

AN15131

How to Set Up RPMSG VKMS on i.MX Chips

Rev. 1.0 — 25 September 2026

Application note

Document information

Information	Content
Keywords	AN15131, RPMSG, VKMS, i.MX chips
Abstract	This document explains implementing RPMSG VKMS on i.MX 8ULP and i.MX 95 platforms.



1 Introduction

Products based on i.MX 8ULP prioritize power efficiency, particularly the wearable devices. Some applications do not require frequent screen updates and benefit from placing CPU cores in Sleep mode to conserve power while maintaining display functionality.

Products such as automotive systems based on i.MX 95 require early-display capability. These systems enable the display at bootloader and achieve seamless display when the bootloader switches to kernel. Finally, these systems switch to display Linux content after the kernel is initialized.

The solution for both cases involves M-core control of the display module:

- Low-power-display case: When the A-core is active, it transmits display information to the M-core, which updates the display. When the A-core enters Sleep mode, the M-core maintains the display and monitors events (such as GPS information) that can wake the A-core.
- Early-display case: The M-core enables the display and shows contents before Linux boot-up is ready, then switches to kernel contents after the A-core transmits the relevant display information.

The VKMS framework in the kernel combined with RPMSG provides the complete solution. The M-core SDK must support updating display and dealing with display RPMSG.

This document describes the implementation methodology.

2 Architecture

VKMS is a software-only model of a KMS driver useful for testing on headless machines. User space applications find a display node with a VKMS driver and simulate the entire process including generating the display frame buffer.

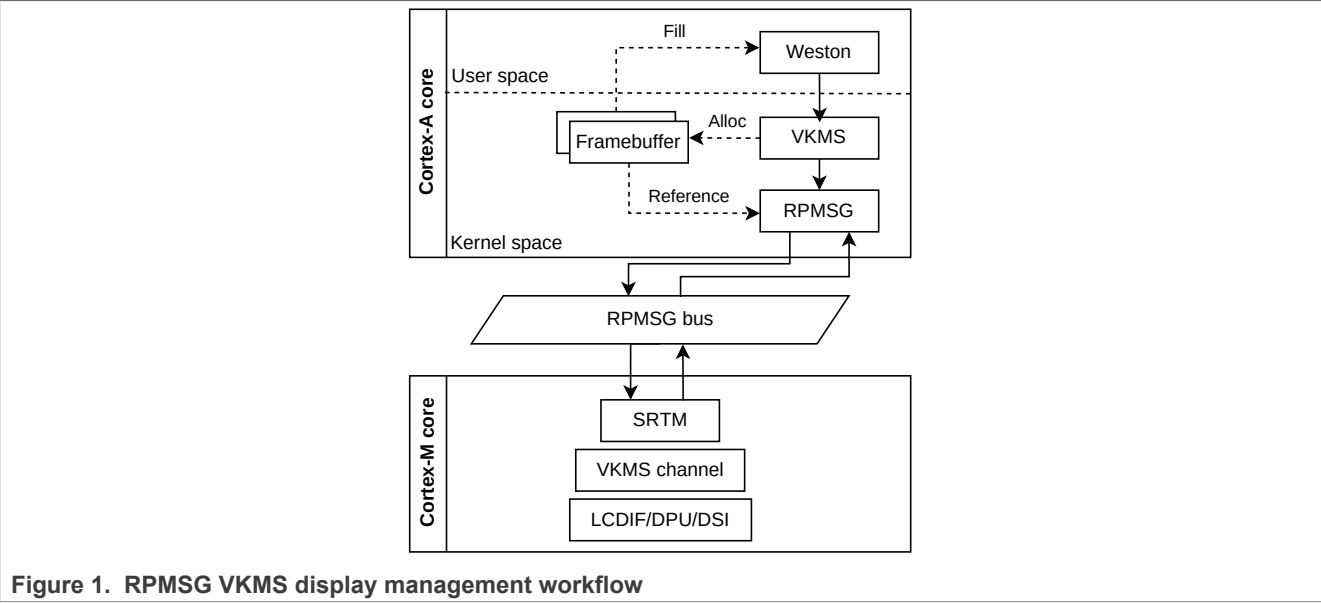
RPMSG is used to facilitate inter-core communication. The RPMSG is a VirtIO-based messaging bus that allows kernel drivers to communicate with remote processors available on the system.

The main modules used in this document are:

- Kernel: VKMS, RPMSG
- MCU SDK: LCDIF/DPU/DSI, SRTM

VKMS interfaces with user space applications to implement frame buffer updates. After VKMS updates the frame buffer, RPMSG transmits the frame buffer to M-core.

On the other hand, M-core owns display and must provide display information to VKMS. The display information is also transmitted through RPMSG.



3 Support for RPMSG VKMS

This section describes the kernel and firmware changes required to enable RPMSG VKMS functionality.

3.1 Driver update in kernel

Constraints that must be considered:

- Implement most function APIs in new files.
 - Convenient for cross-version porting.
- Support CONFIG_DRM_VKMS_IMX configuration option.
 - CONFIG_DRM_VKMS=y alone continues to support the original VKMS.
 - RPMSG VKMS is enabled when both CONFIG_DRM_VKMS=y and CONFIG_DRM_VKMS_IMX=y.

Table 1. VKMS framework

Original VKMS framework	Modification
SHMEM for buffer management	Replace with DMA for contiguous memory allocation
Resolution fixed by XRES_DEF/YRES_DEF. FPS is also fixed	Resolution and FPS determined by M-core
Format offered by vkms_formats[] with multiple selections	Format determined by M-core
The frame buffer is updated in an atomic context protected by a spin lock.	Use workqueue to allow scheduling for RPMSG
Vblank callback generated by local hrtimer (high-resolution timer)	Vblank event transferred from M-core to A-core

There are two ways for updating Vblank:

- A-core hrtimer-based Vblank generation, which is consistent with the original VKMS implementation.
- M-core-based Vblank generation, as described in [Table 1](#).

In the hrtimer-generated Vblank case, only one RPMSG message is issued per frame update. As a result, the update is not guaranteed to complete within the next Vblank cycle after the frame commit. The completion timing depends on workqueue scheduling and RPMSG communication latency.

In the M-core-generated Vblank case, the synchronization between commit and display can be ensured. However, this approach requires two RPMSG messages per single frame update, which increases time cost. Some tests show that hrtimer-generated Vblank case demonstrated better FPS performance than M-core-generated Vblank case. Therefore, both modes are supported. The choice of mode must be made case by case. For more details, refer to [Section 3.3 "Workflow of RPMSG VKMS"](#).

3.2 SRTM service in the M-core

SRTM is a communication service/framework used for inter-processor communication in NXP heterogeneous multicore systems. It supports various use cases, including I/O control and other cross-core services. SRTM also supports VKMS. The mechanisms listed in [Table 2](#) are used to support this implementation.

Table 2. SRTM service in the M-core

SRTM message type	Description	Examples
Request from A-core	A-core sends a message and waits for a reply from M-core	Get resolution, get format, Vblank control
Notify from A-core	A-core sends a message without requiring a reply	Set frame buffer
Notify from M-core	M-core sends a message without requiring a reply	Vblank event

3.3 Workflow of RPMSG VKMS

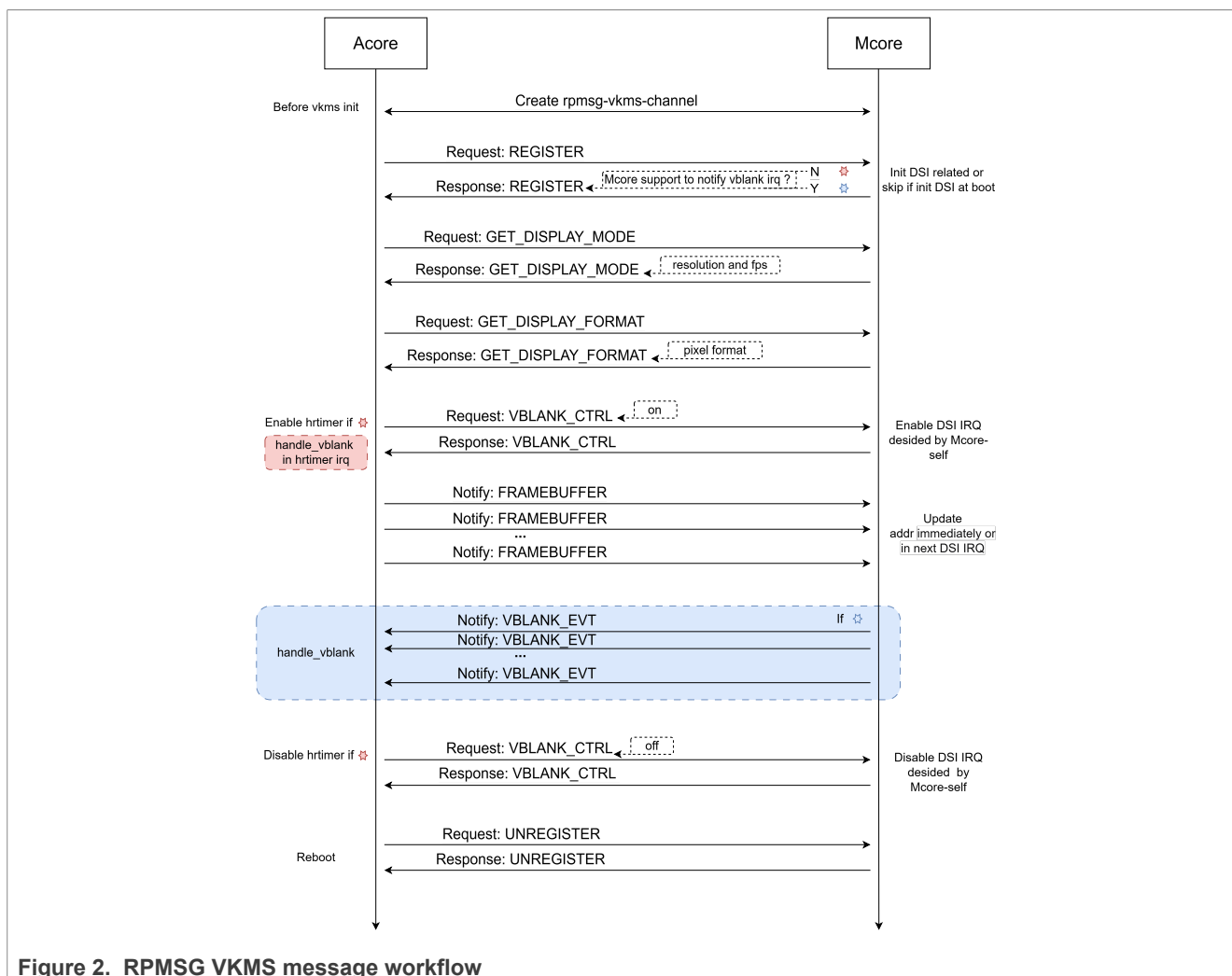


Figure 2. RPMSG VKMS message workflow

The workflow of RPMSG VKMS is as follows:

- Initialization
 - The rpmsg-vkms-channel is created before VKMS initialization.
- Registration Request (from A-core)
 - M-core operates differently based on cases upon receiving the Registration Request:
 - Low-power-display case: Initializes the display.
 - Early display case: No action is required because the display must be initialized before the M-core boots.
 - M-core replies indicating whether it supports Vblank notification. If supported, the M-core handles Vblank interrupts and sends Vblank events back to the A-core.
- Configuration Request (from A-core)
 - M-core replies with the information, such as resolution, FPS, and format.
 - A-core initializes VKMS with this information.
- Enable VBLANK Request (from A-core)
 - A-core always sends this request regardless of M-core Vblank notification support.

- A-core enables hrtimer when M-core does not support Vblank notification and handles Vblank events locally.
- M-core determines Vblank interrupt enablement after receiving this request. The following three cases exist:
 - M-core supports Vblank notification -> enable Vblank interrupt.
 - M-core does not support Vblank notification but updates the frame buffer in Vblank interrupt -> enable Vblank interrupt.
 - M-core does not support Vblank notification and updates the frame buffer immediately upon receiving a message from A-core -> Vblank interrupt not required.
- Frame buffer Notify (from A-core)
 - M-core updates the frame buffer immediately or during the next Vblank interrupt.
- VBLANK_EVT Notify (from M-core)
 - A-core receives Vblank events and processes them when M-core supports Vblank notification.
- Disable VBLANK Request (from A-core)
 - Similar to Enable VBLANK request handling.
- Unregistration Request (from A-core)
 - Required for soft reboot. Without unregistering, registration fails after soft reboot.

4 RPMSG VKMS based on i.MX 8ULP

This section describes the implementation and evaluation of RPMSG VKMS on the i.MX 8ULP platform.

4.1 Setup

The following subsections outline the hardware, software, and deployment requirements for the test environment.

4.1.1 Test environment

Build all binaries on Ubuntu OS.

Table 3. Hardware and software requirements

Category	Description
Hardware	MIMX8ULP-EVK RK055HDMIPI4MA0
Software	• BSP: Linux lf-6.12.20-2.0.0 • MCU: i.MX SDK_2_14_0_EVK-MIMX8ULP • AN15131SW

4.1.2 MCU setup

- Download MCU SDK from [MCUXpresso SDK Builder](#).

- MIMX8UD7xxx10
 - 2.14.1 ([Windows](#), [Linux](#), [macOS](#))
 - 2.14.0 ([Windows](#), [Linux](#), [macOS](#))
 - 2.13.1 ([Windows](#), [Linux](#), [macOS](#))

Figure 3. Downloading MCUXpresso SDK

- Setup toolchain:

```
$ wget https://developer.arm.com/-/media/files/downloads/gnu/12.3.rel1/binrel/arm-gnu-toolchain-12.3.rel1-x86_64-arm-none-eabi.tar.xz
$ sudo tar -xvJf arm-gnu-toolchain-12.3.rel1-x86_64-arm-none-eabi.tar.xz -C /opt
$ export ARMGCC_DIR=/opt/arm-gnu-toolchain-12.3.rel1-x86_64-arm-none-eabi/bin/arm-none-eabi-
```

- Patch and build:

```
# Decompress
$ mkdir -p SDK_2_14_0_MIMX8UD7xxx10_linux
$ tar -zxvf SDK_2_14_0_MIMX8UD7xxx10_linux.tar.gz -C SDK_2_14_0_MIMX8UD7xxx10_linux
# Make copy named low_power_display_vkms
$ cd SDK_2_14_0_MIMX8UD7xxx10_linux
$ mkdir -p ./boards/evkmimx8ulp/demo_apps/low_power_display_vkms
$ cp -r ./boards/evkmimx8ulp/demo_apps/power_mode_switch/. ./boards/evkmimx8ulp/demo_apps/low_power_display_vkms/
# Do patch
$ patch -p1 < AN15131SW/8ULPEVK/SDK_2_14_0_MIMX8UD7xxx10_linux/xxx.patch
# Build
$ cd boards/evkmimx8ulp/demo_apps/low_power_display_vkms/armgcc
$ ./build_debug.sh
```

- MCU output:

```
boards/evkmimx8ulp/demo_apps/low_power_display_vkms/armgcc/debug/sdk20-app.bin
```

4.1.3 Linux setup

- Setup Yocto SDK. For more details, refer to *i.MX Yocto Project User's Guide* (document [UG10164](#)).

```
$ repo init -u https://github.com/nxp-imx/imx-manifest -b imx-linux-walnascar -m imx-6.12.20-2.0.0.xml
$ MACHINE=imx8ulpevk DISTRO=fsl-imx-xwayland source ./imx-setup-release.sh -b bld-xwayland
# firstly build release bootloader and kernel
$ bitbake -c compile -f -v imx-boot
$ bitbake -c compile -f -v linux-imx
```

- Bootloader update:

```
# MCU update
$ cp sdk20-app.bin bld-xwayland/tmp/work/armv8a-mx8ulp-poky-linux/imx-m33-demos/25.03.00/imx8ulp-m33-demo-25.03.00/imx8ulp_m33_TCM_power_mode_switch.bin
$ bitbake -c compile -f -v imx-m33-demos
# uboot update
$ cd bld-xwayland/tmp/work/imx8ulpevk-poky-linux/u-boot-imx/2025.04/git
$ git am -3 AN15131SW/8ULPEVK/u-boot-imx/xxx.patch
$ bitbake -c configure -f -v u-boot-imx && bitbake -c compile -f -v u-boot-imx
# Make bootloader image
$ cd bld-xwayland
$ bitbake -c compile -f -v imx-boot
```

- Get bootloader results:

```
bld-xwayland/tmp/work/imx8ulpevk-poky-linux/imx-boot/1.0/git/imx-boot-  
imx8ulpevk-sd.bin-flash_dualboot  
bld-xwayland/tmp/work/imx8ulpevk-poky-linux/imx-boot/1.0/git/imx-boot-  
imx8ulpevk-sd.bin-flash_dualboot_m33
```

- Kernel update:

```
# patch kernel  
$ cd bld-xwayland/tmp/work/imx8ulpevk-poky-linux/linux-imx/6.12.20+git/git  
$ git am -3 AN15131SW/8ULPEVK/linux-imx/xxx.patch  
# Avoid to fail to insmod release ko after replacing kernel  
$ sed -i '$s/^/#/' scripts/setlocalversion  
$ echo "echo 6.12.20-lts-next-gdfaf2136deb2" >> scripts/setlocalversion  
# Add imx8ulp-evk-vkms.dtb into sources/meta-freescale/conf/machine/imx8ulp-  
lpddr4-evk.conf  
# rebuild kernel  
$ bitbake -c copy_defconfig -f -v linux-imx  
$ bitbake -c configure -f -v linux-imx  
$ bitbake -c compile -f -v linux-imx
```

- Kernel output:

```
bld-xwayland/tmp/work/imx8ulpevk-poky-linux/linux-imx/6.12.20+git/build/arch/  
arm64/boot/Image  
bld-xwayland/tmp/work/imx8ulpevk-poky-linux/linux-imx/6.12.20+git/build/arch/  
arm64/boot/dts/freescale/imx8ulp-evk-vkms.dtb
```

4.1.4 Flash procedure

- Required files:

- imx-boot-imx8ulpevk-sd.bin-flash_singleboot_m33 from release image
- imx-image-full-imx8ulpevk.wic from release image
- bld-xwayland/tmp/work/imx8ulpevk-poky-linux/imx-boot/1.0/git/imx-boot-imx8ulpevk-sd.bin-flash_dualboot
- bld-xwayland/tmp/work/imx8ulpevk-poky-linux/imx-boot/1.0/git/imx-boot-imx8ulpevk-sd.bin-flash_dualboot_m33
- bld-xwayland/tmp/work/imx8ulpevk-poky-linux/linux-imx/6.12.20+git/build/arch/arm64/boot/Image
- bld-xwayland/tmp/work/imx8ulpevk-poky-linux/linux-imx/6.12.20+git/build/arch/arm64/boot/dts/freescale/imx8ulp-evk-vkms.dtb

- Get release image from [L6.12.20-2.0.0_MX8ULP](#) and flash it on MIMX8ULP-EVK.

```
$ sudo uuu -b emmc_all imx-boot-imx8ulpevk-sd.bin-flash_singleboot_m33 imx-  
image-full-imx8ulpevk.wic
```

- Switch EVK to serial download mode (SW5: 01xx_xxxx) and flash bootloader.

```
# uuu_8ulp_dual.auto is in AN15131SW  
$ sudo uuu uuu_8ulp_dual.auto
```

- Switch EVK to A35-emmc/M33-nor mode (SW5: 1000_0010) and boot. Let it stop in U-Boot.

- Connect USB1 to PC.

```
=> ums 0 mmc 0
# Replace Image
# Add imx8ulp-evk-vkms.dtb
# Ctrl + C
=> setenv fdtfile imx8ulp-evk-vkms.dtb
=> saveenv
=> reset
```

- Then, VKMS RPMSG is set up successfully.

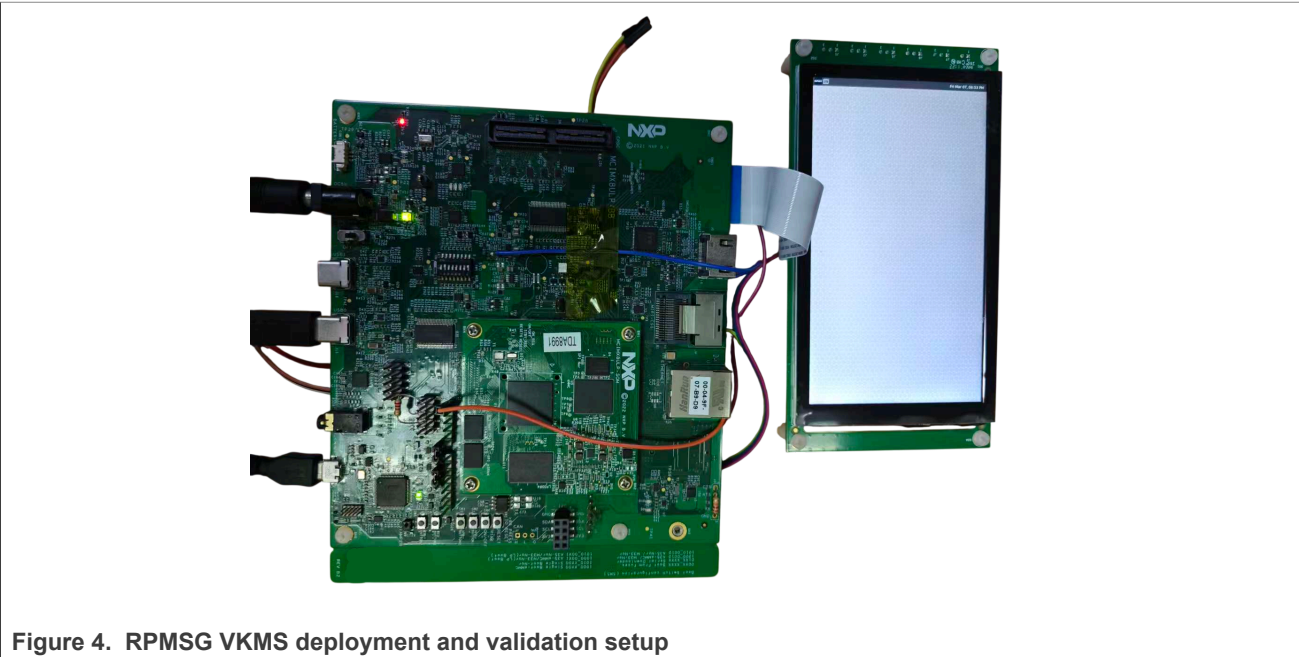


Figure 4. RPMSG VKMS deployment and validation setup

- A-core enters Suspend mode (USB cable disconnected).

```
echo mem > /sys/power/state
```

- M-core wakeup A-core with 'W' in M-core console.

4.1.5 Performance

In this demonstration, the M-core does not support Vblank notification and updates the frame buffer immediately upon receiving messages from the A-core. The system supports 60 FPS.

Table 4. Performance

Case	Panel size	CPU loading	Power consumption
RPMSG VKMS: dual-boot/imx8ulp-evk-vkms.dtb	1280 × 720 at 60 fps	7.5 % (2 CPU as 100 %)	Resume: 400 mW
			Suspend: 329 mW
Release: single-boot/imx8ulp-evk-rk055hdmipi4mv2.dtb	1280 × 720 at 60 fps	7 % (2 CPU as 100 %)	-

The CPU loading is compared with the similar release image.

There is no comparison of power consumption between these two images as M-core closes the power of DDR, DCNANO, and DSI during A-core suspend in release.

4.2 Configuration

This section outlines the configuration parameters and platform settings required for correct RPMSG VKMS operation.

4.2.1 PLL

The RK055HDMIPI4MA0 display panel has a resolution of 1280 × 720 at 60 FPS. The supported format is XRGB8888, 32-bit per pixel. [Table 5](#) shows the current clock rate in the demo.

Table 5. Current clock rate

Module	Clock rate	PLL source
DCNANO	66 MHz	PLL4PFD3DIV2 (396 MHz)/DIV(6)
DSI	79 MHz	PLL4PFD1DIV2 (316 MHz)/DIV(4)

When the M-core supports Vblank notification, the maximum achievable FPS is limited to approximately 40 FPS due to RPMSG bandwidth constraints. In this configuration, the PLL clock can be set lower to reduce power consumption.

```
// SDK_2_14_0_MIMX8UD7xxx10_linux/boards/evkmimx8ulp/demo_apps/  
low_power_display_vkms/lcdif_support.c  
void BOARD_InitLcdifClock(void)  
void BOARD_InitMipiDsiClock(void)
```

The kernel manages PLL4, DCNANO, and DSI clocks during initialization, including configuration and switching operations. Therefore, it is necessary to remove kernel control of these clocks to prevent conflicts with M-core operation.

```
// linux-imx/drivers/clk/imx/clk-imx8ulp.c b/drivers/clk/imx/clk-imx8ulp.c  
int imx8ulp_clk_cgc2_init(struct platform_device *pdev)
```

4.2.2 Domain

- LPAV
 - The release U-Boot checks the Boot mode and configures power domains accordingly:
 - Single-boot: APD assigned to LPAV domain
 - Dual-boot: RTD assigned to LPAV domain
 - LPAV includes DSI and DCNANO modules. For RPMSG VKMS operation, RTD must own the LPAV. Therefore, Dual boot mode is selected.
- DSI and DCNANO
 - The release U-Boot always assigns DSI and DCNANO to the APD domain. This configuration must be modified; otherwise, it affects soft reboot functionality.

```
// u-boot-imx/arch/arm/mach-imx/imx8ulp/soc.c  
void lpav_configure(bool lpav_to_m33)
```

- The release M-core code assigns DSI and DCNANO to the RTD domain and performs the required operations. During A-core suspend/resume cycles, it reassigns DSI and DCNANO back to the APD domain. This logic must be removed to keep DSI and DCNANO permanently in the RTD domain.

```
// SDK_2_14_0_MIMX8UD7xxx10_linux/boards/evkmimx8ulp/demo_apps/  
low_power_display_vkms/app_srtm.c  
void APP_WakeupACore(void)  
void APP_SRTM_SetLPAV(srtm_dispatcher_t dispatcher, void *param1, void  
*param2)
```

5 RPMSG VKMS based on i.MX 95

This section presents the RPMSG VKMS solution and platform-specific configuration for i.MX 95 devices.

5.1 Setup

The setup procedure includes the required development environment, software components, and validation platform.

5.1.1 Test environment

Building all binaries on Ubuntu OS is recommended.

Table 6. Hardware and software requirements

Category	Description
Hardware	i.MX95-19x19-LPDDR5-EVK 1920 × 1080 HDMI Monitor
Software	• BSP: Linux lf-6.18.2-1.0.0 • MCU: mcuxpresso-sdk 26.03.00 • AN15131SW

5.1.2 Software package

This section summarizes the software packages and supporting components required for the RPMSG VKMS implementation.

5.1.2.1 MCU SDK patch

Set up environment for west tool and toolchain:

```
# Install Python virtual environment  
$ python3 -m venv .venv  
# Source the environment  
$ source .venv/bin/activate  
# Install west  
$ pip install -U west  
# Download and configure armgcc toolchain from https://gitlab.arm.com/tooling/  
gnu-toolchains-for-arm/  
# The version of the toolchain is optional and take 14.2.rel1 as example.  
$ export ARMGCC_DIR=${PathOfToolchain}/arm-gnu-toolchain-14.2.rel1-x86_64-arm-  
none-eabi
```

Download and set up the MCUXpresso SDK environment:

```
$ west init -m https://github.com/nxp-mcuxpresso/mcuxsdk-manifests.git
mcuxpresso-sdk --mr v26.03.00
$ cd mcuxpresso-sdk
# Option A: Complete SDK (7-8 GB)
$ west update
# Option B: Board-specific SDK (1-2 GB, recommended)
# Available board names can be found under manifests/boards/
$ west update_board --set board imx95lpd5evk19
$ cd mcuxsdk
```

This demo is based on the MCUXpresso SDK managed by the west. The demo source code must be placed in a separate directory. For example:

```
$ mkdir projects
$ cd projects
# Place rpmsg_lite_vkms_rtos under projects directory
```

5.1.2.2 MPU Yocto meta layer

The meta-imx-vkms layer is based on Yocto Linux L6.18. The build must be performed using this version.

Directory structure:

```
meta-imx-vkms
├── conf
│   └── layer.conf
├── recipes-bsp
│   ├── imx-mkimage
│   │   ├── files
│   │   └── imx-boot_%.bbappend
│   ├── imx-system-manager
│   │   ├── files
│   │   └── imx-system-manager_%.bbappend
│   └── u-boot
│       ├── files
│       └── u-boot-imx_%.bbappend
├── recipes-graphics
│   └── wayland
│       └── weston-init.bbappend
├── recipes-kernel
│   └── linux
│       ├── files
│       └── linux-imx_%.bbappend
```

Key modifications:

- **imx-mkimage:** Integrates the MCU SDK-generated binary.
- **imx-system-manager:** Provides sufficient resource access permissions to the M7 core, enabling display functionality on the MCU side.
- **U-Boot:** Disables selected resources that are assigned to the M7 core.
- **wayland (weston):** Disables G2D usage. As the DPU is handled by the M7 core, the A55 must use the GPU for composition.
- **Linux:** Adds kernel patches to enable VKMS and RPMSG support.

5.1.3 MCU SDK setup

Due to remaining board-specific dependencies, the following modification is required to prevent multiple definition issues caused by `display_support.c`. Enter the examples directory and edit as below:

```
diff --git a/_boards/imx95lpd5evk19/CMakeLists.txt b/_boards/imx95lpd5evk19/CMakeLists.txt
index a100ae6f75..ee498f0406 100644
--- a/_boards/imx95lpd5evk19/CMakeLists.txt
+++ b/_boards/imx95lpd5evk19/CMakeLists.txt
@@ -7,7 +7,7 @@ include(${SdkRootDirPath}/arch/arm/target/ddr.cmake)
   include(${SdkRootDirPath}/arch/arm/target/ext_flash.cmake)
   include(${SdkRootDirPath}/examples/_common/project_setting/arm_common.cmake)
   include(${SdkRootDirPath}/${board_root}/imx95lpd5evk19/${core_id}/
setting.cmake)
-include(${SdkRootDirPath}/examples/_common/project_segments/display_support/
prjseg.cmake)
+include(${SdkRootDirPath}/${board_root}/${board}/project_segments/
display_support/prjseg.cmake OPTIONAL)
```

Configure the build environment and compile the project:

```
# Enter projects directory and build the demo.
$ west build -p always rpmsg_lite_vkms_rtos/ --toolchain armgcc --config debug -
b imx95lpd5evk19 -Dcore_id=cm7
```

After a successful build, the output binary is available at `build/rpmsg_lite_vkms_rtos_remote_cm7.bin`. This output binary is used in the Yocto build and is replaced only when the M7 firmware is updated.

5.1.4 Linux setup

Initialize and build the Yocto environment as follows:

```
# Initialize repo
repo init -u https://github.com/nxp-imx/imx-manifest.git -b imx-linux-whinlatter
-m imx-6.18.2-1.0.0.xml
# Sync sources
repo sync
# Add meta layer
# Place meta-imx-vkms under sources directory
# Setup build environment
MACHINE=imx95-19x19-lpddr5-evk DISTRO=fsl-imx-xwayland source ./imx-setup-
release.sh -b build
```

Modify `build/conf/bblayers.conf` to include the new layer:

```
BBLAYERS += "${BSPDIR}/sources/meta-imx-vkms"
```

Start the build:

```
# core-image-minimal and multimedia are optional
$ bitbake imx-image-full
```

5.1.5 Flash procedure

After a successful build, the image file is generated at:

```
<Yocto_Build_Dir>/build/tmp/deploy/images/imx95-19x19-lpddr5-evk/imx-image-full-  
imx95-19x19-lpddr5-evk.rootfs.wic.zst
```

Flash the image to an SD card or eMMC using dd, fastboot, or flash tools.

5.1.6 Expected display behavior

In this demo, the LVDS1 interface is connected to an LVDS-to-HDMI bridge, which drives an external monitor.

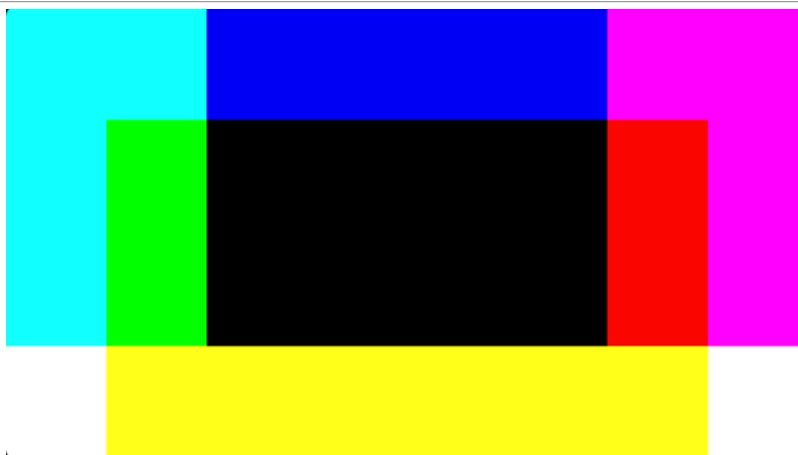
After the board is powered on with the LCD display connected, an initial overlapped color pattern (test frames) appears on the screen. This color pattern indicates that the M7-side display pipeline is active and functioning.

Once Linux finishes booting and the VKMS driver is loaded, the Weston frame buffers are sent to the M7 core. The screen content switches from the initial test pattern to the Weston desktop environment.

This behavior confirms that:

- The RPMSG communication between the A55 (Linux) and M7 core is functioning correctly.
- The VKMS driver is successfully initialized on the Linux side.
- The display frames are changed from the M7 domain to the Linux graphics.

Note: The M7 still hosts the display modules.



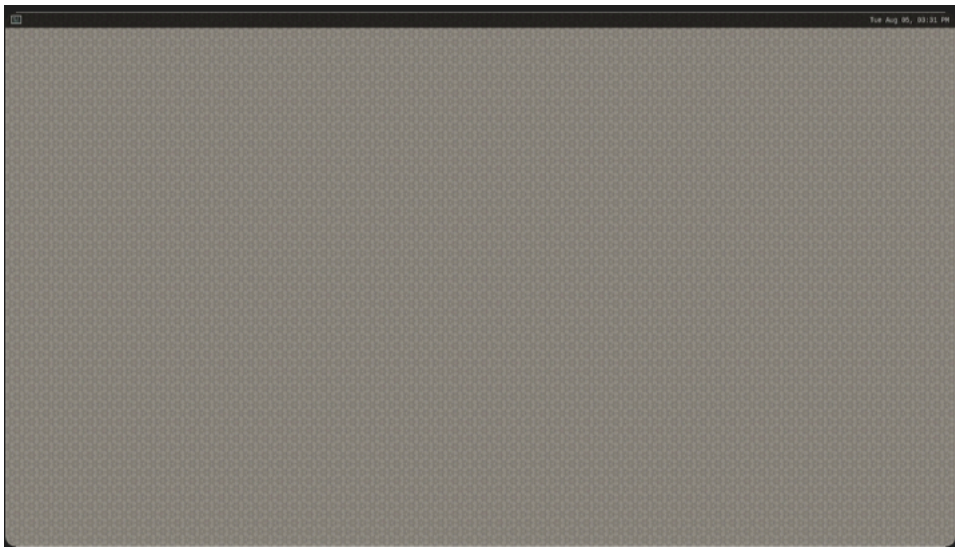


Figure 5. Display behavior

6 Acronyms

[Table 7](#) lists the acronyms used in this document.

Table 7. Acronyms

Term	Description
VKMS	Virtual kernel mode setting
RPMSG	Remote processor messaging
A-core	Cortex-A cores
M-core	Cortex-M cores
SRTM	Simplified real-time messaging
Vblank	The vertical blanking interval after the last line of one frame has been scanned out and before the first line of the next frame begins scan out.

7 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 2026 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.

8 Revision history

[Table 8](#) summarizes the revisions to this document.

Table 8. Revision history

Document ID	Release date	Description
AN15131 v.1.0	25 September 2026	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, μ Vision, 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.

Contents

1	Introduction	2
2	Architecture	2
3	Support for RPMSG VKMS	3
3.1	Driver update in kernel	3
3.2	SRTM service in the M-core	4
3.3	Workflow of RPMSG VKMS	5
4	RPMSG VKMS based on i.MX 8ULP	6
4.1	Setup	6
4.1.1	Test environment	6
4.1.2	MCU setup	6
4.1.3	Linux setup	7
4.1.4	Flash procedure	8
4.1.5	Performance	9
4.2	Configuration	10
4.2.1	PLL	10
4.2.2	Domain	10
5	RPMSG VKMS based on i.MX 95	11
5.1	Setup	11
5.1.1	Test environment	11
5.1.2	Software package	11
5.1.2.1	MCU SDK patch	11
5.1.2.2	MPU Yocto meta layer	12
5.1.3	MCU SDK setup	13
5.1.4	Linux setup	13
5.1.5	Flash procedure	14
5.1.6	Expected display behavior	14
6	Acronyms	15
7	Note about the source code in the document	15
8	Revision history	17
	Legal information	18

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.