IMXDCHPE

i.MX Digital Cockpit

Hardware Partitioning Enablement for i.MX 8QuadMax

Rev. LF5.10.35_2.0.0 — 21 June 2021

User's Guide

Contents

1 Introduction......................................1
2 System Boot Sequence...................3
3 Linux BSP Integration..................... 6
4 Signing boot images in Digital Cockpit... 6
5 SD/eMMC Image Flashing.............. 8
6 References.......................................9
7 Revision History............................ 10

1 Introduction

The main feature of Cockpit project is the hardware isolation.

Each OS has its own hardware resources.

One partition is running on Arm Cortex-A53 cluster for Instrument Cluster, and one is running on Arm Cortex-A72 cluster for In-Vehicle Infotainment, while two others are running RTOS on Arm Cortex-M4s.

Instrument Cluster's HLOS uses SD card in the CPU board slot as boot media.
In-Vehicle Infotainment's HLOS uses eMMC as boot media. For this reason, two SD card images must be built:

- One with a standard Linux BSP, used to flash boot image and HLOS images for Cortex-A72 on eMMC.
- One used at runtime for booting the board, this card contains firmware, HLOS images for Cortex-A53 and Cortex-M4s.

All BSP components are modified to allow running two HLOS and two RTOS in a secure way.

- SCU Firmware board file
- Arm Trusted Firmware
- U-Boot/SPL
- mkimage
- Linux Kernel
- TEE

The Cockpit architecture provides a way to implement two different ECU software systems in a single SoC ECU.

The ECU software systems enforce different safety levels, so it is important to isolate them from each other to avoid interference. The Digital Cockpit concept implements strong hardware isolation without the need for a Hypervisor.

Hardware isolation is under the control of the System Controller Unit (SCU) secured with Extended Resource Domain Controller (XRDC).

This isolation is implemented by:

- Defining hardware partitions, each with its own resources.
- Defining Memory regions.
- Assigning IO pads to hardware partitions.
- Routing IRQs to their respective subsystem.

The following figure shows how hardware resources are allocated between the two partitions in this release.

Resources allocated to Cortex-A53 cluster (IC) are colored in red, those allocated to Cortex-A72 cluster (IVI) are in blue, and the resources uncolored are assigned by default to Cortex-A53's partition.

LPUART0 is assigned to the Cortex-A53 cluster, whereas LPUART2 is assigned to the Cortex-A72 cluster.

Each cluster has one identical display controller and one identical GPU, GC7000.

Only one GIC is available and allocated to SCU but made accessible to both Clusters. Interrupts dedicated to a Cortex-A cluster are only routed to it.

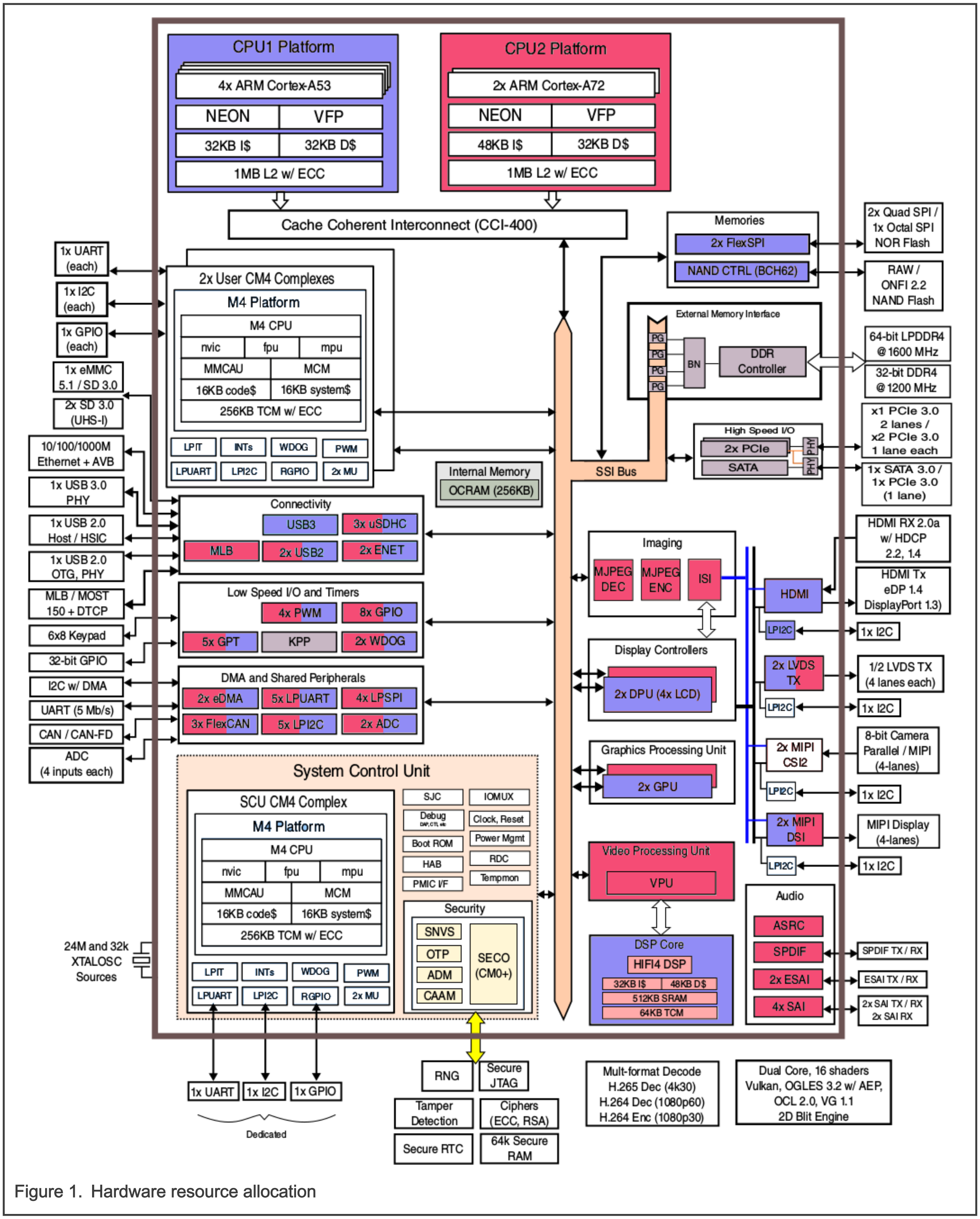

Figure 1. Hardware resource allocation

2 System Boot Sequence

2.1 System boot sequence

The partitioning for the subsystems is defined at boot time and is immutable after the system is booted.

The following sequence diagram shows the **Secure Boot** flow across different CPUs.

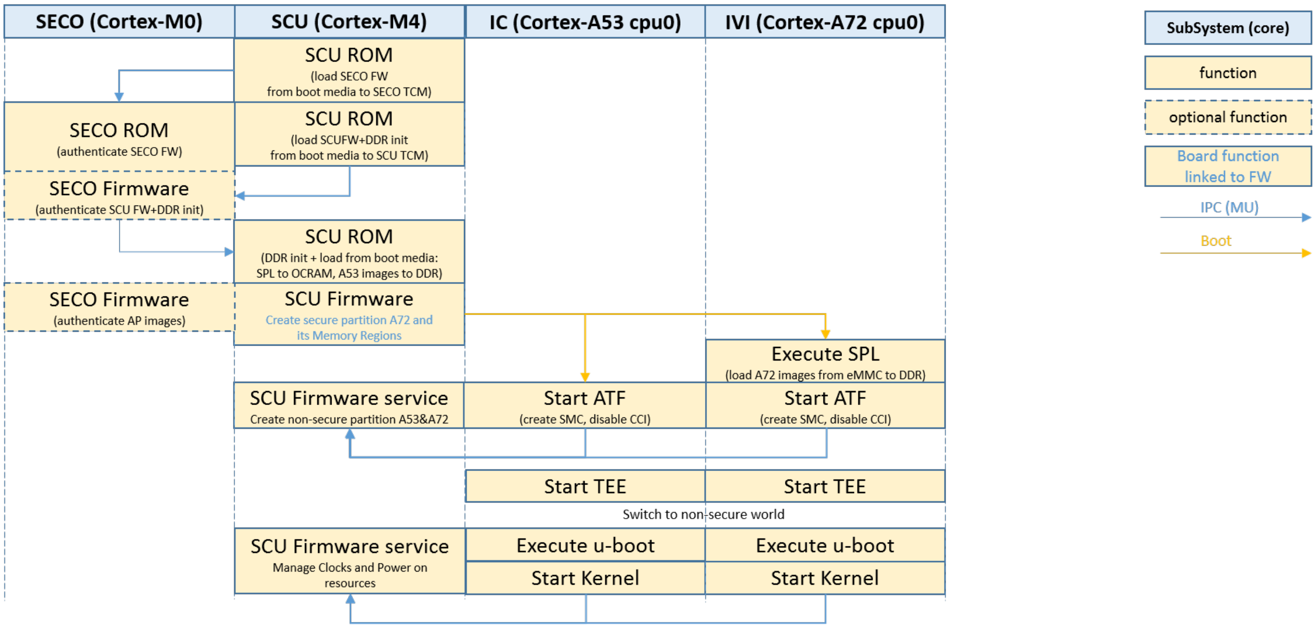
SECO (Cortex-M0)
SCU (Cortex-M4)
IC (Cortex-A53 cpu0)
IVI (Cortex-A72 cpu0)
SCU ROM
(load SECO FW from boot media to SECO TCM)
SECO ROM
(authenticate SECO FW)
SCU ROM
(load SCUFW+DDR init from boot media to SCU TCM)
SECO Firmware
(authenticate SCU FW+DDR init)
SCU ROM
(DDR init + load from boot media: SPL to OCRAM, A53 images to DDR)
SECO Firmware
(authenticate AP images)
SCU Firmware
Create secure partition A72 and its Memory Regions
Execute SPL
(load A72 images from eMMC to DDR)
SCU Firmware service
Create non-secure partition A53&A72
Start ATF
(create SMC, disable CCI)
Start ATF
(create SMC, disable CCI)
Start TEE
Start TEE
Switch to non-secure world
SCU Firmware service
Manage Clocks and Power on resources
Execute u-boot
Execute u-boot
Start Kernel
Start Kernel
SubSystem (core)
function
optional function
Board function linked to FW
IPC (MU)
Boot

Figure 2. Boot sequence

2.2 Boot details

2.2.1 SECO boot

The images that will be loaded as part of the boot process are stored in containers. A container contains multiple images, as blocks of data specifying the load address. The containers can be authenticated, see Section Signing boot images in Digital Cockpit (e.g., in certain lifecycle states, the SECO FW needs to be authenticated, but the SCU FW can be loaded even it is not signed).

At least three containers are needed for the boot process:

- The first container with the SECO FW image only. This container is NXP signed, and NXP provided.
- The second container (can be signed by customer using OEM SRKs) that typically contains:
  - SCU FW
  - DDR initialization image (optional)
  - Images for the user-controlled Cortex-M4s (optional)
  - The TEE image for Cortex-A53 (optional)
  - The U-Boot-SPL for Cortex-A72 cluster.
  - Boot image for each Cortex-A53 cluster, containing:
    - Arm Trusted Firmware (BL31)
    - U-boot

- The third container (can also be signed by customer using OEM SRKs) that contains:
  - TEE image for Cortex-A72 (optional)
  - Boot image for Cortex-A72 cluster, containing:
    - Arm Trusted Firmware (BL31)
    - U-Boot

The second and third containers containing Cortex-A boot images can be encrypted, but in this release, they are not encrypted by default.

2.2.2 SCU boot

SCU ROM supports the following boot devices:

- FlexSPI NOR Flash (on FlexSPI0)
- NAND Flash
- SD/MMC
- USB

The current release boots from the SD card slot and eMMC located on the CPU board.

In the i.MX 8QuadMax processor B0 revision, the SCU ROM loads the following from the SD card:

1. Cortex-A53's boot image at 0x8000_0000 in DDR (ATF, TEE, U-Boot)
2. Cortex-A72 SPL at address 0x0 in OCRAM
3. Cortex-M4s images in their respective TCM or into DDR RAM.

The Cortex-A72 boots at address 0x0 and executes SPL, which then loads the Cortex-A72's images (ATF, TEE, U-Boot) at 0xC000_0000 from eMMC.

Alternatively, it is possible to revert to a legacy boot mode (without SPL and TEE) where SCU ROM loads the two Cortex-A boot images (ATF and U-Boot) from the SD card, respectively Cortex-A53's image at 0x8000_0000, and Cortex-A72's image at 0xC000_0000 in DDR RAM. In this legacy mode, the TEE for Cortex-A72 cannot be loaded, as its address (0xFE00_0000) is not reachable by SCU ROM.

SCU enforces the hardware partitioning by setting the Hardware Resources, Memory Regions and IO pads.

The partitioning is done in the *Board System Config* phase of the SCU Firmware initialization, which is implemented in the board specific customizable source file. At this step, a second secure partition is created for the Cortex-A72 cluster.

NOTE

Each ATF instance will later create a non-secure partition and assign most resources (hardware peripheral and hardware accelerators) to it.

SCU Firmware releases Cortex-A53 from reset at address 0x8000 0000, and executes *Board Init* phase, which is also implemented in the board specific customizable source file. Similarly, Cortex-A72 is released from reset at 0x0 (OCRAM SPL) or 0x80000000 depending on compile option of SCFW (NOSPL=1).

2.2.3 Hardware partitions

Table 1. Hardware partitions

Partition number	Secure/non-secure	Comments
0	N/A	Contains all hardware resources at SCU boot

Table continues on the next page...

Table 1. Hardware partitions (continued)

Partition number	Secure/non-secure	Comments
1	Secure	All hardware resources are given to this partition used by Cortex-A53 cluster
2	Secure	Contains Crypto, CAAM resources, used by SECO
3	Secure	Contains all the hardware resources that are removed afterwards from Cortex-A53 partition 1, for Cortex-A72 cluster
4	Non-secure	Contains only a shared memory region for Cortex-A53 to Cortex-A72 communication
5	Non-secure	Contains all the hardware resources required by Cortex-M4 core 0
6	Non-secure	Contains all the hardware resources required by Cortex-M4 core 1
7	Non-secure	Contains only a shared memory region for Cortex-A53 to Cortex-M4 communication
8	Non-secure	Depending on boot order, contains all hardware resources for Cortex-A53 or Cortex-A72 cluster
9	Non-secure	Depending on boot order, contains all hardware resources for Cortex-A53 or Cortex-A72 cluster

2.2.4 Cortex-A53 (IC) boot details

SCU ROM loads ATF and U-Boot for Cortex-A53 cluster from boot media into RAM.

Cortex-A53 is booting first as it runs the Instrument Cluster, which has a faster boot time requirement than Infotainment.

Cortex-A53 core#0 is booting at address 0x8000 0000 where ATF and U-Boot resides.

ATF is a secure partition and it creates a non-secure partition moving most of the hardware resources to this partition through the SCUFW service.

U-Boot loads the Linux kernel from the SD card, and then executes it.

2.2.5 Cortex-A72 (IVI) boot details

SCU ROM loads SPL for the Cortex-A72 cluster from the boot media into OCRAM.

Cortex-A72 core#0 is booting at address 0x0, an alias for OCRAM address where SPL resides. Cortex-A72 executes SPL, which loads TEE at 0xFE00_0000, loads ATF and U-boot at address 0xC000_0000, and then Cortex-A72 jumps to ATF. Similar to Cortex-A53, ATF creates a non-secure partition and moves most of the hardware resources to this partition through SCUFW service.

2.2.6 Cortex-M4s (RPMSG demo) boot details

SCU ROM loads a FreeRTOS image on both M4_0 and M4_1 from boot media into their respective TCM.

The Cortex-M4 images contains a demonstration of a simple “ping-pong” communication with Cortex-A53 using an RPMSG channel. Those Cortex-M4 images are pre-built images downloaded by the Yocto recipe.

3 Linux BSP Integration

Follow the instructions on the meta-cockpit README in meta-imx layer for integrating cockpit into a Linux BSP build.

3.1 Digital Cockpit hardware requirements

- 1 x Linux Host Machine with a minimum 120 GB HDD space available and Internet connection
- 1 x MCIMX8QM-CPU revision B0 board
- 1 x MCIMX8-8X-BB
- 1 x SD card (tested on 16 GB SD card)
- 3 x HDMI displays
- 2 x IMX-LVDS-HDMI
- 1 x IMX-MIPI-HDMI (optional)

4 Signing boot images in Digital Cockpit

See the Secure Boot on i.MX 8 and i.MX 8X Families using AHAB for a detailed explanation of the signing procedure.

The CST tool version 3.3.0 is required for this release. Download and install the CST tool before proceeding. The following instructions assume that the environment variable $CST contains the CST tool location (ex: `export CST=/home/user/CST/release/linux64/bin/cst`).

The image signing step must be performed **after** the generation of the boot image by the tool 'mkimage'.

4.1 Signing SCFW and Cortex-A53 boot image

Parse the log of the building step of boot image (located at `yocto/build/tmp/work/imx8qmmekcockpit-poky-linux/imx-boot/1.0-r0/temp/log.do_compile`), and search for offset values on the same line as '*CST: CONTAINER 0*' after the key word Output: flash.bin.

As recommended in Secure Boot document, prepare a CST configuration file named `csf_boot_image.txt`.

Store the previously found offset values in this file.

The following is an example of resulting configuration file:

```
[Header]
Target = AHAB
Version = 1.0
[Install SRK]
# SRK table generated by srktool
File = "/home/b47544/cockpit/official/CST/release/crts/SRK_1_2_3_4_table.bin"
# Public key certificate in PEM format
Source = "/home/b47544/cockpit/official/CST/release/crts/SRK1_sha384_secp384r1_v3_usr_crt.pem"
# Index of the public key certificate within the SRK table (0 .. 3)
Source index = 0
# Type of SRK set (NXP or OEM)
Source set = OEM
# bitmask of the revoked SRKs
Revocations = 0x0
[Authenticate Data]
# Binary to be signed generated by mkimage
File = "iMX8QM/flash.bin"
# Offsets = Container header  Signature block (printed out by mkimage)
Offsets   = 0x400             0x710
```

Put your file in the mkimage folder for the i.MX 8QuadMax, and run the following command:

```
$CST -i iMX8QM/csf_boot_image.txt -o iMX8QM/signed-boot-spl-container-a53.img
```

Then replace the old boot image with the signed one:

```
mv iMX8QM/signed-boot-spl-container-a53.img iMX8QM/flash.bin
```

4.2 Signing the Cortex-A72 boot image

By default, the Cortex-A72 boot image container (containing ATF, U-Boot, and OP-TEE) is loaded and authenticated by U-Boot-SPL executed by Cortex-A72.

CAUTION

A limitation in the AHAB firmware version 2.6.1 and older versions does not allow authenticating the boot image by SPL-A72 (as Cortex-A72 does not belong to domain ID 0 or 1). For those AHAB versions, if authentication is needed, it is recommended to switch to a legacy boot mode where Cortex-A72 and Cortex-A53 images are both loaded and authenticated by SCU (domain ID 0).

To enable the authentication by U-Boot-SPL-A72, modify the configuration file `<yocto>/build/tmp/work/imx8qmmekcockpit-poky-linux/u-boot-imx-a72/2021.04-r0/git/configs/ imx8qm_mek_cockpit_a72_defconfig` and add the line CONFIG_AHAB_BOOT=y.

Parse the log of the building step of boot image (located at `yocto/build/tmp/work/imx8qmmekcockpit-poky-linux/imx-boot/1.0-r0/temp/log.do_compile`), and search for offset values on the same line as ‘*CST: CONTAINER 0*’ after the key word `Output: u-boot-atf-container-a72.img`.

As recommended in Secure Boot document, prepare a CST configuration file named `cst_uboot_atf_a72.txt`.

Store the previously found offset values in this file.

The following is an example of resulting configuration file:

```
[Header]
Target = AHAB
Version = 1.0
[Install SRK]
# SRK table generated by srktool
File = "/home/b47544/cockpit/official/CST/release/crts/SRK_1_2_3_4_table.bin"
# Public key certificate in PEM format on this example only using SRK key
Source = "/home/b47544/cockpit/official/CST/release/crts/SRK1_sha384_secp384r1_v3_usr_crt.pem"
# Index of the public key certificate within the SRK table (0 .. 3)
Source index = 0
# Type of SRK set (NXP or OEM)
Source set = OEM
# bitmask of the revoked SRKs
Revocations = 0x0
[Authenticate Data]
# Binary to be signed generated by mkimage
File = "iMX8QM/u-boot-atf-container-a72.img"
# Offsets = Container header  Signature block (printed out by mkimage)
Offsets   = 0x0             0x110
```

Put your file in the mkimage folder for the i.MX 8QuadMax, and run the following command:

```
$CST -i iMX8QM/cst_uboot_atf_a72.txt -o iMX8QM/signed-u-boot-atf-container-a72.img
```

Then append the signed Cortex-A72 container after the Cortex-A53 image container using the following script:

```
flashbinsize=$(wc -c iMX8QM/signed-boot-spl-container-a53.img | awk '{print $1}')
padcnt=$((($flashbinsize + 0x400 - 1) / 0x400)) dd if=iMX8QM/signed-u-boot-atf-container-a72.img
of=iMX8QM/flash.bin bs=1K seek=$padcnt
```

5 SD/eMMC Image Flashing

This chapter explains how to flash the two OSes on the SD card and eMMC.

5.1 Flashing dual-Linux OS images

To simplify the build and flashing process, the same generated image can be used for the two OSes: this is possible because Cortex-A53 only loads the first and second containers of the boot image, while Cortex-A72 only loads the third container of the boot image.

The result of the build process is a compressed image which can be found in the following location:

```
tmp/deploy/images/imx8qmmekcockpit/imx-image-<type>-imx8qmmekcockpit-<timestamp>.wic.bz2
```

where <type> is one of *multimedia* or *full*, depending on whether you've built an image with or without OpenCV + QT5 + Machine Learning packages, and <timestamp> is the image timestamp (i.e., 20200509080732). Depending on the configuration chosen for IVI cluster, one additional image is available to facilitate flashing the eMMC used by the Cortex-A72 cluster.

1. First step before flashing the image on the SD card contained the modified booting image is to use a standard Yocto build to flash the eMMC for Cortex-A72 cluster. Download a Yocto image that supports the i.MX 8QuadMax chip, e.g.,

   https://www.nxp.com/webapp/sps/download/license.jsp?colCode=L5.10.9_1.0.0_MX8QM&appType=file1&location=null&DOWNLOAD_ID=null

2. Unzip the downloaded file `L5.10.9_1.0.0_images_MX8QMMEK.zip` inside the created directory, decompress and flash `fsl-image-validation-imx-imx8qmmek.sdcard` to the SD card:

   ```
   $: sudo dd if=fsl-image-validation-imx-imx8qmmek.sdcard of=/dev/path/to/sdcard/device bs=1M
   ```

3. Enlarge the second partition to at least 4 GB using a tool like gparted. It is necessary to copy flashing files. Copy the image generated from the Yocto build for Cockpit solution to the root filesystem partition from the SD Card.

   ```
   $: sudo cp imx-image-<type>- imx8qmmekcockpit-<timestamp>.wic.bz2 /path/to/SDcard/2nd_partition/
   home/root/
   ```

4. Boot the SD Card and flash the eMMC:

   ```
   #: sudo dd if=imx-image-<type>-imx8qmmekcockpit-<timestamp>.wic.bz2 of=/dev/mmcblk0 bs=1M && sync
   #: sudo dd if=imx-image-<type>-imx8qmmekcockpit-<timestamp>.wic.bz2 of=/dev/mmcblk0 bs=1M && sync
   ```

5. Re-flash the SD Card with the image used for the Cortex-A53 cluster:

   ```
   $: bunzip2 -c -df imx-image-<type>-imx8qmmekcockpit-<timestamp>.wic.bz2 | sudo dd of=/dev/
   path/to/sdcard/device bs=1M && sync
   ```

5.2 Flashing the bootloader using uuu

Put the board in serial boot by switching the boot switch. Then, power up the board.

Flash the rootfs image to eMMC using a normal (non-Cockpit) bootloader:

```
$: sudo uuu -v -b emmc_all imx-boot-imx8qmmek-sd.bin-flash imx-image-<type>-imx8qmmekcockpit-
<timestamp>.wic.bz2
```

Flash the rootfs image to SD using a normal (non-Cockpit) bootloader:

```
$: sudo uuu -v -b sd_all imx-boot-imx8qmmek-sd.bin-flash imx-image-<type>-imx8qmmekcockpit-
<timestamp>.wic.bz2
```

5.3 Flashing the non-OP-TEE bootloader using UUU

Put the board in serial boot by switching the boot switch. Then, power up the board. Write Cockpit bootloader on SD using a normal (non-Cockpit) bootloader:

```
$: sudo uuu -v -b sd imx-boot-imx8qmmek-sd.bin-flash imx-boot-imx8qmmek-sd.bin-flash_cockpit
$: sudo uuu -v -b emmc imx-boot-imx8qmmek-sd.bin-flash imx-boot-imx8qmmek-sd.bin-flash_cockpit_spl
```

Once the flashing of the images above is complete, switch back the boot switch to the SD1 configuration. When the board is powered up again, it will boot the cockpit configuration from SD and eMMC.

5.4 Flashing the OP-TEE bootloader using UUU

Programming OP-TEE enabled bootloader using UUU is not supported. You can flash the entire SD/eMMC using UUU instead.

5.5 Flashing entire SD/eMMC using UUU

There is also an option of flashing the entire eMMC or SD at once using UUU. For this, you need the `uuu-imx8qm-cockpit-scripts.zip` file, which contains a script for eMMC and one for SD flashing. These scripts need a normal (non-Cockpit) bootloader and rootfs image (usually a `.wic` file). Their name needs to be `_flash.bin` for bootloader and `_rootfs.sdcard` for the rootfs image, so rename them accordingly before moving on. They also need to be placed in the current working directory, along with the zip file. You do not need to decompress the zip file.

Then, you need to tell UUU what script to use from the zip file. For eMMC, run this one:

```
$: sudo uuu uuu-imx8qm-cockpit-scripts/uuu.cockpit.emmc
```

For SD, run this one:

```
$: sudo uuu uuu-imx8qm-cockpit-scripts/uuu.cockpit.sd
```

This step takes a while to finish, due to the size of the rootfs image.

6 References

- i.MX 8 Family of Applications Processors
- i.MX Software and Development Tools
- Secure Boot on i.MX 8 and i.MX 8X Families using AHAB

7 Revision History

Table 2. Revision history

Revision number	Date	Substantive changes
LF5.10.35_2.0.0	06/2021	Initial release of Linux Cockpit

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Limited warranty and liability — Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

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.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. 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.

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 21 June 2021
Document identifier: IMXDCHPE