AN14473

Run Two Linux Operating Systems in Parallel Using Jailhouse and a RAM Disk on i.MX 95 EVK

Rev. 1.0 — 24 October 2024 Application note

Document information

Information	Content
Keywords	AN14473, Jailhouse, RAM disk, Linux, i.MX 95, Real-Time, Multiple OSes, Dual kernel
Abstract	This application note guides how to run two Linux Operating Systems in parallel, using Jailhouse and a RAM disk.

1 Introduction

This document guides how to run two Linux Operating Systems in parallel, using Jailhouse and a RAM disk.

Jailhouse is a partitioning Hypervisor based on Linux. It can run bare-metal, or other operating systems, along the main Linux operating system. Its main purpose is to ensure resource isolation, by splitting the existing hardware into blocks called **cells**, preventing concurrent access to the same peripheral. The main operating system runs in the **root cell**, while the guest software/operating system runs in the **inmate cell**.

Figure 1 shows the architecture we are trying to achieve. There is a Linux root cell running Linux BSP LF-6.6.36-2.1.0 and a Linux inmate cell running the same kernel. The root cell starts from the SD card and has the exclusive access to several peripherals (for example, four A55 cores, 1.5 G RAM, ETH1, and LPUART4-8). The inmate cell uses a RAM disk for the root file system and have exclusive access to two A55 cores, 7 G RAM, LPUART3, ETH0, and so on. The two cells can communicate through the Messaging Units (MU) and the shared memory (IVSHMEM) implemented by Jailhouse, visible as virtual PCI devices. This communication can be demonstrated with two connected virtual Ethernet devices.

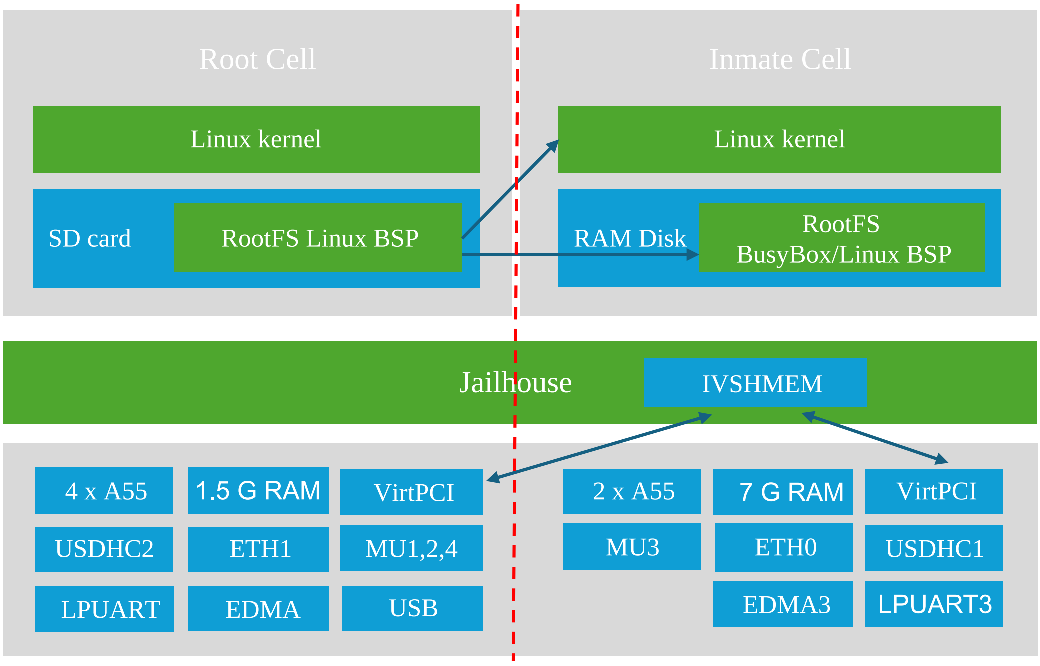

Figure 1. System architecture

Section 2 presents the implementation of this architecture. Section 3 provides additional explanations about the inner workings and configuration.

2 Implementation

2.1 Hardware

- i.MX 95 19 × 19 LPDDR5 EVK
- Ubuntu PC + SD card reader

2.2 Outline

Follow the below steps:

1. Build and write the Linux BSP image on the SD card.
2. Create and add the `initramfs` to the SD card.
3. Start the root cell and the inmate cell.
4. Test the communication.

2.3 Build and write the Linux BSP image

The LF-6.6.36_2.1.0 version is used.

- On the Linux PC, set up the Yocto environment according to **Section 3** - **Section 5** from the *i.MX Yocto Project User's Guide* (document UG10164). Stop at **Section 5.3**, and do not build the image yet.
- Use the Jailhouse configuration for the System Manager. Add in the `conf/local.conf` file the following variant: `IMXBOOT_VARIANT = "jailhouse".`

```
$ echo "IMXBOOT_VARIANT = \"jailhouse\"" >> conf/local.conf
```

- Build the image:

```
$ bitbake imx-image-full
```

- Write the resulted `imx-image-full-imx95-19x19-lpddr5-evk.rootfs.wic` image located in the `tmp/deploy/images/imx95-19x19-lpddr5-evk` directory on the SD card using the following command:

```
$ zstd -fdc imx-image-full-imx95-19x19-lpddr5-evk.rootfs.wic.zst | sudo dd of=/
dev/mmcblk<x> bs=1M status=progress && sync
```

2.4 Create the `ramdisk` image

The inmate cell uses a RAM disk for the root file system. The `ramdisk` image is a compressed (*.gz*) `cpio` archive containing the file system, including an `init` script.

You can either use BusyBox, to create a minimal `rootfs` which provides the most common Linux utilities, or an NXP image, such as core-image-minimal.

2.4.1 BusyBox

To use BusyBox, perform the following steps:

1. Install the Arm cross-compiler toolchain on the PC, using the following command:

```
$ sudo apt install gcc-aarch64-linux-gnu
```

2. Download the latest BusyBox version.

```
$ wget https://busybox.net/downloads/busybox-1.36.1.tar.bz2
$ tar -xvf busybox-1.36.1.tar.bz2
```



```
$ cd busybox-1.36.1
```

3. Run the `menuconfig` and make sure to enable `Settings ---> Build static binary (no shared libs)`

```
$ ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- make menuconfig
```

4. Save the new configuration and build the sources.

```
$ ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- make -j $(nproc --all)
$ ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- make install
```

The BusyBox binaries are installed in the `_install` directory.

5. Create the structure of the root filesystem.

```
$ cd ..
$ mkdir initramfs
$ mkdir -p initramfs/bin initramfs/sbin initramfs/etc initramfs/proc
 initramfs/sys initramfs/dev initramfs/usr/bin initramfs/usr/sbin
$ cp -a busybox-1.36.1/_install/* ./initramfs
```

6. Create the `init` script `initramfs/init`.

```
#!/bin/sh
mount -t devtmpfs devtmpfs /dev
mount -t proc none /proc
mount -t sysfs none /sys
exec /bin/sh
```

7. Make the script executable.

```
$ chmod +x initramfs/init
```

8. Add any additional files and executables that you may need in the `initramfs` directory.
9. Create the `initramfs` archive.

```
$ cd initramfs
$ find . -print0 | cpio --null -ov --format=newc | gzip -9 > ../
initramfs.cpio.gz
$ cd ..
```

10. Copy the resulting archive `initramfs.cpio.gz` onto the root partition of the SD card, in `/root`.

2.4.2 NXP's core-image-minimal

To use an NXP image, perform the following steps:

1. Add to the `conf/local.conf` file the `cpio.gz` image type:

```
$ echo "IMAGE_FSTYPES:append = \" cpio.gz\"" >> conf/local.conf
```

2. Build the minimal `rootfs`.

```
$ bitbake core-image-minimal
```

3. Go to the deployment directory `tmp/deploy/images/imx95-19x19-lpddr5-evk/`.
4. Copy the resulting archive `core-image-minimal-imx95-19x19-lpddr5-evk.rootfs.cpio.gz` onto the root partition of the SD card, in `/root`.

2.5 Run the inmate Linux cell

Jailhouse is already built into the NXP Linux kernel. There is no need to compile it separately.

1. Connect the USB debug port of the board to the PC using a USB cable. This operation creates four virtual serial ports on the PC. Open all in a terminal emulator using the following parameters: 115200 baud rate, 8 data bits, no parity, and 1 stop bit. One is the console for the root cell, one for the inmate cell, and one for the System Manager.
2. Boot the board, stop it in U-Boot, and run the `jh_mmcboot` command. This command sets the kernel device tree as `imx95-19x19-evk-root.dtb`, and limits the memory space used by the main kernel, then boots the Linux kernel. The `imx95-19x19-evk-root.dtb` device tree disables the peripherals used by the inmate cell.

```
u-boot => run jh_mmcboot
```

3. Run the inmate Linux cell:

```
root@imx95evk:~# export PATH=$PATH:/usr/share/jailhouse/tools/
root@imx95evk:~# modprobe jailhouse
root@imx95evk:~# jailhouse enable /usr/share/jailhouse/cells/imx95.cell
```

If the BusyBox is used, run:

```
root@imx95evk:~# jailhouse cell linux /usr/share/jailhouse/cells/imx95-linux-demo.cell /run/media/boot-mmcblk1p1/Image -d /run/media/boot-mmcblk1p1/imx95-19x19-evk-inmate.dtb -i initramfs.cpio.gz -c "clk_ignore_unused
 console=ttyLP2,115200 earlycon=lpuart32,mmio32,0x44380010,115200"
```

If the `core-image-minimal` is used, run:

```
root@imx95evk:~# jailhouse cell linux /usr/share/jailhouse/cells/imx95-linux-demo.cell /run/media/boot-mmcblk1p1/Image -d /run/media/boot-mmcblk1p1/imx95-19x19-evk-inmate.dtb -i core-image-minimal-imx95-19x19-lpddr5-evk.rootfs.cpio.gz -c "clk_ignore_unused console=ttyLP2,115200
 earlycon=lpuart32,mmio32,0x44380010,115200 rdinit=/sbin/init"
```

4. At this point, the second Linux prompt can be seen on one of the serial ports opened at Step 1.
5. To test the network communication via the virtual Ethernet devices, assign an IP address to the available Ethernet interface: **eth2** in the root cell and **eth0** in the inmate cell. You can use the `ping` command to test the communication.

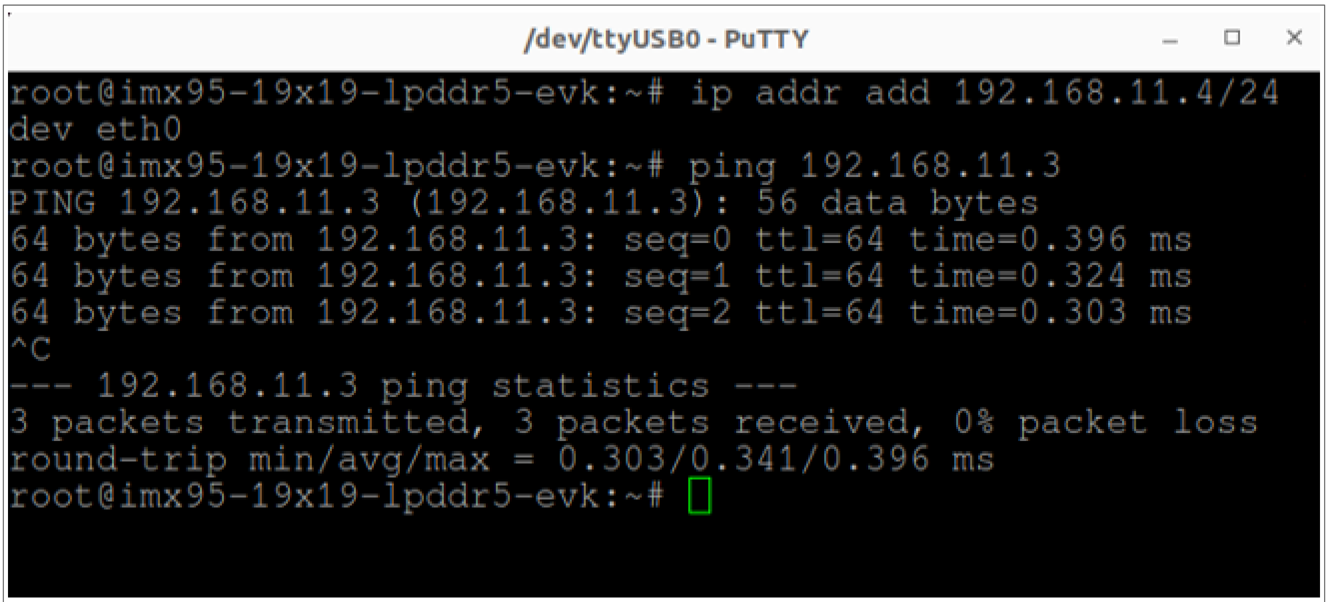

Figure 2. Setup in the inmate cell

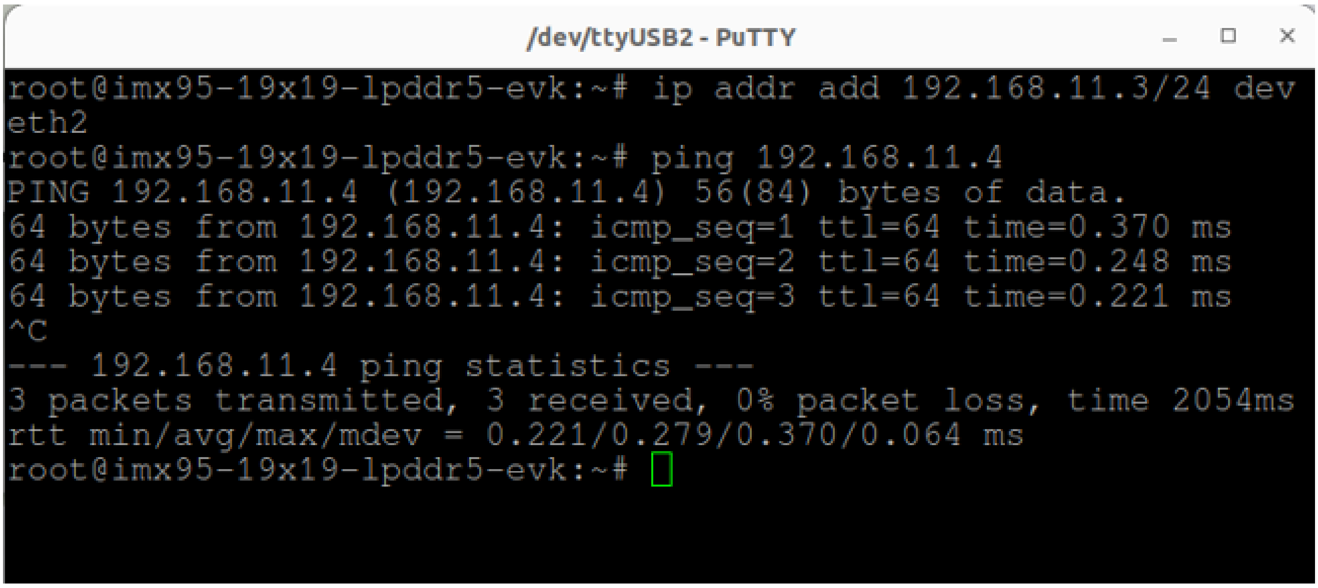

Figure 3. Setup in the root cell

3 Configuration and tuning

During the initial boot, the kernel is started with a special device tree (`imx95-19x19-evk-root.dtb`) for the root cell, which disables the devices that are later used by the inmate cell. Initially, the kernel uses all six cores. After enabling Jailhouse, the hypervisor moves the Linux into the root cell, but still using all the cores. When creating the inmate cell, the hypervisor partitions the hardware, so that each cell only has access to the hardware assigned in the configuration. It disables two cores from the root cell and assigns them to the inmate cell.

The device tree used for the inmate cell is `imx95-19x19-evk-inmate.dtb`. In it, are configured the peripherals that the inmate cell can access (for example USDHC1 and LPUART3). These devices are disabled from the root cell device tree (`imx95-19x19-evk-root.dtb`). For example, to use the USDHC1 from the root cell, disable the USDHC1 from the inmate cell device tree, and enable it in the root device tree.

In the imx-jailhouse project, there are some files of interest:

- `configs/arm64/*` - root/inmate cell configurations: peripheral allocation and isolation.
  - `imx95.c` - root cell configuration.
  - `imx95-linux-demo.c` - inmate cell configuration.

  These files are compiled into binaries with the `.cell` extension.
- `tools/` - executable programs to configure and command the Jailhouse hypervisor.
  - `jailhouse enable <sysconfig.cell>` - starts the Jailhouse hypervisor and wraps the running Linux into the root cell.
  - `jailhouse cell [collect | create | destroy | linux | load | shutdown | start | stats] <args>` - controls the cells. For more details about each command, check the Jailhouse Documentation.

Cell configuration file explained

Each `(non-)root` cell is statically configured through a `*.c` file, describing which hardware resources the cell can access. The configuration parameters are assigned through some predefined structures implemented in the `include/jailhouse/cell-config.h` file. The root cell structure must have the `struct jailhouse_system` in the header, while the non-root cell structure must have the `struct jailhouse_cell_desc`. Some configurations of interest are commented below:

```
.cpus = {
      0x18, /* the mask of cores to be used: 011000 => CPU3 & CPU4 */
},
/*memory regions to which the cell has access and with which rights (flags)*/
.mem_regions = {
      ...
      /* lpuart3 */  // Example of allocating LPUART3 exclusive access
      {
            .phys_start = 0x42570000,
            .virt_start = 0x42570000,
            .size = 0x1000,
            .flags = JAILHOUSE_MEM_READ | JAILHOUSE_MEM_WRITE |
            JAILHOUSE_MEM_IO,
      },
      ...
}
.irqchips = {
      {
            /* lpuart3/usdhc1 */
            .address = 0x48000000,
            .pin_base = 32,
            .pin_bitmap = {
                  // 86 = USDHC1 interrupt number
                  // 64 = LPUART3 interrupt number
                  0, 0, (1 << (86 + 32 - 32 - 64)) | (1 << (64 + 32 - 32 - 64)),
 0
                  // interrupts 0-31, 32-63, 64-95, 96-127
            },
      },
      ...
},
```

The communication between the cells is ensured via shared memory using virtual Inter-VM Shared Memory (ivshmem) PCI devices implemented by Jailhouse. Most of the memory regions described in the file above are pertaining to ivshmem. For more details, please check the ivshmem documentation.

4 Note about the source code in the document

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

Copyright 2024 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 must be 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.

5 Revision history

Table 1 summarizes the revisions to this document.

Table 1. Revision history

Document ID	Release date	Description
AN14473 v.1.0	24 October 2024	Initial public release

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.

AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, μVision, Versatile — are trademarks and/or registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved.

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

Contents