UG10155

i.MX Debian Linux SDK User Guide

Rev. LDLSDK_24.12 — 19 May 2025 **User guide**

Document information

Information	Content
Keywords	UG10155, i.MX, Debian, Linux, SDK, Flexbuild
Abstract	The i.MX Debian Linux SDK Distribution is a Debian-based Linux enablement software for NXP i.MX series processors that are based on Arm cores to provide quick evaluation for customers.

1 Overview

Debian is a free Operating System (OS), also known as Debian GNU/Linux. It provides a wide range of application software, and comes with a total of over 118,000 packages, precompiled software bundled up in a nice format for easy installation for various machines or embedded devices.

The i.MX Debian Linux SDK distribution composes of NXP-specific custom components and open source software developed by the community-supported Debian Project. It aims to provide an easy-to-use and convenient development solution for users' quick evaluation with widely available deb packages on the ARM64 i.MX platforms of NXP.

The i.MX Debian Linux SDK distribution uses Flexbuild (a flexible and easy-to-use build system developed by NXP) to generate the i.MX BSP composite firmware (including ATF, U-Boot, OP-TEE, kernel, DTB, peripheral firmware, initramfs), custom Debian-desktop, and Debian-server RootFS images. It compiles NXP-specific hardware-accelerated components for various i.MX hardware blocks and peripherals (such as GPU, NPU, VPU, ISP, SEC, Wi-Fi/Bluetooth, and Audio) based on Debian runtime dependencies.

Users can use Flexbuild to easily build Debian-based RootFS, Linux kernel, BSP components, and miscellaneous Userspace applications for various use cases (like graphics, multimedia, networking, connectivity, security, and AI/ML) to streamline the system build with flexible customization and efficient CI/CD. Flexbuild Git repository is available at GitHub.

Users can also use the `flex-installer` tool to easily install various Distros to the target storage device (SD/eMMC card or USB/SATA/NVMe disk). For details, see Section 3.

NXP provides Debian-based SDK source and prebuilt demo images as Linux offering for i.MX MPU platforms. The following table provides an overview of the i.MX Debian Linux SDK distribution.

Table 1. Overview of i.MX Debian Linux SDK

Distro Variant	• Debian Base (basic packages) • Debian Server (more packages without GUI Desktop) • Debian Desktop (with GNOME GUI Desktop besides the packages of Debian server)
Deployment of the prebuilt i.MX Debian distro images	NXP provides a script tool flex-installer to automatically download and install the prebuilt i.MX BSP image and Debian RootFS image with customizable partitions of the target storage device. The entire disk space of the SD/eMMC card or USB/SATA disk is accessible with the formatted EXT4 partition. `flex-installer` can also convert the tarball images to a single `.wic` image. Optionally, you can use the balenaEtcher tool to flash the i.MX BSP composite firmware into the SD card on the Windows host machine if the Linux host is not available.
Supported boards	• i.MX 8M Plus EVK • i.MX 8M Mini EVK • i.MX 93 11x11 EVK • i.MX 93 11x11 FRDM • i.MX 91 11x11 FRDM
Host Requirement to build Debian Linux SDK with Flexbuild	• Debian 12. • Build in Docker hosted on Ubuntu LTS or any other distro.
Duration of build	30 minutes - 3 hours
Consumed disk space	30 GB - 50 GB for all i.MX boards.
Installing a new package	Installing a package is as simple as running `apt install <package>` since there is a deb package manager for Debian.
Patching source of component	It is easy to patch i.MX-specific components in Flexbuild, but inconvenient to patch the upstream Debian package because they are installed as deb packages.

2 Release Notes

2.1 What is new in this release

The following new features are added in the i.MX Debian Linux SDK 24.12 release:

- Flexbuild upgraded to 2.16.2412
- Debian 12.8 (base, desktop, server) RootFS with update
- Linux kernel upgraded to LTS 6.6.36
- U-Boot upgraded to 2024.04
- ATF upgraded to v2.10.0
- GPU driver upgraded to `imx-gpu-viv-6.4.11.p2.8d-aarch64` (compiled based on Debian 12 runtime dependency)
- GPUPerfCnt driver upgraded to `libgpuperfcnt-6.4.11.p2.8d-aarch64` (based on Debian 12)
- VPU driver upgraded to `imx-vpu-hantro-vc-1.10.0d` (based on Debian 12)
- ISP driver upgraded to `isp-imx-4.2.2.24.3d)` (based on Debian 12
- Supported eIQ AI/ML and GoPoint components
  - `Tensorflow-lite 2.16.2` with GPU/NPU acceleration
  - `tflite_ethosu_delegate`
  - `tflite_vx_delegate`
  - `tim_vx`
  - `ethosu_driver_stack`
  - `ethosu_firmware`
  - `ethosu_vela`
  - `eiq_examples`
  - `nnstreamer`
  - `nnstreamer_edge`
  - `ssat`
  - `tvm`
  - `nnshark`
  - `imx_demo_experience`
  - `imx_nnstreamer_examples`
  - `imx_smart_kitchen`
  - `imx_smart_fitness`
- DPDK L2FWD and L3FWD applications
- Gstreamer 1.24.0 and various plugins for i.MX

Supported platforms in the i.MX Debian Linux SDK v24.12 release:

- i.MX 8M Plus EVK
- i.MX 8M Mini EVK
- i.MX 93 11x11 EVK
- i.MX 93 11x11 FRDM
- i.MX 91 11x11 FRDM

Note: *Other i.MX platforms may work with Debian but without warranty due to no full test yet.*

Supported features on i.MX 8M Plus EVK and i.MX 8M Mini EVK:

- Debian 12.8 Desktop
- HDMI monitor display

- DSI MIPI Touchscreen display
- Desktop GUI with GPU acceleration
- Multimedia video playback with VPU codec
- MIPI CSI Camera OS08A20 with ISP (only on i.MX 8M Plus EVK)
- MIPI CSI Camera OV5640
- Web browsers (Chromium, Firefox)
- Support Qt6 application
- Wi-Fi + Bluetooth
- eIQ TensorFlow Lite support
- Gstreamer support
- DPDK for networking acceleration

Supported features on i.MX 93 EVK and FRDM:

- Debian 12 Server (Recommended)
- Debian 12 Desktop (PoC, unrecommended yet, which can run but without ideal performance due to no GPU)
- HDMI monitor display
- LVDS Touchscreen display (only on i.MX 93 EVK)
- CSI MIPI Camera AP1302 with ISP
- Wi-Fi + Bluetooth
- eIQ TensorFlow Lite support
- Gstreamer support
- DPDK for networking acceleration

Supported features on i.MX 91 FRDM:

- Debian 12 Server (not support Desktop)
- Wi-Fi + Bluetooth

2.2 Known issues/limitations

The following table lists some key known issues of Debian Linux on the i.MX boards.

Table 2. Known issues and workarounds for i.MX Family SoC

ID	Description	Workaround
DEDI-71	Display: sometimes needs to reboot twice after installing the Debian Desktop.	It will be fixed in next release.

3 Quick Start with Debian on the i.MX Platforms

To deploy the prebuilt i.MX Debian Distro demo images flexibly with less duplication for various i.MX platforms, Flexbuild compiles and assembles the distro images as three parts: BSP composite firmware (board-specific), boot image, and RootFS image (arch-specific for reuse on multiple i.MX platforms).

- BSP firmware image
  The board-specific BSP composite firmware image (such as `firmware_imx8mpevk_sdboot.img`) consists of the ATF, U-Boot, OP-TEE OS, kernel, dtb, peripheral firmware, and initramfs. It provides an entire tiny Linux environment, in which uses can run `flex-installer` to deploy Debian Distro or run any Linux tool to diagnose or repair the system if the Debian Distro is not bootable on the target i.MX board.
  If an x86 Linux host is available, use the `flex-installer` or `dd` command to install the tiny BSP firmware image to the SD card. Otherwise, if there is only a Windows host, use the Etcher tool to install this image.
- Boot image
  This boot image tarball (`boot_IMX_arm64_lts_6.6.36.tar.zst`) consists of the kernel, dtb, Linux modules, Linux firmware, and Distro boot script for reuse on all the ARM64 i.MX boards.
- RootFS image
  The Debian RootFS (`rootfs_lsdk2412_debian_base_arm64.tar.zst`, `rootfs_lsdk2412_debian_desktop_arm64.tar.zst`, and `rootfs_lsdk2412_debian_server_arm64.tar.zst`) consists of the standard Debian 12 deb packages and i.MX-specific driver components with custom configurations for various i.MX hardware blocks.
  Note: *For external users, only the prebuilt `rootfs_lsdk2412_debian_base_arm64.tar.zst` is downloadable directly. Users need to install the Debian base rootfs by `flex-installer` first, and then run the `debian-post-install-pkg` command to install the NXP-specific packages and extra deb packages to upgrade to Debian Desktop or Server.*

Table 3. Unified 64 MB layout of the i.MX BSP composite firmware image generated by Flexbuild

Firmware definition		Max. size	Offset
`Boot loader flash.bin`		4 M	32k or 33k
U-Boot Env		512 K	0x400000
Reserved 1		512 K	0x480000
Reserved 2		1 M	0x500000
Kernel+dtb	`lsdk_tinylinux_imx.itb`	16 M	0x800000
Initramfs		42 M	0x1800000

Table 4. Default partitions of the SD/USB/SATA storage media installed by flex-installer

Region 1 Partition Table 32K/33K	Region 2 Raw 64 - 256 MiB Composite firmware	Region 3 EXT4 512 MiB Boot Partition-1	Region 4 EXT4 8 GiB Backup Partition-2	Region 5 EXT4 Remaining RootFS Partition-3
MBR or GPT	Bootloader env Firmware	kernel & dtb distro boot.scr modules & firmware	Backup partition or Second distro	Debian desktop RootFS or Debian server RootFS

To customize the partitions of the target storage device, use the `flex-installer -i pf -p <partition_list> -d <device>` command.

For examples:

```
$ flex-installer -i pf -d /dev/sdx
(default 3 partitions as 3P=512M:8G:-1)
$ flex-installer -i pf -d /dev/mmcblk0 -p 2P=2G:-1
(customize 2 partitions)
$ flex-installer -i pf -d /dev/mmcblk1 -p 4P=800M:6G:10G:-1
(customize 4 partitions)
```

Note: ***-1*** *indicates the remaining space of the target storage device.*

3.1 Hardware setup

The following hardware is required:

- Micro-SD card Reader
- Micro-SD card (32 GB or larger recommended)
- USB Micro-B or Type-C cable for UART serial communication
- HDMI monitor and HDMI cable for display
- USB mouse and Keyboard (for controlling the UI)
- Ethernet cable (for network access)

3.2 Creating an SD card on the Linux host

The following table lists and describes the options used in the `flex-installer` commands.

Table 5. `flex-installer` command options

Command option	Description	Supported value
`-m <machine>`	Refers to the board name.	`imx8mpevk`, `imx8mmevk`, `imx8mnevk`, `imx8mqevk`, `imx8qmmek`, `imx8qxpmek`, `imx8ulpevk`, `imx91evk`, `imx91frdm`, `imx93evk`, `imx93frdm`.
`-f <firmware>`	Refers to the firmware image.	`firmware_<machine>_<boottype>.img`, for example, `firmware_imx93frdm_sdboot.img`.
`-b <boot_partition>`	Refers to the bootpartition image. There is a set of bootpartition images for each of the Linux kernel versions and platform (64-bit) supported by Layerscape Debian.	`boot_LS_arm64_<lts_version>.tar.zst` (as compressed tarball) or `boot_LS_arm64_lts_6.6.36` (as a directory).
`-B, --bootpart`	Specifies the boot partition number to override the default (default boot partition is the first partition).	For example, `-B 2` or `--bootpart=2`.
`-r <rootfs>`	Refers to the NXP Layerscape Debian RootFS image.	`rootfs_lsdk2412_debian_server_arm64.tar.zst` (compressed tarball) or `rootfs_lsdk2412_debian_server_arm64` (as a directory).
`-R, --rootpart`	Specifies the root partition number to override the default (default root partition is the third partition).	For example, specify the second partition as the root partition: `-R 2` or `--rootpart=2`.
`-d <device>`	Refers to the storage device (SD, USB, or SATA).	`/dev/<device_name>`.

Table 5. `flex-installer` command options*...continued*

Command option	Description	Supported value
	• Use the command `cat /proc/partitions` to see a list of devices and their sizes to ensure that the correct device names are chosen. • The SD/USB/SATA storage drive in the Linux PC is detected as `/dev/sdX`. Where, `X` is a letter, such as a, b, c. Ensure to choose the correct device name, because the data on this device will be replaced. • If the Linux host machine supports read/write SD card directly without an extra SD card reader device, the device name of the SD card is typically `mmcblk0`.	
`-u <url>`	Specifies the URL of the distro web server to override the default one for automatically downloading distro.	URL of the distro web server.

To install the prebuilt NXP i.MX Debian Distro images by `flex-installer`, perform the following steps:

1. Download `flex-installer`.

```
$ wget http://www.nxp.com/lgfiles/sdk/lsdk2412/flex-installer
$ chmod +x flex-installer; sudo mv flex-installer /usr/bin
```

2. Plug the SD card into the Linux host and install the images as follows.

```
$ flex-installer -i pf -d /dev/mmcblk1
(format SD card)

$ flex-installer -i auto -d /dev/mmcblk1 -m imx8mpevk
(automatically download and install images)
```

   It takes 2 minutes to install the i.MX BSP composite firmware and Debian-base RootFS image onto the SD card.

3. Plug the SD card into the i.MX board and install the extra packages as follows.
   Optionally, set the HTTP proxy for apt in `/etc/apt/apt.conf.d/proxy.conf` if required in your network environment.

```
$ dhclient -i end0
(setup network interface by DHCP or setting it manually)

$ date -s "12 DEC 2024 15:00:00"
(setting correct system time is required)

$ debian-post-install-pkg desktop
(install NXP-specific driver packages and extra deb packages for GNOME GUI
 Desktop version)
or
$ debian-post-install-pkg server
(install NXP-specific driver packages and extra deb packages for Server
 version without GUI Desktop)
```

   This step installs the prebuilt NXP-specific hardware driver components and extra deb packages in half an hour.

After finishing the installation, run the reboot command to boot up the Debian Desktop/Server system. Then, log in with the username `debian` or `root` (no password required by default).

Note:

- *Only the prebuilt Debian-base RootFS is downloadable from the nxp.com website. The prebuilt `debian-desktop` and `debian-server` are not accessible for external users.*

Users can build custom `debian-desktop` or `debian-server` image in Flexbuild if needed (see Section 4).

- *Sometimes, the downloaded images (`firmware_<board>_sdboot.img`, `boot_IMX_arm64_lts_6.6.36.tar.zst`, or `rootfs_lsdk2412_debian_base_arm64.tar.zst`) may be damaged and incompleted since the unstable network break. Remove the incompleted images locally and re-download the images by reruning the flex-installer and ensure that the network is reliable.*

3.3 Creating an SD card on the Windows host

To create an SD card on the Windows host, perform the following steps:

1. Download the balenaEtcher flasher tool (https://github.com/balena-io/etcher/releases/tag/v1.18.13) and install it on the Windows host.
2. Download the prebuilt i.MX BSP composite firmware.
   You can create a folder (e.g., `C:/Debian`) and download the following board-specific image to this folder.

```
http://www.nxp.com/lgfiles/sdk/lsdk2412/firmware_imx8mpevk_sdboot.img
http://www.nxp.com/lgfiles/sdk/lsdk2412/firmware_imx93frdm_sdboot.img
http://www.nxp.com/lgfiles/sdk/lsdk2412/sd_pt_32k.img
(or sd_pt_33k.img for imx8mm and imx8mq)
```

3. Combine the partition table image with the BSP composite firmware under the cmd prompt as follows.

```
C:\Windows\System32> cd C:/Debian
C:\Debian>  dir
C:\Debian>  copy /b sd_pt_32k.img + firmware_imx8mpevk_sdboot.img
 firmware_imx8mpevk_sdboot.wic
```

   The new image `firmware_imx8mpevk_sdboot.wic` is generated.
4. Run the balenaEtcher tool, choose the generated `.wic` file and SD card, and then start flashing the image.

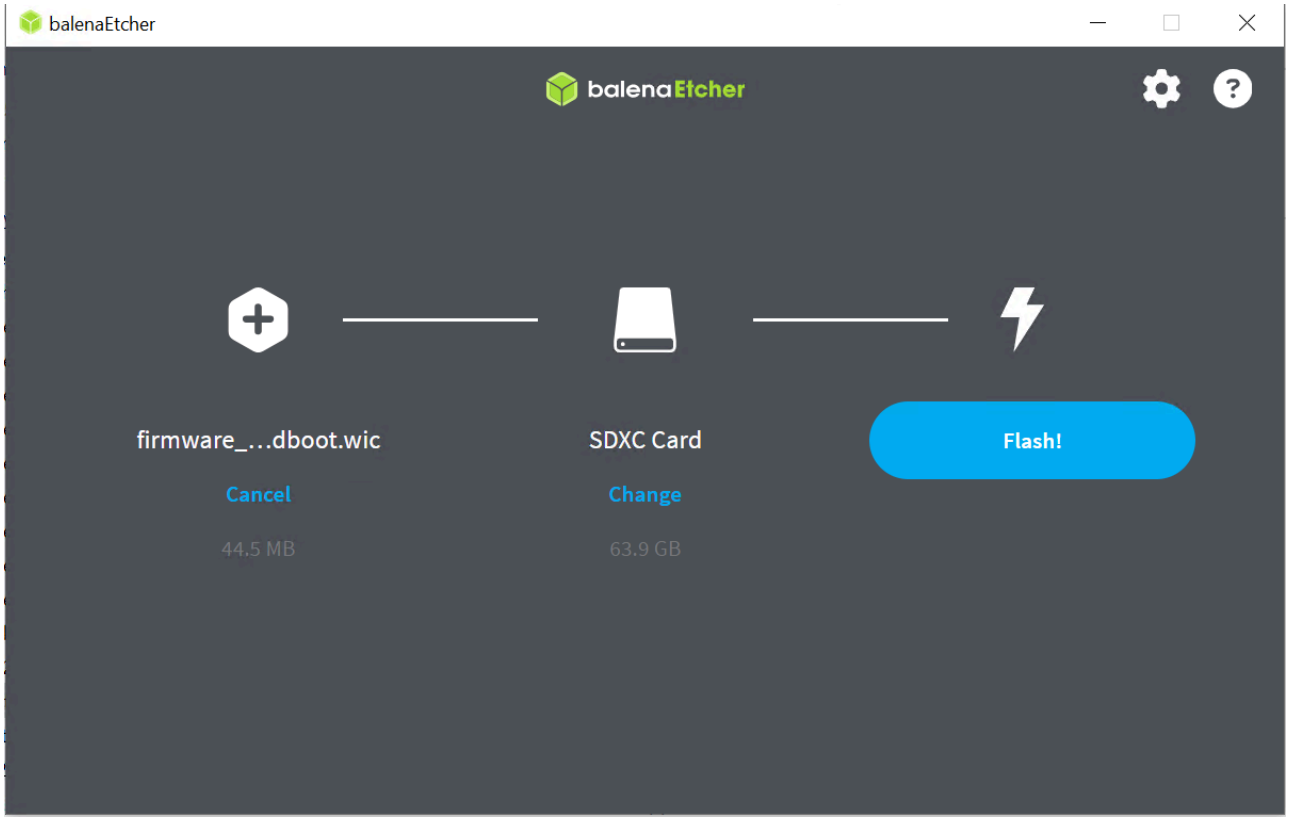

Figure 1. Flashing image by the Etcher tool

5. Boot up TinyLinux and install the i.MX Debian Distro by `flex-installer` as follows.
   a. Unplug the SD card from the Windows host and plug it into the target board. Then, set the DIP switch for SD boot if needed.
   b. After powering on the target board, run the following commands under the U-Boot prompt to boot TinyLinux.

```
=> setenv tinylinux 'mmc read 0xa0000000 0x4000 0x1f000 && bootm
 a0000000#imx8mpevk'
=> saveenv; run tinylinux
```

   c. Log into the TinyLinux with the username `root`, set up the network on the board, and install the Debian Distro by the following commands:

```
root@TinyLinux:~#  udhcpc -i eth0
(DHCP dynamic IP or manually set static IP)

root@TinyLinux:~#  flex-installer -i pf -d /dev/mmcblk1
(format SD card)

root@TinyLinux:~#  flex-installer -i auto -d /dev/mmcblk1 -m imx8mpevk
(automatically download and install Debian-base, boot, and firmware
 images.)
```

   This takes 2 minutes to download and install the i.MX Debian base RootFS image, boot tarball image, and BSP firmware image onto the SD card.
   After finishing the installation in TinyLinux, run the reboot command to reset the system to log into the Debian base system.

d. Once logging into the Debian base system with the username `debian` or `root`, run the following commands to install extra packages (Optionally, set the HTTP proxy for apt in `/etc/apt/apt.conf.d/proxy.conf` if needed in your network environment).

```
$ dhclient -i end0
(setup Ethernet network interface by DHCP or setting it manually)

$ sudo date -s "12 DEC 2024 15:00:00"
(setting correct system time is required)

$ debian-post-install-pkg desktop
(install NXP-specific desktop packages and extra deb  packages for GNOME
 GUI Desktop version)
or
$ debian-post-install-pkg server
(install NXP-specific server packages and extra deb  packages for Server
 version without GUI Desktop)
```

This step installs the prebuilt i.MX-specific hardware driver components and extra deb packages in half an hour.

After finishing the installation, run the reboot command to boot up the Debian Desktop/Server system.

3.4 Debian applications on the i.MX platforms

Connect the HDMI or DSI MIPI Display, Mouse, Keyboard, and the Ethernet cable to the i.MX evaluation board. Insert the SD card in the board and power on the board. After approximately 20 seconds, the board should boot to the Debian GNOME Desktop home screen after login with the username `debian`.

Click the **Settings** icon at the top right corner shown as follows.

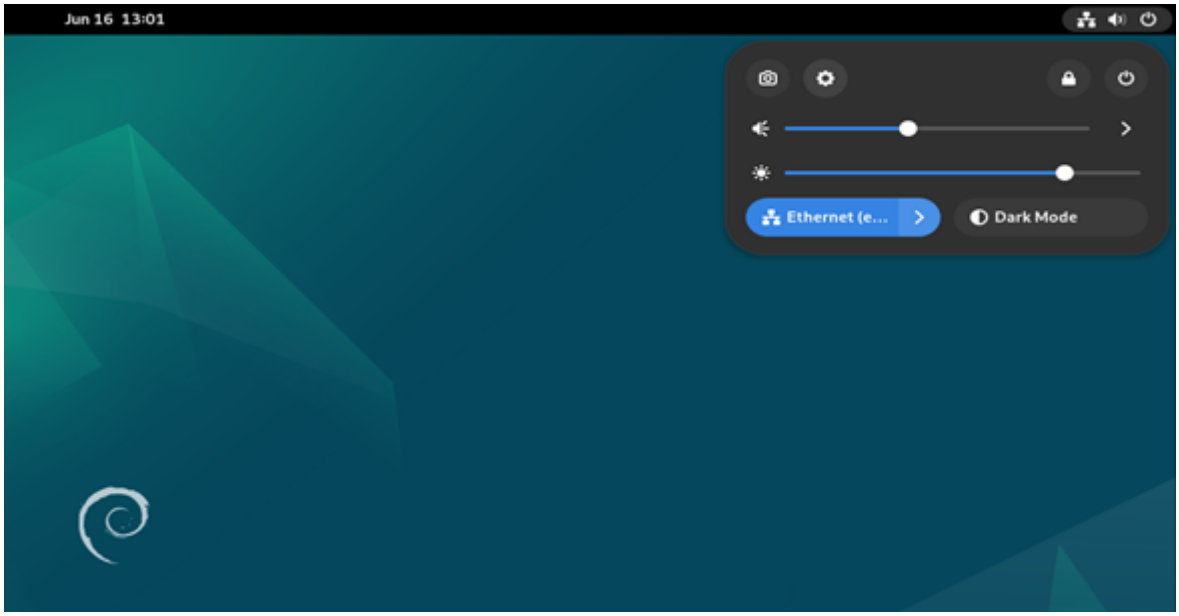

Figure 2. Click the Settings icon

Then, you can see the **Settings** configuration UI shown as follows.

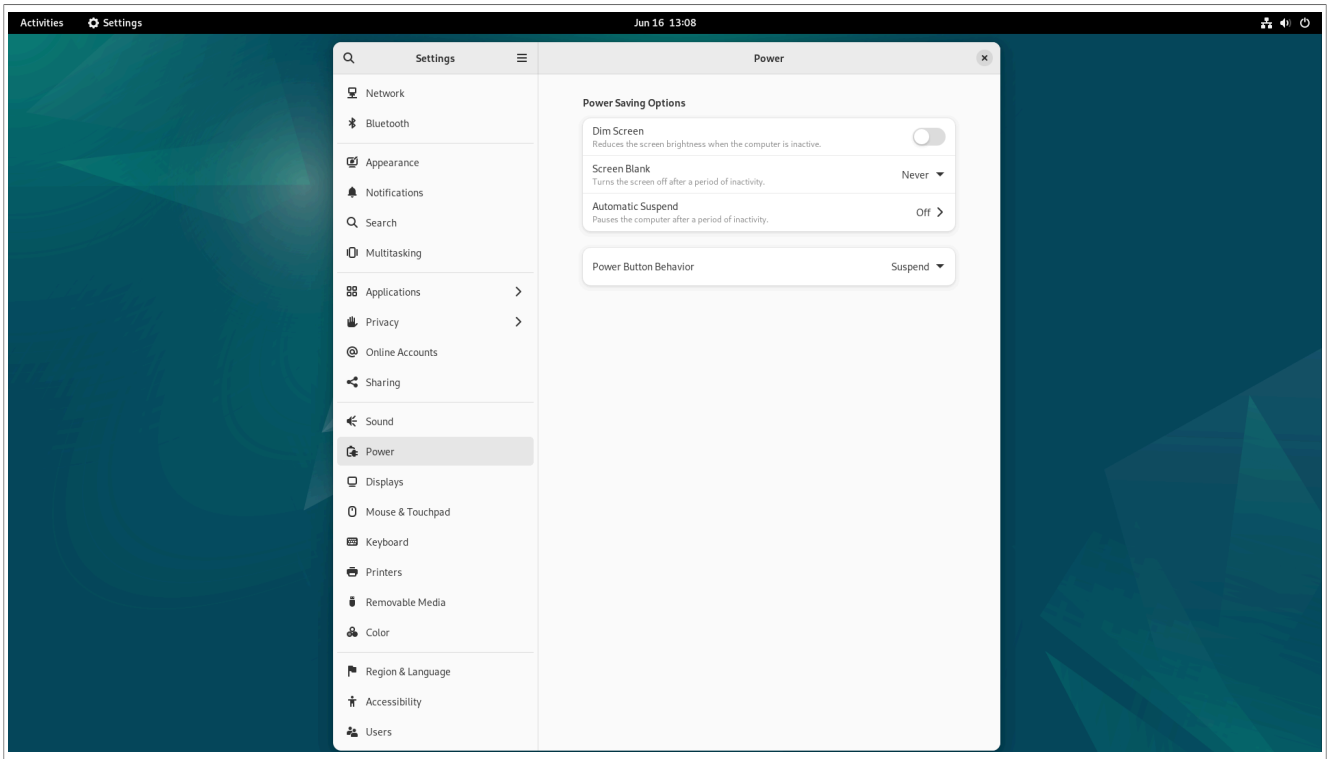

Figure 3. Configuration UI for Debian system Settings

To prevent the Debian system from automatically suspending or dim screen, set **Automatic Suspend** to **Off**, disable **Dim Screen**, and set **Screen Blank** to **Never** if needed.

3.4.1 Camera with Cheese

On the i.MX 8M Plus EVK board, the default dtb file `imx8mp-evk.dtb` is used for camera OV5640. To use Camera OS08A20, change the default dtb file as follows:

```
U-Boot=>  setenv fdtfile imx8mp-evk-os08a20.dtb
U-Boot=>  saveenv;boot
```

- To capture a photo:
  Click the **Cheese** icon, click the **Photo** button, and then click the **Camera** icon. It then takes a photo using the webcam.
- To record a video:
  Click the **Cheese** icon, click the **Video** button, and then click the **Camera** icon. It then records a video using the webcam.

Figure 4. Screenshot of the camera with Cheese on Debian

3.4.2 OpenCL

Run the `glinfo` command to check the OpenCL information as follows.

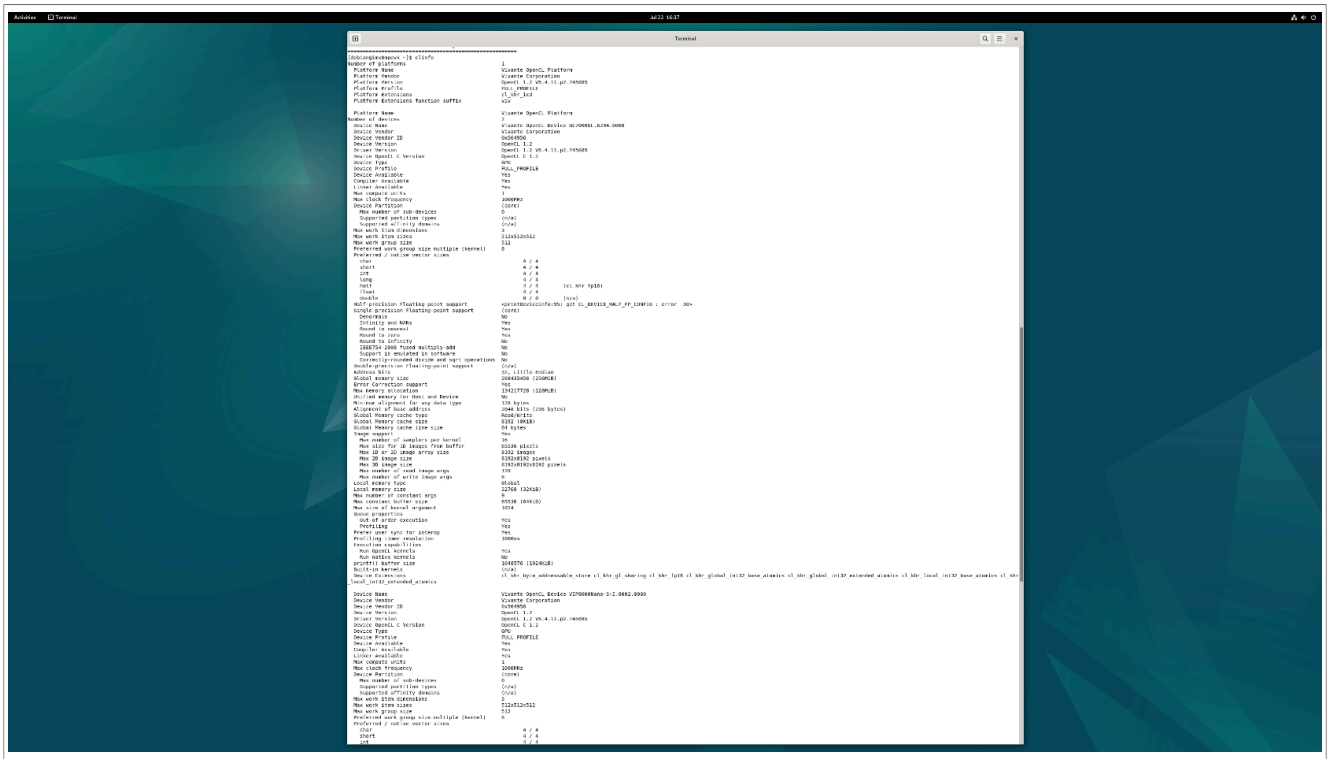

Figure 5. Screenshot of clinfo for OpenCL

3.4.3 OpenGL ES demo with 3D object

Run the `glmark2-es2-wayland` command in the Terminal window to check the OpenGLES demo with a 3D object.

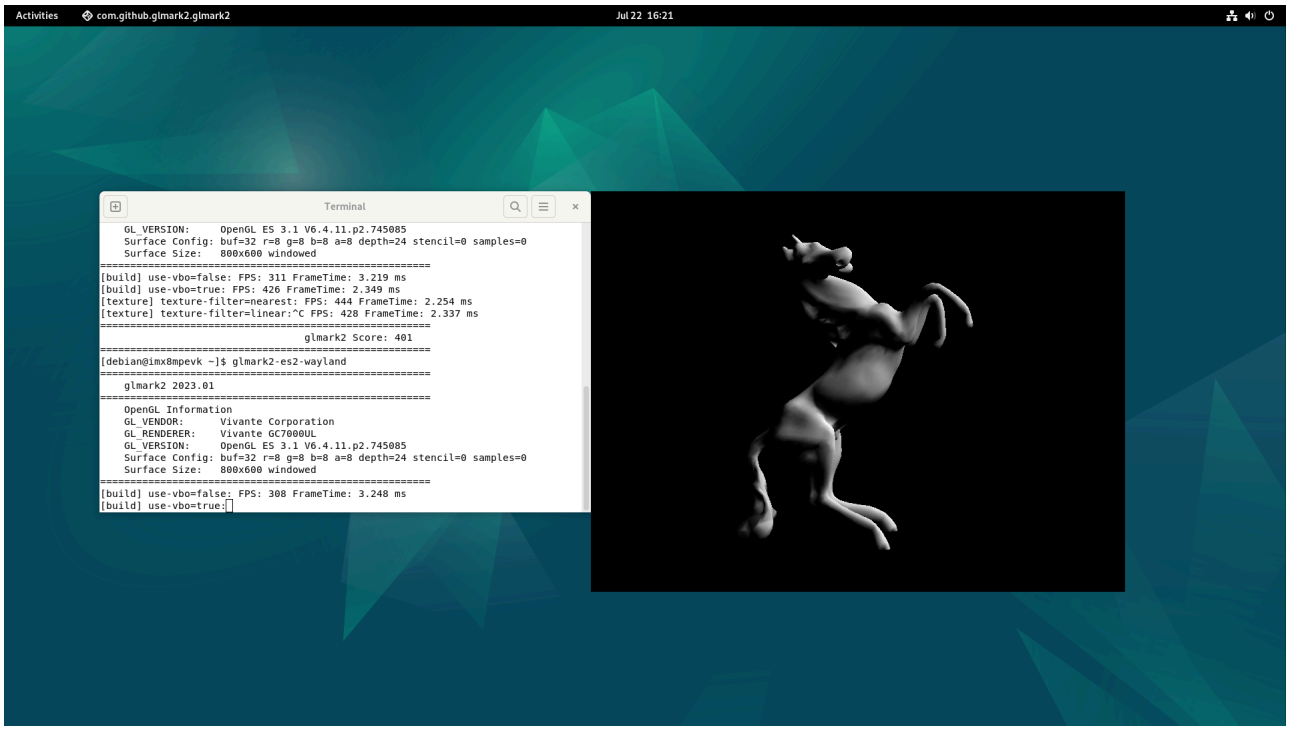

Figure 6. Screenshot of glmark2-es2-wayland for the OpenGLES demo

3.4.4 Video playback and web browser

Users can download a sample of video to the i.MX board and play it by the default Totem video player.

Click the **Chromium web browser** icon to launch the browser to surf the Internet.

The following picture shows a screenshot of running the Totem video playback, Chromium browser, and Terminal window on the Debian 12 Desktop system on the NXP i.MX 8M Plus EVK board.

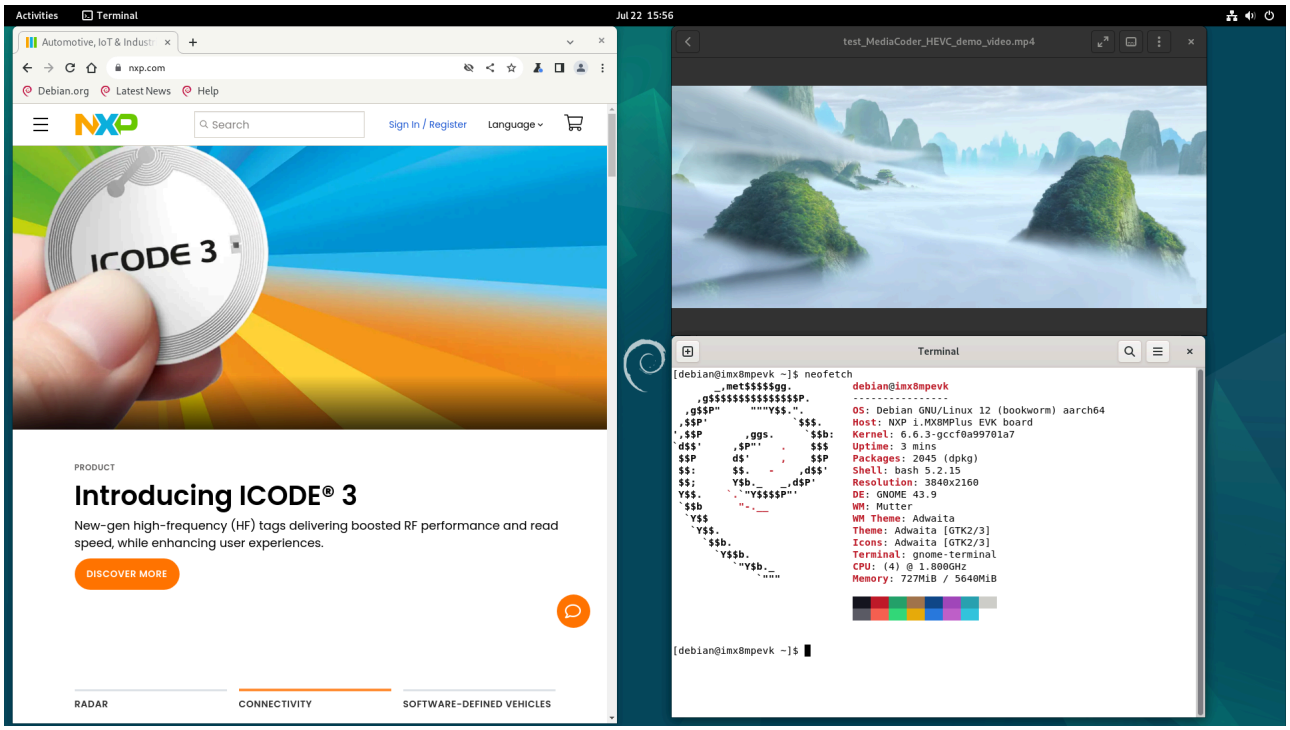

Figure 7. Screenshot of Debian desktop on i.MX

Note:

- *If there is no sound with Headphone audio on the board, click **Settings** -> **Sound**, select the proper **Input Device**, and change **Output Device** to **Headphones - Built-in Audio** if needed.*
- *Sometimes if the cheese application **video record** cannot be stopped normally on i.MX 8M Plus EVK, check **Settings** -> **Sound**, select the proper **Input Device**, and change **Output Device** to **Headphones - Built-in Audio** to ensure that the current pulsesrc device is `alsa_input.platform-sound-wm8960.stereo-fallback` instead of `alsa_input.platform-sound-xcvr.iec958-stereo`.*

3.4.5 NPU with TensorFlow Lite

- On i.MX 8M Plus EVK
  For example, copy `yolov5n-seg_640_float.tflite` to the i.MX 8M Plus EVK board and run the following command:

```
$ /usr/bin/tensorflow-lite-2.16.2/examples/benchmark_model \
  --external_delegate_path=/usr/lib/libvx_delegate.so \
  --graph=~/yolov5n-seg_640_float.tflite
```

- On i.MX 93 EVK
  For example, copy `mobilenet_v1_1.0_224_quant.tflite` to the i.MX 93 EVK board and run the following commands:

```
$ cd /usr/bin/tensorflow-lite-2.16.2/examples
$ vela mobilenet_v1_1.0_224_quant.tflite
$ ./benchmark_model --graph=output/mobilenet_v1_1.0_224_quant_vela.tflite \
  --external_delegate_path=/usr/lib/libethosu_delegate.so
```

3.4.6 GoPoint demos

The following parts of GoPoint demos are supported on Debian on i.MX 8M Plus EVK:

- Image Classification
- Object Detection
- i.MX Smart Kitchen
- i.MX Smart Fitness

To run GoPoint demos, perform the following steps:

1. Log in as debian (not root).
2. Run the following command:

```
mkdir -p /opt/gopoint-apps/downloads
```

3. Run the command `gopoint tui`.
4. Select the correct `/dev/videoX` in the GUI menu.
5. Click **Run**.

Note:

- *GoPoint with Camera OV5640 works on i.MX 8M Plus EVK (Camera OS08A20 not working).*
- *Cheese with Camera OV5640 and OS08A20 works on i.MX 8M Plus EVK.*
- *GoPoint does not work on i.MX 93 because Debian does not support Weston and PXP in this release yet.*

The GoPoint menu is shown as follows.

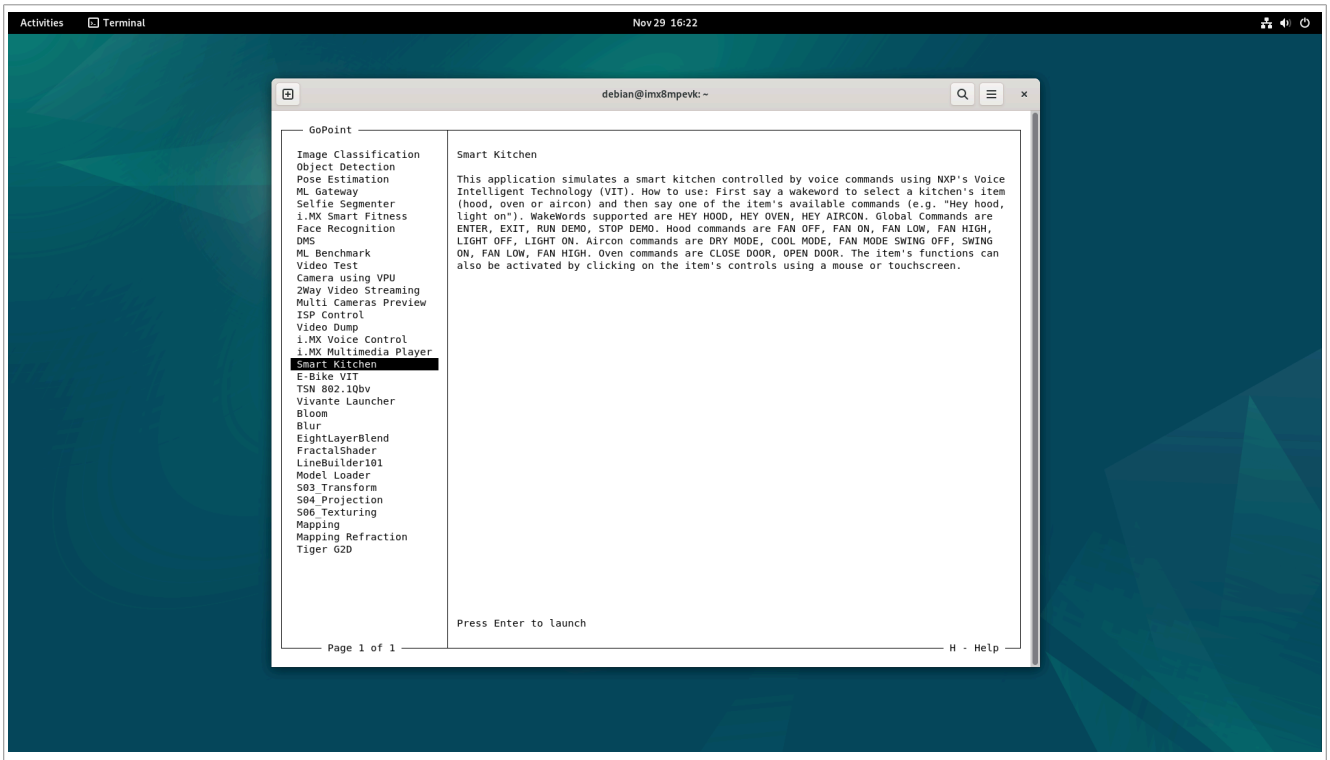

Figure 8. Screenshot of GoPoint Demo memu on i.MX platform

For example, select **Smart Kitchen** and press the **Enter** key. The smart kitchen GUI is shown as follows.

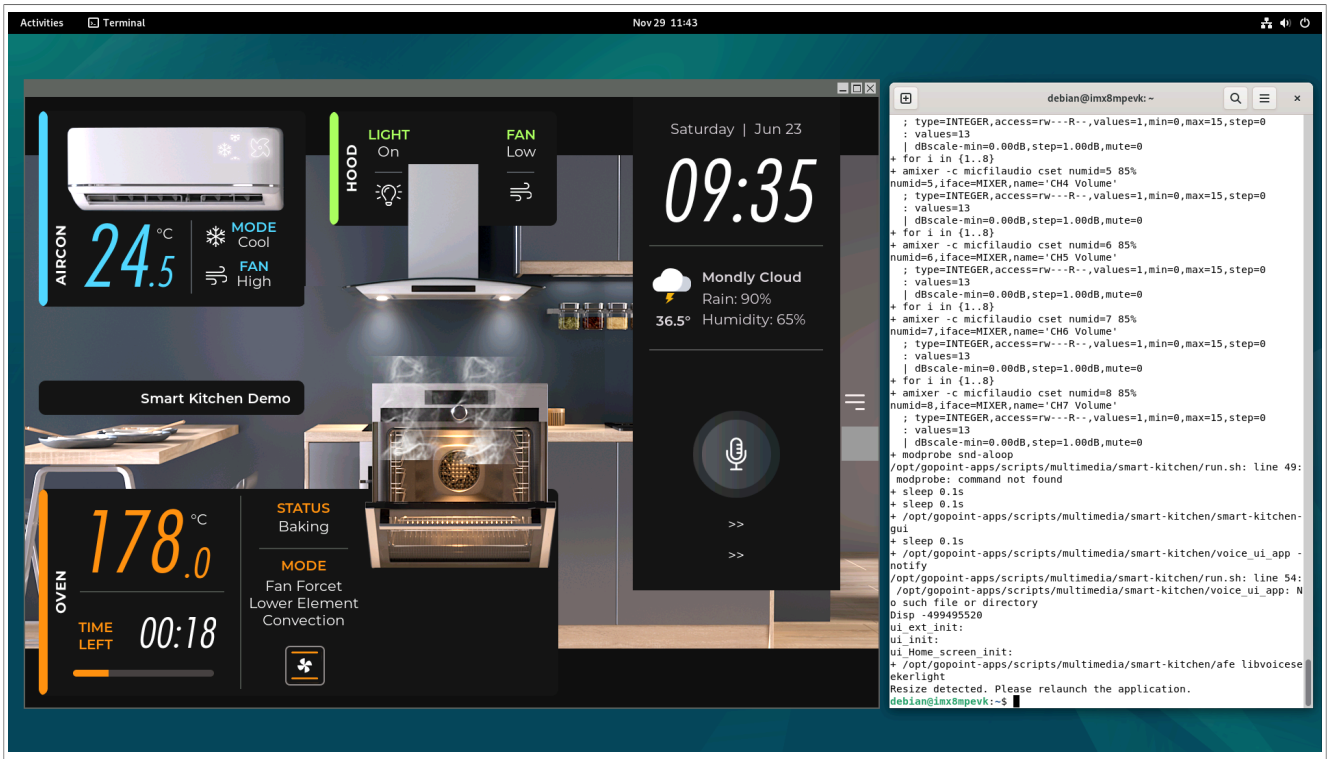

Figure 9. Screenshot of GoPoint Smart Kitchen on i.MX platform

3.4.7 Enabling the Wi-Fi module on the i.MX platform

Run the following commands to set up the Wi-Fi connection.

```
$ modprobe moal mod_para=nxp/wifi_mod_para.conf
(This step loads the Wi-Fi/BT module firmware and it shows the log "wlan: Driver
 loaded successfully")
$ wpa_passphrase <SSID_name> <password> >> /etc/wpa_supplicant.conf
$ wpa_supplicant -d -B -i wlp1s0 -c /etc/wpa_supplicant.conf -Dnl80211
$ dhclient -i wlp1s0
```

Note:

- *After loading the Wi-Fi module firmware, users can use the Wi-Fi GUI in **Debian Settings** to configure Wi-Fi AP/STA. The Wi-Fi AP and Station cannot be used at the same time.*
- *For the AP mode, install the `dnsmasq` package by running `sudo apt install dnsmasq`.*

3.4.8 Qt 6 application on Debian desktop

To support Qt 6 applications, the dependent packages `libqt6core6`, `qt6-base-dev`, and `qt6-wayland` are preinstalled in the i.MX Debian Desktop RootFS by default. Users can build a custom Qt 6 application and put it into Debian desktop RootFS on the target i.MX board. The following picture is a screenshot of the Qt 6 demo application based on the Debian 12 Desktop on the i.MX 8M Plus EVK board.

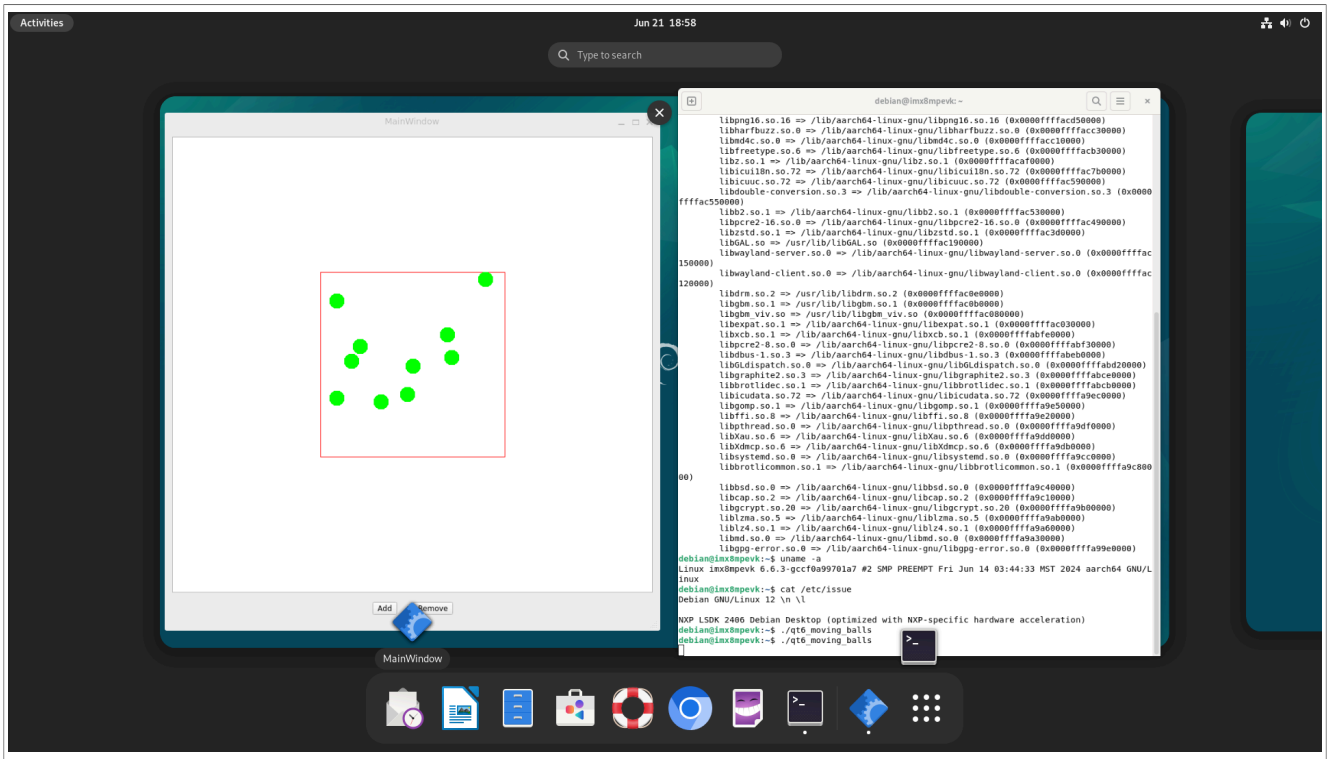

Figure 10. Screenshot of the Qt 6 demo application based on Debian on the i.MX platform

4 Building Debian Images with Flexbuild

4.1 Introduction

Flexbuild is a component-oriented lightweight build system and integration platform with capabilities of flexible, ease-to-use, scalable system building, and Distro deployment, developed by NXP for i.MX platforms.

Flexbuild provides a set of scripts, tools, and make files to compile i.MX-specific driver components, create board-specific BSP composite firmware, boot image, and custom Debian (base, desktop, server) RootFS image. It provides an easy way to create a full-fledged Debian Distro with hardware-accelerated components for i.MX platforms, using a single command. Once the image is built, users can directly deploy it onto an SD card.

4.2 Build environment

Host prerequisites to build the i.MX Debian Distro:

- Option 1: Debian 12 or Ubuntu 22.04 host
  It works to build all components except the eIQ AI/ML components in this build environment.
  It requires a docker container installed on Ubuntu LTS (e.g., 22.04, 20.04) or other Distro host machines to build the eIQ AI/ML components.
- Option 2: Build in a docker
  If Debian 12 or Ubuntu 22.04 host is not available, install the Docker Engine on your Ubuntu or other Distro host machine.

Perform the following steps to install the Docker:

1. Run the following command.

```
sudo apt install docker.io
```

2. Users must have the `sudo` permission for Docker commands or be added to the Docker group as follows. Change the current group to "docker", and add the account to it and restart the Docker service.

```
$ sudo newgrp - docker
$ sudo usermod -aG docker <accountname>
$ sudo gpasswd -a <accountname> docker
$ sudo service docker restart
```

3. Verify that the Docker installation is successful by running the hello-world image.

```
$ docker run hello-world
$ docker ps -a
```

Note:

Linux host machine should be able to access the external Internet in your network environment.

If the Linux host machine is under a subnet that needs the HTTP proxy to access the external Internet, set the HTTP proxy as follows in `/etc/profile.d/proxy.sh` and source it.

```
export http_proxy="http://<domain>:<port>"
export https_proxy="https://<domain>:<port>"
export no_proxy="localhost"
```

4.3 Flexbuild usages

4.3.1 Getting Flexbuild

The Flexbuild repository is hosted at https://github.com/nxp/flexbuild.

Run the following command to clone the repository:

```
$ git clone https://github.com/nxp/flexbuild
```

4.3.2 Flexbuild repository structure

The following is a screenshot of the Flexbuild repository structure.

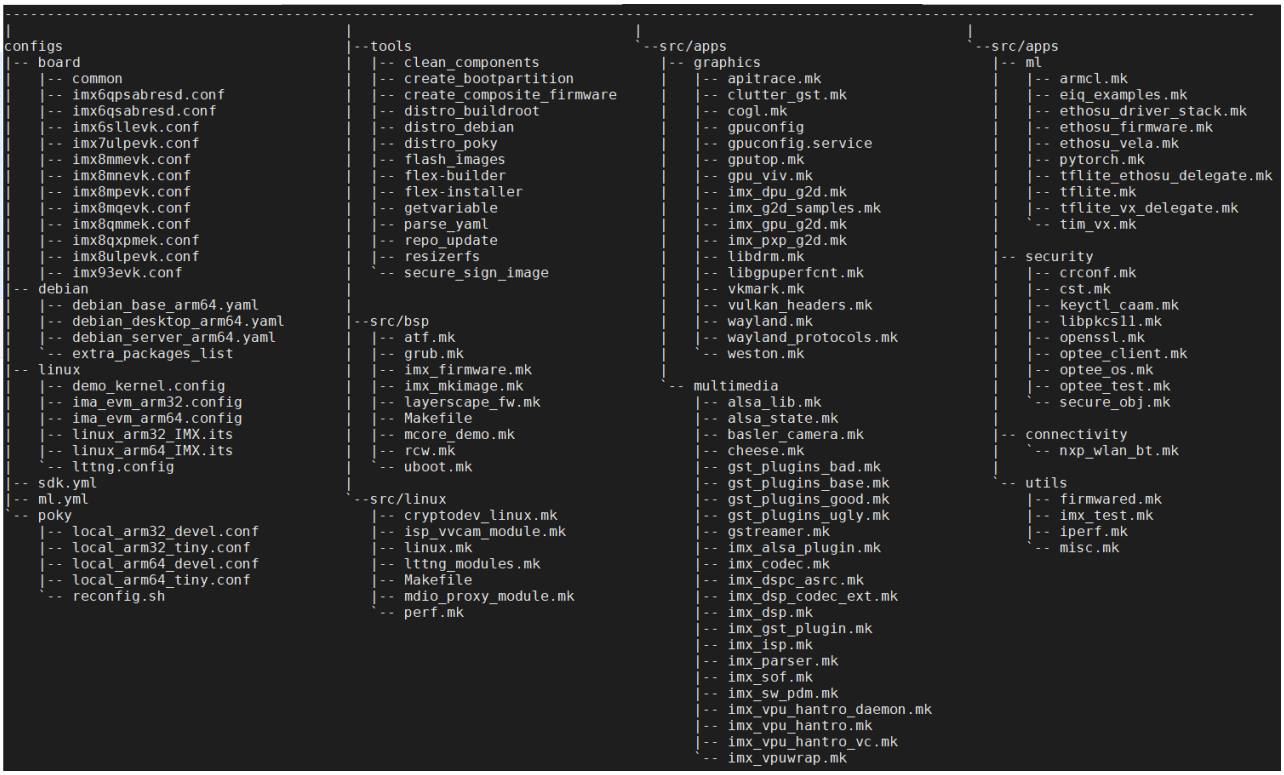

Figure 11. Flexbuild repository structure

4.3.3 Building Debian images in Flexbuild

Run the following commands for the first time to set up the build environment.

```
$ cd flexbuild
$ . setup.env  (in host environment)
$ bld docker   (create or attach to docker)
$ . setup.env  (in docker environment)
$ bld host-dep (install host dependent packages)
```

Flexbuild usage:

```
$ bld -m <machine>
or
```

```
$  bld <target> [ <option> ]
```

Most used examples with automated build:

```
// automatically build BSP composite firmware + kernel + iMX-specific driver
 components + Debian RootFS
$ bld -m imx8mpevk
$ bld -m imx93evk
$ bld -m imx93frdm
```

Most used example with separate build:

```
$ bld uboot -m imx8mpevk
(compile u-boot image for imx8mpevk)

$ bld atf -m imx8mpevk -b sd
(compile ATF image for SD boot on imx8mpevk)

$ bld linux
(compile linux kernel for all arm64 i.MX machines)

$ bld linux:menuconfig
(customize kernel config options in menu)

$ bld linux
(compile kernel based on the custom config)

$ bld boot
(generate boot partition tarball including kernel,dtb,modules,distro bootscript
 for iMX machines)

$ bld bsp -m imx8mpevk
(generate BSP firmware including atf,u-boot,optee_os,kernel,dtb,peripheral-
firmware,initramfs)

$ bld rfs -r debian:desktop
(generate Debian desktop rootfs with more graphics and multimedia packages for
 GUI Desktop)

$ bld rfs -r debian:server
(generate Debian server rootfs with server related packages, no GUI Desktop)

$ bld rfs -r debian:base
(generate Debian  base rootfs with base packages)

$ bld rfs -r poky:tiny
(generate poky-based arm64 tiny RootFS)

$ bld itb -r poky:tiny
(generate itb image including kernel, dtb and poky tiny initramfs)

$ bld itb -r debian:base
(generate itb image including kernel, dtb and debian base RootFS)

$ bld apps -r debian:server
(compile iMX-specific apps against runtime dependencies of Debian server RootFS)

$ bld ml
```

```
(compile eIQ AI/ML components against runtime dependencies of Debian desktop
 RootFS)

$ bld gopoint
(compile GoPoint components against runtime dependencies of Debian desktop
RootFS)

$ bld merge-apps
(merge iMX-specific apps into target Debian desktop RootFS)

$ bld merge-apps -r debian:server
(merge iMX-specific apps into target Debian server RootFS)

$ bld packrfs
(pack and compress target rootfs as rootfs_lsdk_debian_desktop_arm64.tar.zst)

$ bld packapps
(pack and compress target app components as apps_arm64_debian_desktop.tar.zst)

$ bld repo-fetch linux
(fetch git repository of Linux kernel component from remote repos)

$ bld repo-fetch uboot
(fetch git repository of uboot component from remote repos)

$ bld repo-fetch
(fetch git repositories of all components from remote repos)

$ bld security
(build security components for i.MX platforms)

$ bld list
(list enabled machines and supported components)

$ bld docker
(create or attach docker container to build in docker)

$ bld clean
(clean all obsolete firmware/linux/apps images except rootfs image)

$ bld clean-apps
(clean obsolete apps images based on debian desktop)

$ bld clean-apps -r debian:server
(clean obsolete apps images based on debian server)

$ bld clean-rfs
(clean target debian-desktop RootFS, '-r debian:desktop' by default)

$ bld clean-rfs -r debian:server
(clean target debian-server RootFS)

$ bld clean-bsp
(clean obsolete bsp image)

$ bld clean-linux
(clean obsolete linux image)

$ bld dpdk
(build DPDK component based on Debian Desktop for i.MX platforms)
```

```
$ bld graphics
(build graphics components for i.MX platforms)

$ bld multimedia
(build multimedia components for i.MX platforms)

$ bld security
(build security components for i.MX platforms)

$ bld list
(list enabled machines and supported components)
```

4.3.4 How to add or remove a deb package in Flexbuild

Besides adding or removing a deb package by `sudo apt install <package>` or `sudo apt remove <package>` directly on the Debian system on the target i.MX board, users can also add or remove a deb package in/from Flexbuild during the build stage for customization.

If there is already an existing Debian RootFS on the host machine, run the following commands to install a new deb package or remove a deb package:

```
$ sudo chroot build_lsdk2412/rfs/rootfs_lsdk2412_debian_desktop_arm64 apt
 install <package>
$ sudo chroot build_lsdk2406/rfs/rootfs_lsdk2406_debian_desktop_arm64 apt remove
  <package>
$ bld packrfs
(pack the target Debian RootFS as .tar.zst if needed)
```

If there is no Debian RootFS yet on the host machine or you want to clean the old RootFS to rebuild, add a new package name or remove the unneeded package name in/from `configs/debian/debian_desktop_arm64.yaml`, and then run the following commands:

```
$ sudo rm -rf components_lsdk2412/bookworm_desktop_arm64
$ bld clean-rfs
(clean Debian desktop RootFS)
$ bld rfs
(build Debian desktop RootFS with the newly added deb package)
```

Note: *Option `-r debian:desktop` can be omitted by default. Add the option `-r debian:server` for Debian server version.*

4.3.5 How to add a new custom component in Flexbuild

To add a new component called `hello_world`, perform the following steps:

1. Set the relevant URL and tag/commit information for the new component.
   You can edit `configs/sdk.yml` to set the repository URL with a tag or commit for the `hello_world` Git tree if needed.
2. Create a makefile `src/apps/<subsystem>/hello_world.mk` to add the build object for this component.
   According to the various types of the build system (e.g., `make`, `cmake`, `meson`) in the new component, refer to the following examples to add the `hello_world.mk` file.
   - For building with `make`, refer to `src/apps/utils/imx_test.mk`.
   - For building with `cmake`, refer to `src/apps/graphics/gputop.mk`.
   - For building with `meson`, refer to `src/apps/multimedia/cheese.mk`.

3. Build the new component based on the target ARM64 Debian RootFS.

```
$ bld <component> [ -r <distro_type:distro_variant> ]
e.g.
$ bld hello_world
(Add '-r debian:server' for server version, '-r debian:desktop' can be
 omitted by default)
```

4. Merger the new component into the target Debian RootFS.

```
$ bld merge-apps
(Add '-r debian:server' for server version, '-r debian:desktop' can be
 omitted by default)
```

5. Pack the target Debian RootFS.

```
$ bld packrfs
(Add '-r debian:server' for server version, '-r debian:desktop' can be
 omitted by default)
```

Note: *Users can disable group components when they are not needed. For example, change* `PKG_GROUPS_ML: y` *to* ***n*** *in* `configs/sdk.ym` *to disable all eIQ AI/ML components to be compiled by default.*

4.3.6 How to add a new board in Flexbuild

To add a custom i.MX board called `imx8mpabc`, perform the following steps:

1. Fetch the source Git repositories of various components in Flexbuild for the first time.

```
$ git clone https://github.com/nxp/flexbuild
$ cd flexbuild
$ . setup.env  (in host environment)
$ bld docker   (create or attach to docker)
$ . setup.env  (in docker environment)
$ bld host-dep (install host dependent packages)
$ bld repo-fetch uboot
$ bld repo-fetch linux
```

2. (Optional) Add the board-specific BSP related patches for the custom board if needed.
   - Modify or add U-Boot patches in the `components_lsdk2412/bsp/uboot` repository.
   - Modify or add Linux kernel patches in the `components_lsdk2412/linux/linux` repository.
3. Add configs for a custom board in Flexbuild.
   - Add a configuration file in `configs/board/<board>.conf`.
     Copy an existing configuration file of a similar board and make necessary changes in the new `.conf` file.
   - (Optional) Add a node for the new board in `configs/linux/linux_arm64_IMX.its` to generate the `.itb` image.
4. Build the BSP composite firmware image for the new board.

```
$ bld clean-bsp
(optionally, to clean the obsolete bsp images)
$ bld atf -m imx8mpabc -b sd
$ bld uboot -m imx8mpabc -b sd
$ bld linux
$ bld bsp -m imx8mpabc
$ bld boot
```

   This generates the `firmware_imx8mpabc_sdboot.img` and `boot_IMX_arm64_lts.tar.zst` images for the new board.

5. Build application components based on Debian RootFS if needed.

```
$ bld rfs
$ bld apps
$ bld merge-apps
$ bld packrfs
(Add '-r debian:server' for server version, '-r debian:desktop' can be
 omitted by default)
```

6. Deploy the Distro image on the SD card.
   - To install the BSP composite firmware image only onto the SD card, run the following command:

```
$ sudo dd if=firmware_imx8mpabc_sdboot.img of=/dev/mmcblkX bs=1k seek=32
```

   - To install the custom Debian Distro images onto the SD card, run the following commands:

```
$ flex-installer -i pf -d <device>
(partition and format SD card)
$ flex-installer -d <device> -m <machine> -f <firmware> -b <boot> -r
 <rootfs>

e.g.
$ cd build_lsdk2412/images
$ flex-installer -i pf -d /dev/mmcblkX
$ flex-installer -d /dev/mmcblk1 -m imx8mpabc \
                 -f firmware_imx8mpabc_sdboot.img \
                 -b boot_IMX_arm64_lts_6.6.36 \
                 -r ../../build_lsdk2412/rfs/
rootfs_lsdk2412_debian_desktop_arm64

(The path of image can be directory or .tar.zst or .tar.gz format.)
```

7. Boot up Debian on the i.MX board.
   Plug the SD card in the target i.MX board and power it on. It automatically boots the Debian system.
   Under U-Boot, if the automated Distro boot is not supported on the i.MX board, boot it manually by setting the appropriate U-Boot environment.
   (Optional) To boot up the TinyLinux instead of the Debian OS, run the following commands under U-Boot:

```
=> mmc read $load_addr 0x4000 0x1f000 && bootm $load_addr#<board_name>
e.g.
=> mmc read 0xa0000000 0x4000 0x1f000 && bootm a0000000#imx8mpabc
```

5 Related Documentation

For more information about i.MX productions, see the following documentations:

- *i.MX Linux User's Guide* (UG10163)
  Provides information on installing U-Boot and Linux OS and using i.MX-specific features.
- *i.MX Machine Learning User's Guide* (UG10166)
  Provides the machine learning information.
- *i.MX Linux Reference Manual* (RM00293)
  Provides information on Linux drivers for i.MX.
- *i.MX Graphics User's Guide* (UG10159)
  Describes the graphics features.
- *i.MX Porting Guide* (UG10165)
  Provides the instructions on porting the BSP to a new board.
- *i.MX 8M Plus EVK Quick Start Guide* (8MPLUSEVKQSG)
- *i.MX 8M Mini EVK Quick Start Guide* (8MMINIEVKQSG)
- *i.MX 93 EVK Quick Start Guide* (IMX93EVKQSG)

6 Note About the Source Code in the Document

Example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2025 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

7 Revision History

The following table provides the revision history for this document.

Revision history

Document ID	Release date	Description
UG10155 v.LDLSDK_24.12	19 May 2025	Corrected the typo of the revision number from "IDLSDK_24.12" to "LDLSDK_24.12".
UG10155 v.IDLSDK_24.12	30 December 2024	Release for Debian Linux SDK v24.12.
UG10155 v.IDLSDK_24.06	16 August 2024	Initial release for Debian Linux SDK v24.06.

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Bluetooth — the Bluetooth wordmark and logos are registered trademarks owned by Bluetooth SIG, Inc. and any use of such marks by NXP Semiconductors is under license.

eIQ — is a trademark of NXP B.V.

Microsoft, Azure, and ThreadX — are trademarks of the Microsoft group of companies.

TensorFlow, the TensorFlow logo and any related marks — are trademarks of Google Inc.

Contents