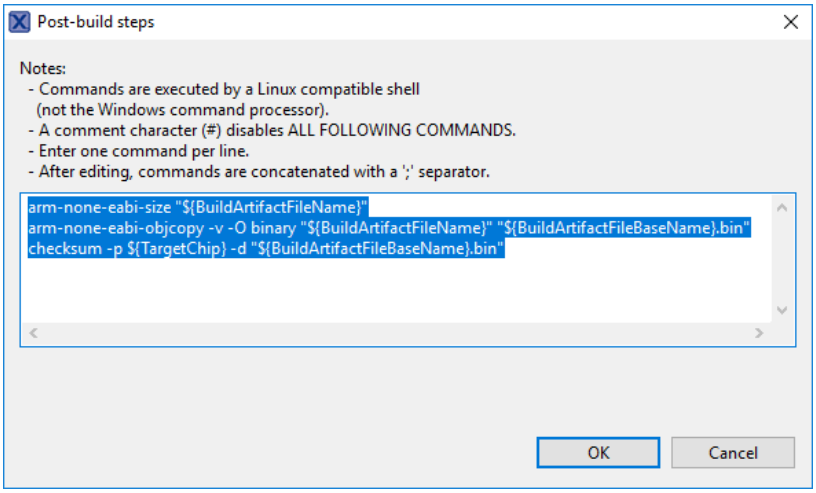

Figure 34. Post-build Commands to Generate .bin File.

If the build process is done successfully, a **.bin** file is generated and placed in **Debug** folder of the MCUXpresso Project.

3.2 USB Mass Storage Device (MSD) Mode

Now users can program the generated firmware image into SLN-LOCAL-IOT kit via USB MSD mode.

Power cycle the SLN-LOCAL-IOT kit, while pressing **SW2** until the pink LED is illuminated as shown in Figure 35. During MSD mode, the pink LED will turn on and off in 3 second intervals.

Figure 35. MSD Update Mode and SW2 Location.

Navigate to the PC's file explorer and confirm that the SLN-LOCAL-IOT kit is mounted as a USB mass storage drive. A mounted kit is displayed on the file explorer as shown in Figure 36.

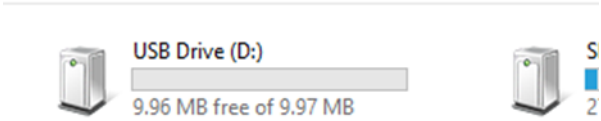

Figure 36. SLN-LOCAL-IOT Kit Mounted as a USB Mass Storage Drive.

Drag and drop the generated **.bin** file onto the MSD drive. This will start the download process and write the **.bin** file to flash. After the image has been programmed into flash, it will begin to execute.

3.3 Over-The-Wire (OTW) Update Mode

OTW update is a process of pushing new application firmware to a device from a remote machine. The device can be connected via UART, SPI, TCP socket, or I2C. When the device is updated with the new firmware in the flash memory, it reboots into the new application assuming all necessary checks have passed.

The OTW update for the SLN-LOCAL-IOT kit is driven by using a simple JSON interface, making it easy to implement host side code on the production line. The kit is currently supported by UART but can be extended to support other interfaces such as TCP sockets, SPI, and I2C.

There are two types of messages transferred: requests and responses. Each transfer contains two pieces: a 4-byte size field and a JSON message. This allows the OTW data interface to be compatible across a wide range of interfaces.

Users can enter the OTW update mode either by pressing **SW1** while booting the kit or by serial terminal **Shell** or the **host control tool** enclosed in the SLN-LOCAL-IOT software package.

NXP provides OTW update tools which are currently intended as a unit test. These tools in the current release are not intended to use for production environment. For more details of the request – response flow and the transfer format structure, please review the development guide SLN-LOCAL-IOT-DG.pdf.

4. Software Tools

This chapter describes the software tools included in SLN-LOCAL-IOT software package.

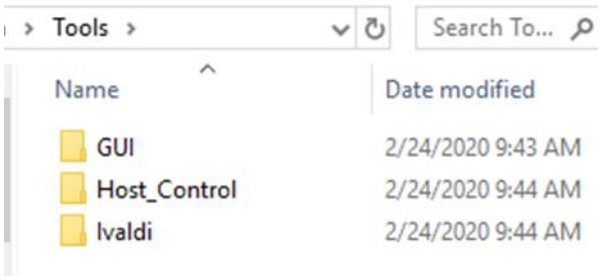

Figure 37. Contents under Tools Folder of SLN-LOCAL-IOT Software Package.

Figure 37 shows what tools are included in the SLN-LOCAL-IOT software package. GUI is for a demo emulating a washing machine. Host_Control folder contains eRPC based host control interface and it is for OTW update. Ivaldi is an NXP provided manufacturing tool.

4.1 GUI – Host Emulator

This emulator is intended to present the washing machine control demo. When the SLN-LOCAL-IOT kit detects a command, the emulator shows on screen which command was detected. Please note that the GUI is only available for Microsoft Windows.

Make sure the SLN-LOCAL-IOT kit is plugged in and waiting for the wake word.

Open the **Tools/GUI folder** in the SW package. The folder structure is shown in Figure 16. Double click the **.exe** file to execute the emulator. You will see that the command set has changed to **wash** on the serial terminal, as shown in Figure 38, and **a washing machine image** like Figure 39 on screen.

⚠	**RUN THE WASHING MACHINE EMULATOR AS AN ADMINISTRATOR ON WINDOWS. THIS ALLOWS THE GUI TO DETECT THE UART PORT OF KIT.**

```
SHELL>>
Switched to Wash command set.
```

Figure 38. Changing to Wash Command Set When Opening the GUI.

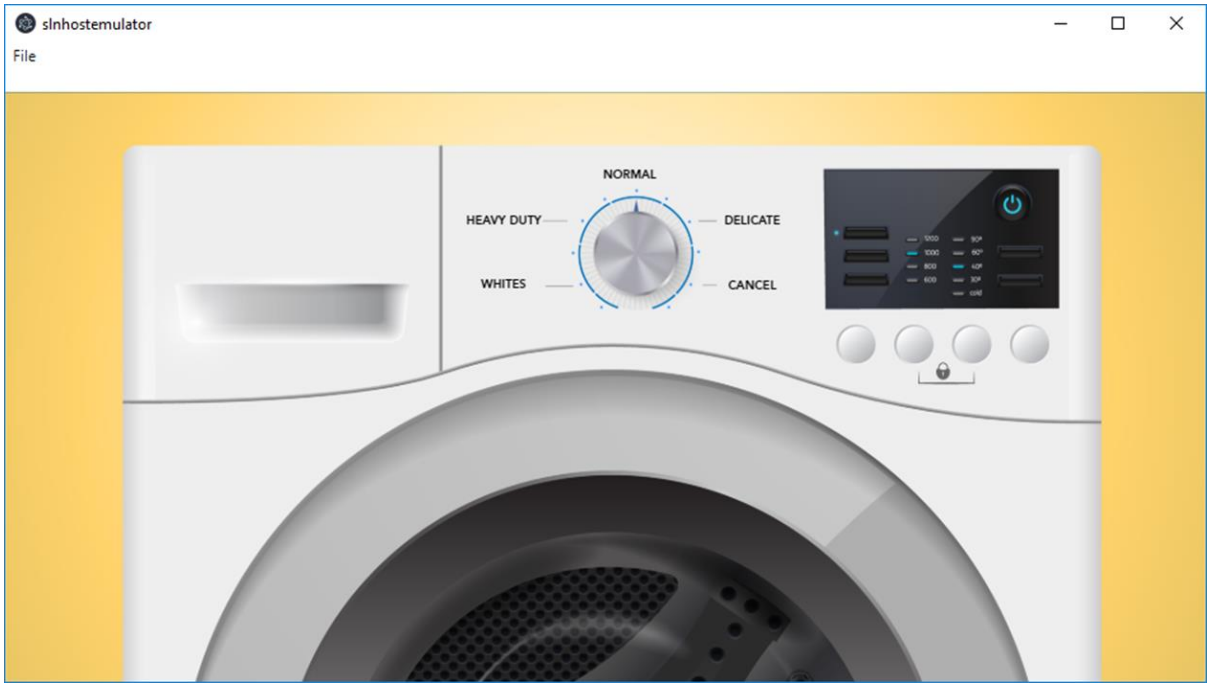

Figure 39. Washing Machine Emulator.

Say "**Hey, NXP**" followed by one of the wash commands – **Wash Normal**, **Wash Heavy Duty**, **Wash Delicate**, **Wash Whites**, and **Cancel**. The emulator indicates which command was detected.

4.2 eRPC Based Host Control Tool

The eRPC based SLN-LOCAL-IOT host control interface currently supports only for Linux environment. Users are recommended to review README.md and Developer Guide SLN-LOCAL-IOT-DG.pdf prior to execute the host interface.

Users can run the unit test following the instruction in the above documents.

4.3 Ivaldi – a Manufacturing Tool

NXP provides a Factory Automation Environment that can be used for securely programming devices on the production line. This collection of scripts is called Ivaldi.

Ivalidi is a package that is responsible for manufacturing programming and reprogramming without needing J-Link. It uses the serial downloader mode of the i.MX RT106L's boot ROM to communicate with an application called Flashloader that is programmed into the RT106L. Ivaldi was created to focus on the build infrastructure of a customer's development and manufacturing cycle. Its primary focuses are:

- Factory programming and device set up
- Enabling High Assurance Booting (HAB) and Encrypted eXecute In Place (eXIP)
- Signing images for Application Bank A or Bank B
- Writing and accessing One Time Programmable (OTP) fuses

For more details of the manufacturing tool, please review the development guide SLN-LOCAL-IOT-DG.

5. References

The following references are available to supplement this document:

- SLN-LOCAL-IOT-DG, Solution Developer's Guide
- Hardware files (Gerbers, schematics, BOM)

6. Revision History

Revision	Date	Changes
0	10/2019	Initial Release for alpha 1.0
1	1/2020	Updates for beta preRC release.
1.1	2/2020	Updates for global launch.

How to Reach Us:

Home Page:
www.nxp.com

Web Support:
www.nxp.com/support