Renesas RA Family

Building a Vision AI Application using the RA8P1 MCU with Ethos-U55 NPU

Introduction

This application project is designed to provide guidelines for implementing Vision AI applications using Renesas RA8P1 MCUs with Ethos-U NPU support.
Key guidelines and demonstrations covered in this application project include:

- Compiling a pre-trained TensorFlow Lite model to enable inference execution on the Ethos-U NPU by the RUHMI Framework.
- Compiling a pre-trained TensorFlow Lite model to enable inference execution on the CPU by the RUHMI Framework.
- Integrating the RUHMI compilation results in an embedded project to perform runtime inference using real-time camera input from the OV5640 camera sensor via the MIPI CSI interface.
- Displaying the inference result and real-time camera feed on the parallel graphics LCD included with the EK-RA8P1 kit.
- Implementing a dual-core solution with low-power mode features

The contents of this application project can be used as a reference for developing other vision AI applications.

Required Resources

Target Devices

- RA8P1

Software and development tools

- e² studio IDE v2025-12 with LLVM 21.1.1 support
- Renesas Flexible Software Package (FSP) v6.4.0
- Renesas RUHMI (Robust Unified Heterogeneous Model Integration) Framework (installed as an add-on for e² studio).
- Tera Term console or a similar application.

The links to download the FSP and e² studio IDE are available at https://github.com/renesas/fsp.

Hardware

- EK-RA8P1 with included Camera Module and Parallel graphics LCD panel
  - Evaluation Kit for RA8P1 MCU Group (renesas.com/ra/ek-ra8p1)
- Workstation running Windows® 10 or Windows® 11.
- One USB Type-C device cable

Prerequisites and Intended Audience

This application project assumes that the audience understands AI/ML design concepts and has good experience using the Renesas e² studio IDE and the Renesas RA Smart Configurator. Basic instructions on how to import and compile an application are not provided. Refer to the "Importing an Existing Project into e² studio" section in the Flexible Software Package (FSP) User's Manual (UM) for guidance.

The intended audience is all users who are or will be developing AI applications using Renesas RA MCUs, specifically those with Ethos-U NPU support and pre-quantized int8 models. Model training and quantization are outside the scope of this application project.

Contents

1. RA8P1 MCUs with Ethos-U NPU

This section briefly introduces the RA8P1 MCU features and how these features contribute to improved inference speed and enhanced user visual experience. Other RA8 MCUs may have a selection of these features. Additionally, brief information regarding how the e² studio IDE and FSP support these features is provided. In this example application, the e² studio IDE and the LLVM Embedded Toolchain for Arm are used. The tools reference information, and screenshots are presented based on the e² studio IDE and the LLVM Embedded Toolchain for Arm.

1.1 Dual Core

The RA8P1 is based on the Arm® Cortex-M85 Processor and Arm® Cortex®-M33 Processor. The following table captures some of the core features available for AI applications. Please refer to the Hardware User's Manual for any additional information.

Table 1 RA8P1 Dual Core Features

Feature	Cortex-M85 Features	Cortex-M33 Features	Comments
Architecture	ARMv8.1-M architecture profile	ARMv8-M architecture profile	Refer to R01AN7881 for developing a dual-core application.
Instruction Clock	Running up to 1 GHz	Running up to 250 MHz	
Rational/Float Representation	Floating Point Unit (FPU) compliant with the ANSI/IEEE Std 754-2008. Scalar half, single, and double-precision floating-point operation. Interrupt notification when an FPU exception occurs.	Floating Point Unit (FPU) compliant with the ANSI/IEEE Std 754-2008. Single-precision floating-point operation. Interrupt notification when an FPU exception occurs.	Improves AI application performance when using a float model. Needs more model storage space and runs slower than a quantized model. See **Note 1*** for how to enable the FPU.
Processing Extensions	M-profile Vector Extension (MVE) Integer, half-precision, and single-precision floating-point MVE (MVE-F)	Armv8-DSP Extension	Refer to R11AN0756 to understand how to use the Helium technology and CMSIS with quantized AI models.
Caches	Cache – Instruction cache: 16 KB with ECC – Data cache: 16 KB with ECC	Cache – Code-bus Cache (C-Cache): 16 KB with ECC – System-bus Cache (S-Cache): 16 KB with ECC	Refer to the FSP UM section titled "Cortex-M85 Caches" to understand the general guidelines. Refer to R11AN0538 to understand the Cortex-M33 usage of the C-Cache and S-Cache
Tightly Coupled Memory (TCM)	Tightly Coupled Memory (TCM) – ITCM: 128 KB (16 blocks × 8 KB) with ECC – DTCM: 128 KB (16 blocks × 8 KB) with ECC	Tightly Coupled Memory (TCM) – CTCM: 64 KB with ECC – STCM: 64 KB with ECC	See **Note 2***
Low power modes	Low power modes – Sleep mode – Deep Sleep mode	Low power modes – Sleep mode – Deep Sleep mode	Refer to the Hardware User's Manual on the enter and exit of the CPU low power mode. In this application, the deep sleep mode is

			used for power saving on CM85 when the image classification is not actively running.
Interrupts Available	96 Interrupts	96 Interrupts	The interrupts are used to progress system states and trigger subsequent data sensing and data processing.

Note 1*: For both cores, navigate to the project properties: **Properties -> C/C++ Build->Settings->Tools Settings->CPU->** and set **Float ABI** to **FP Instruction (hard)**.

Note 2*: The RA8P1 Cortex-M85 integrates 256 KB of TCM RAM with 128 KB of Instruction TCM with ECC and 128 KB of Data TCM with ECC. TCM RAM, being part of the CPU subsystem, provides a consistent response time with optimized access. The TCM memory areas are not cached. By default, the FSP BSP has both ICTM and DTCM enabled with regions defined in the linker script. Application code can allocate selected code and data to these regions.

When designing an AI application, the ITCM can be used to store real-time interrupt routines or interrupt callback routines. For models or sensor data that fit the DTCM, placing them in DTCM can be used to improve access speed and determinism. As the camera image buffers are larger than the DTCM, the DTCM is not used in this example.

1.1.1 Inter-Processor Communication (IPC)

Inter-processor communication (IPC) supports hardware sharing and communication between two processors. There are three IPC mechanisms between the CM85 and CM33 cores.

- IPC can generate Maskable and Non-maskable interrupt events to support communication between the processors. In this application project, the Non-maskable Interrupt issued by CM33 is used to wake the CM85 core from Deep Sleep mode.
- IPC semaphore mechanisms can support applications to achieve mutually exclusive access between processors.
- IPC data communication message FIFOs can be used for simple data communication between the two cores.

Refer to application project R01AN7881 for more examples of IPC usage in a dual-core system.

1.1.2 Shared Memory

Shared Memory allocation for the dual-core system needs to be carried out manually. Please refer to the application note R01AN7881 for the use cases and examples of implementation of a Shared Memory region.

1.2 Memory Sub-Systems

There is an abundance of memory available on the EK-RA8P1 to support the various needs of an AI application.

Table 2. Memory Sub-Systems

Types	Total Size	Typical Usage in an AI Application	Usage in this Application Project
User SRAM	• 1664 KB if ECC is enabled • 1872 KB if ECC	Input data for the AI model, Tensor Arena, Sensor Data	Tensor Arena for the MobileNet V1 projects, image buffer as input to AI inference

	is disabled		
Code MRAM See **Note 1***	1 MB	AI model, Static Image, application code	Text Image, application code, Used for GLCDC framebuffer for the EfficientNet Lite0 projects
External SDRAM See **Note 2***	64 MB organized as 16M x 32 bits	Sensor Data, GLCDC framebuffer, AI model, Tensor Arena	VIN output buffer, Graphics background, and foreground framebuffer, Used for the EfficientNet Lite0 projects for runtime AI model storage and Tensor Arena usage
External OSPI See **Note 2***	64 MB	AI model, Static Image	Used for the EfficientNet Lite0 projects for AI model storage

Note 1*: Renesas RA8P1 features 1 MB of Code MRAM with fast read/write access, low power consumption, and high endurance. There is also a 1 MB Prefetch buffer, which accelerates CPU instruction fetches from the code MRAM.

Note 2*: The RA8P1 has external address space, which is divided into 8 CS areas, an SDRAM area (SDCS), and OSPI areas (CS0 and CS1). On the EK-RA8P1, an SDRAM with a 32-bit data bus is available for good performance throughput. Additionally, an OSPI flash of 64MB is provided on board, which can facilitate AI application development.

Refer to Application Note R01AN7880 for details on the RA8P1 Memory Topologies.

1.3 Neural Processing Unit

The RA8P1 provides the Arm® EthosTM-U55 NPU. This NPU offers improvement of AI inference performance and contributes significantly to the image classification performance we observed with this implementation.

For the Ethos-U55 hardware features, tools, and FSP support for using Ethos-U55, please refer to Application Note R01AN7712.

The FSP UM includes a section describing the APIs; please review it to understand the API details.

1.4 Human-Machine Interface Features

There is a group of Human Machine Interface (HMI) peripherals that support data sensing and rendering of the application results. For details on the specification of these peripherals, please refer to the Hardware User's Manual of the RA8P1. For details on how to use these modules with FSP support, please refer to the FSP User's Manual. This section highlights the key features and how they are used in this image classification application.

1.4.1 2D Drawing Engine (DRW)

The RA8P1 MCU incorporates a 2D drawing engine (DRW) capable of performing vector drawing and image rasterization, the BitBLT function (fill, copy, stretch, etc.), and supports various types of color formats. FSP supports this hardware feature through the r_drw stack.

In this application, DRW is used to stream the camera image, display the AI model information, and print the inference result on the parallel graphics LCD. FSP driver r_drw is used to access the DRW engine in this application.

1.4.2 Graphic LCD Controller (GLCDC)

The GLCDC on the RA8P1 MCU supports a single-color background and two graphics planes. A variety of pixel formats are supported. The GLCDC peripheral works with several sub-peripherals to form a pixel

data processing pipeline. The GLCDC controller can interface with a parallel graphics LCD or a MIPI DSI interface.

In this application project, the parallel graphics LCD interface is used. The input for the GLCDC is RGB565, and the output is RGB888. The parallel graphics LCD includes a touch controller. The touch IRQ is used in this application project by the Cortex-M33 to wake up the CM85 from deep sleep mode to resume image classification functionality.

1.4.3 MIPI CSI and Video Input Module (VIN) Interface

The MIPI CSI-2 is a MIPI Camera Serial Interface 2 receiver module. This module supports MIPI CSI-2 V3.0 and MIPI D-PHY V2.5 (80 Mbps to 720 Mbps/lane and up to 2 lanes). The image signal received by the MIPI-CSI2 is transferred to the video input module (VIN) inside the MCU.

The VIN is a video capture module that stores YCbCr-422 data and RGB data in external memory through the MIPI CSI-2 interface. The module has one video channel that can control the capture of data into a capture area of up to 4096 × 4096 pixels when scaling is off (2048 × 2048 pixels when scaling is on). It can also provide vertical and horizontal scaling of the data up to three and two times, respectively.

This application project uses the MIPI-CSI and the VIN module to stream the images from the camera.

Note that the EK-RA8P1 hardware configuration allows usage of either MIPI-DSI or MIPI-CSI, but not both at the same time.

1.4.4 MIPI DSI

The MIPI PHY on the RA8P1 is MIPI D-PHY that supports the MIPI Alliance Specification Version 2.1 for the D-PHY Specification. This D-PHY supports two lanes with a maximum rate of 720 Mbps per lane. The MIPI DSI on the RA8D1 is the MIPI DSI-2 Host model, which supports the MIPI Alliance Specification for Display Serial Interface 2 (DSI-2) Specification.

The MIPI DSI interface accesses the MCU bus system through the GLCDC interface, so when MIPI DSI is used, the GLCDC module also needs to be incorporated into the system. When a parallel graphics LCD interface is used, the GLCDC can be used without adding the MIPI DSI stack.

This module is not used in this example project. Refer to R11AN0756 for an example usage of the CEU with MIPI-DSI for camera image streaming. The MIPI LCD panel in EK-RA8D1 can be used with the EK-RA8P1.

1.5 Communication Interfaces

1.5.1 Serial Communication Interface (SCI)

SCI comprises 6 channels for asynchronous and synchronous serial interfaces that can be configured for many standard serial communication interfaces, for example, UART, Smart Card Interface, and simple SPI, etc. In AI applications, these interfaces can be used for data collection as well as external module configuration and control.

In this image classification application, Channel 8 is configured as a UART to communicate with the J-Link OB Virtual COM port for status display on the Windows terminal (Tera Term).

1.5.2 I2C Bus Interface (IIC)

The I2C bus interface (IIC) comprises two channels. The IIC module conforms to and offers a subset of the NXP I2C (Inter-Integrated Circuit) bus interface functions. This module can be utilized for sensor communication and data collection in an AI application.

In this image classification application, channel 1 is for the following two functionalities:

- In master mode, communicate with the I2C slave interface on the OV5640 control block.
- In master mode, configure the LCD touch controller.

1.6 Timers

1.6.1 General PWM Timer (GPT)

The General PWM Timer (GPT) is a 32-bit timer with GPT32 × 14 channels on RA8P1. The clock sources can be independently selected for each channel. Each channel has two input/output pins, which can be used for input capture and output comparison. Generally, this peripheral can be used in an AI application to provide sensor clocks or time tracking.

In this image classification application, two channels are used:

- Channel 0 is used as a periodic timer to track the various execution times of the system.
- Channel 12 is used to generate the input clock for the OV5640 using the output pin P501.

1.6.2 Lower Power Asynchronous General Purpose Timer (AGT)

The Low Power Asynchronous General Purpose Timer (AGT) is a 16-bit timer with 2 channels that can be used for pulse output, external pulse width or period measurement, and counting external events. This timer consists of a reload register and a down counter. In this application project, the AGT channel 1 is used by the Cortex-M33 core as a periodic timer to define the operational period of the Cortex-M85. When the operating period expires, Cortex-M33 sends an IPC message-FIFO to request the Cortex-M85 to enter Deep Sleep Mode.

2. Deploy an Open-Source AI Model using the Renesas RUHMI Framework

Renesas RUHMI Framework is an AI GUI Tool integrating an embedded AI compiler with e^2 studio. It is powered by EdgeCortix® MERA™ 2.0. This AI tool can compile highly optimized models in minutes to run efficiently on Renesas embedded processors. For more details on the usage of RUHMI, please refer to the RUHMI Framework Quick Start Guide R11QS0065.

RUHMI Framework offers the following major functionalities:

- Translates and optimizes the AI model to perform inference using Ethos-U or RA MCU CPU.
- Supports multiple ML frameworks like Tensor Flow, PyTorch, and ONNX.
- Supports quantization of float models.

3. The Image Classification AI Application

This section describes how the application project architects the operation of the various hardware units, data processing, communication among them, the AI inference, and the graphics operations. The following are some of the key design considerations that are implemented in this application:

- Decouple the AI inference operation path and display operation path.
- Utilize the Dual Core features to optimize the power performance.
- Utilize SRAM access performance advantage over SDRAM for key operations in the system
- Configure the Graphics system for optimal camera data capture and camera image streaming effect.

3.1 System Architecture

This section describes two system architectures used for image classification on the RA8 platform: MobileNet V1 Model and EfficientNet Lite 0 Model.

3.1.1 Block Diagram using the MobileNet V1 Model

Figure 1 shows the system architecture for the MobileNet V1 model on the RA8 dual-core platform.

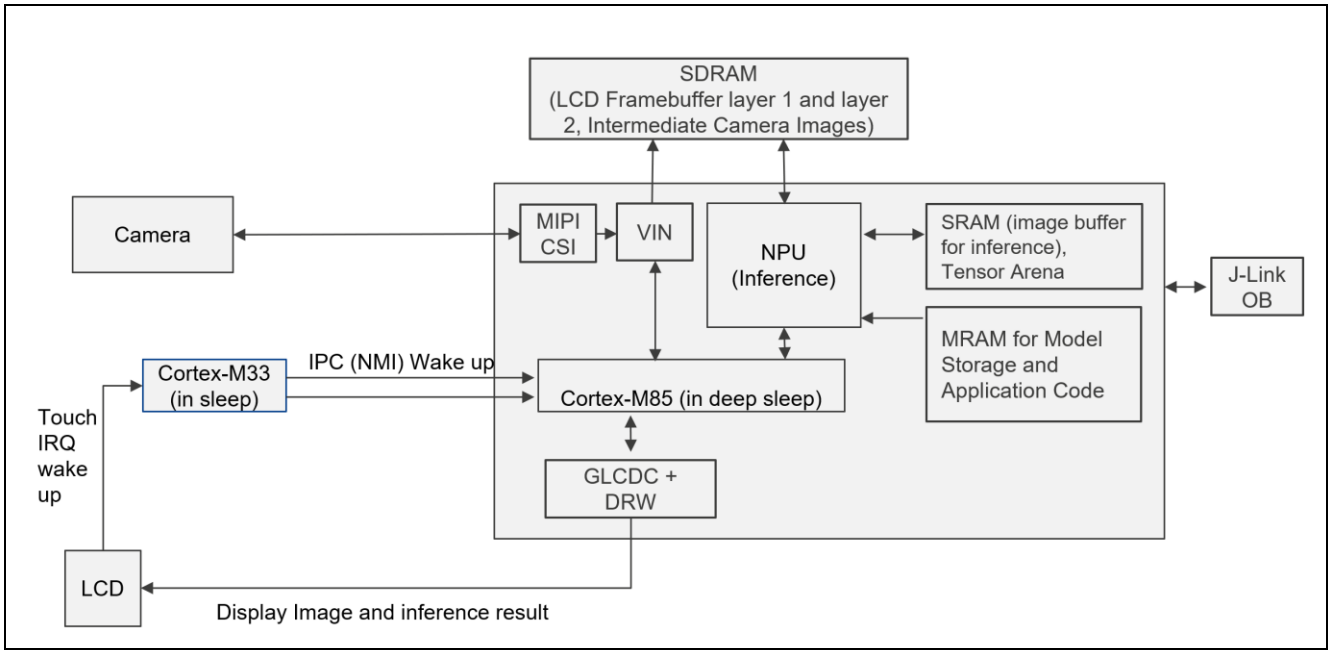

Figure 1. System Architecture for mobilenet v1 model

3.1.2 Block Diagram using the EfficientNet Lite0 Model

Figure 2 shows the system architecture for the EfficientNet Lite0 model model on the RA8 dual-core platform.

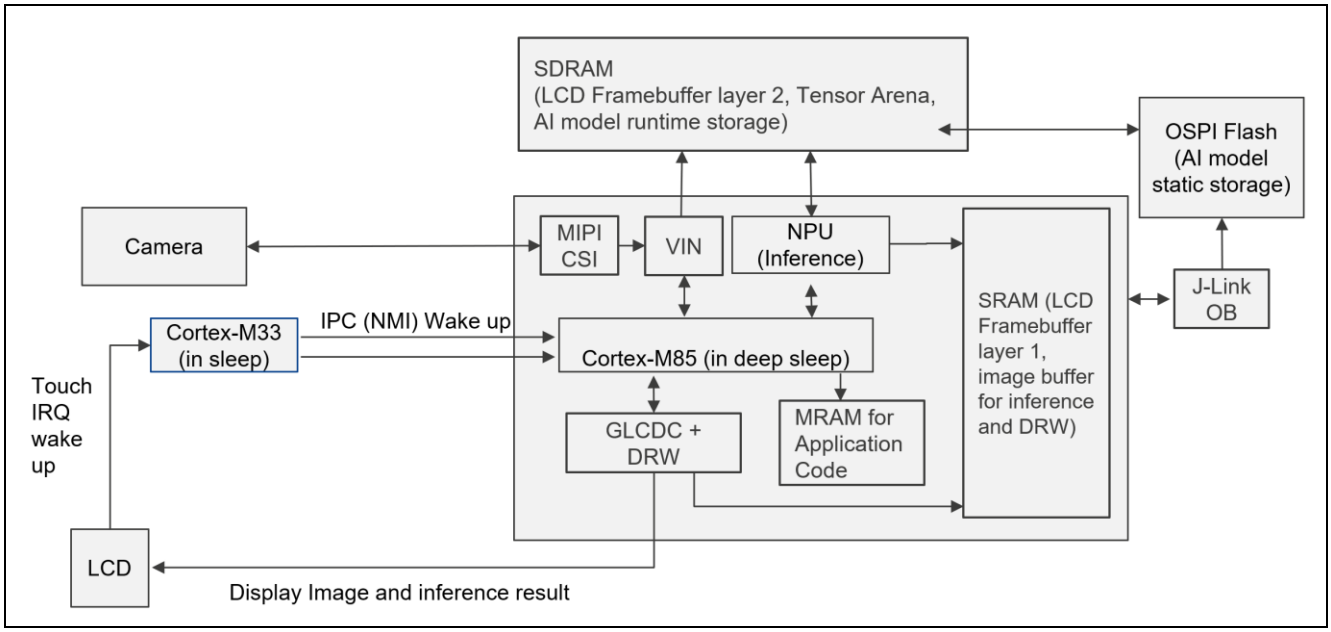

Figure 2. System Architecture for EfficientNet Lite 0 Model

For both architectures, the Cortex-M85 performs the following major functions:

1. Initialize the hardware components and FSP drivers: SDRAM, GLCDC, DRW, MIPI-CSI, VIN, Ethos-U55 NPU, OV5640, IPC, and LPM.
2. Stream the camera image data in 640x480 RGB565 format and display the image on the parallel graphics LCD.

3. Preprocess the camera image to 224x224 RGB888 format and feed it to the AI inference engine for runtime image classification.
4. Use the Ethos-U or the CPU to perform the runtime inference operation.
5. Enter Deep Sleep Mode after receiving the IPC Message (‘1’) from Cortex-M33
6. Exit the Deep Sleep Mode after receiving IPC NMI from Cortex-M33 and resume normal camera image streaming and AI inference.

The Cortex-M33 performs the following major functions:

1. Initialize the hardware components: AGT, IPC, LCD touch, and touch IRQ.
2. Monitor the AGT timer overflow event.
3. Send an IPC message to trigger the Cortex-M85 to enter Deep Sleep Mode when an AGT timer expires.
4. When the user touches the LCD, send an IPC NMI to wake up the Cortex-M85

For the EfficientNet Lite0 model, the Corex-M85 also copies the AI model to the SDRAM during the system initialization stage.

3.2 FSP Driver Usage

The following FSP v6.4.0 drivers are used in the software development of the application.

Selected software components

FreeRTOS - Memory Management - Heap 4	v11.1.0+fsp.6.4.0
FreeRTOS	v11.1.0+fsp.6.4.0
ARM Ethos-U Profiler	v25.2.0+fsp.6.4.0
Arm CMSIS Version 6 - Core (M)	v6.1.0+fsp.6.4.0
Arm DSP Library Source	v1.16.2+fsp.6.4.0
Arm NN Library Source	v7.0.0+fsp.6.4.0
Arm CMSIS View Library Source	v1.2.0+renesas.0.fsp.6.4.0
Ethos-U core driver	v25.2.0+renesas.0.fsp.6.4.0
Flatbuffers	v23.5.26+renesas.0.fsp.6.4.0
TFLM Core Lib	v25.2.0+renesas.0.fsp.6.4.0
TFLM cmsis-nn kernel	v25.2.0+renesas.0.fsp.6.4.0
TFLM Ethos-U kernel	v25.2.0+renesas.0.fsp.6.4.0
RA8P1-EK Board Support Files	v6.4.0
Board support package for RA8P1	v6.4.0
Board support package for R7KA8P1KFLCAC	v6.4.0
Board support package for RA8P1 - Events	v6.4.0
Board support package for RA8P1 - FSP Data	v6.4.0
Board support package for RA8P1 - Linker	v6.4.0
Board Support Package Common Files	v6.4.0
Direct Memory Access Controller	v6.4.0
TES D/AVE 2D Port	v6.4.0
Graphics LCD Controller	v6.4.0
General PWM Timer	v6.4.0
I2C Master Interface	v6.4.0
I/O Port	v6.4.0
MIPI CSI	v6.4.0
MIPI PHY Host	v6.4.0
Octa Serial Peripheral Interface Flash	v6.4.0
SCI UART	v6.4.0
Video Input (VIN)	v6.4.0
Ethos-U Driver Wrapper	v6.4.0
FreeRTOS Port	v6.4.0
TES DAVE 2D Drawing Engine	v3.8.0+fsp.6.4.0

Figure 3. FSP Drivers and Third-Party Libraries Used

3.3 Dual Core Architecture and Low Power Feature

The following timeline graph describes the dual core communication states.

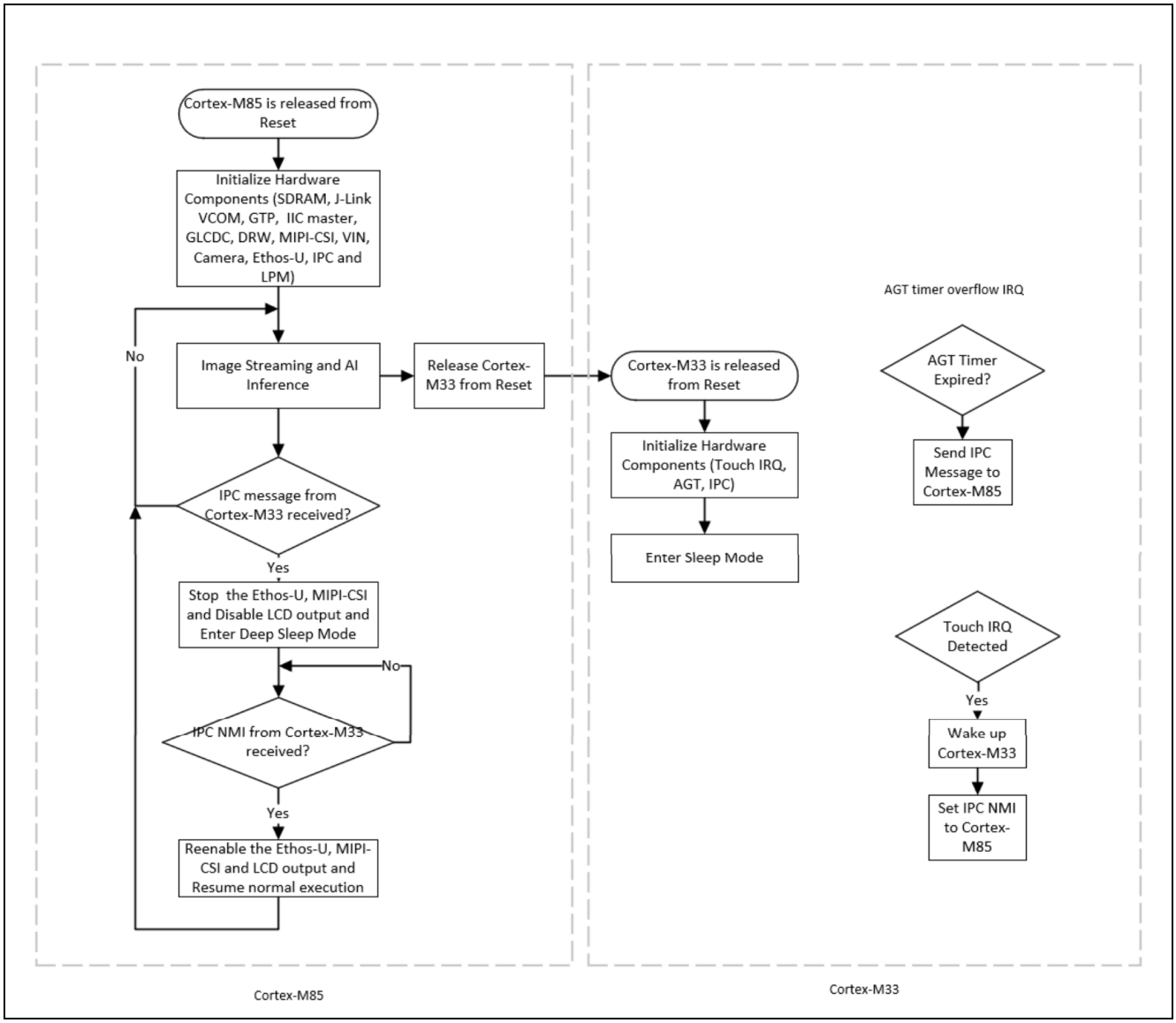

Figure 4. Dual Core Communication and Low Power Feature

3.4 Memory Allocation

The following tables detail the memory usage of the system when using FSP v6.4.0. The BSP Stack, Heap, Thread Stack, and Thread Heap are allocated in the e^2 studio Stack view, while other usages are allocated by the compiler based on the application code.

3.4.1 Memory Allocation using MobileNet V1

The RA8P1 on the EK-RA8P1 features a 1MB code MRAM. For both Ethos-U inference and CPU inference, the AI model and application code are stored in the code MRAM. The application code for CPU inference is larger than that for Ethos-U inference, as more CMSIS-NN kernels are used, which consume more memory than Ethos-U inference.

Table 3. Code MRAM Usage (MobileNet V1 0.25)

Category	Ethos-U Inference (KB)	CPU Inference (KB)
AI Model	510	460
Application code (including FSP package and user application)	220	274

The following table outlines major SRAM usage. For models with a large Tensor Arena, it may be allocated in SDRAM. However, inference performance is typically lower than when the Tensor Arena resides in SRAM. If the application is updated with additional functionality, the stack and heap usage need to be adjusted.

Table 4. SRAM Memory Usage (MobileNet V1 0.25)

Category	Description		Ethos-U Inference (KB)	CPU Inference (KB)
Stack and Heap	BSP Main Stack		4	4
	BSP Heap		12	12
	Display Thread Stack		8	5
	Camera Thread Stack		6	7
	AI Inference Thread Stack		8	18
	Thread Heap		16	14
AI-specific usage	Tensor Arena		402	607
	Model input image	model_buffer_int8[ ]	150	150
	Copy of model input image	model_input[]	Allocated from the tensor Arena area	150
Application Code			11	11

The GLCDC framebuffer fb_foreground and the VIN output buffers are stored in SDRAM. The VIN buffers are used as the first layer of the LCD display.

Table 5. SDRAM Memory Usage (MobileNet V1 0.25)

Category	Description	Ethos-U Inference/CPU Inference	
SDRAM	Camera Output/VIN Output	vin_image_buffer_1, vin_image_buffer_2, vin_image_buffer_3	1800 KB
	GLCDC framebuffer (fb_foreground)	350 KB	

3.4.2 Memory Allocation using EfficientNet Lite0

The RA8P1 on the EK-RA8P1 features a 1MB code MRAM. For both Ethos-U inference and CPU inference, the AI model is loaded to OSPI at compile time and then copied to SDRAM at runtime at the beginning of the

application code execution. The application code for CPU inference is larger than that for Ethos-U inference, as more CMSIS-NN kernels are used, which consume more memory than Ethos-U inference.

Table 6. Code MRAM Usage (EfficientNet Lite0)

Category	Ethos-U Inference (KB)	CPU Inference (KB)
Application code (including FSP package and user application)	220	407

The following table outlines the primary uses of SRAM. In this design, the AI model and Tensor Arena are placed in SDRAM. To preserve display performance, the camera image buffer and GLCDC frame buffers are allocated in SRAM to prevent contention on the SDRAM bus. If the application is expanded with additional features, the stack and heap sizes may need to be adjusted.

Table 7. SRAM Memory Usage (EfficientNet Lite0)

Category	Description		Ethos-U Inference (KB)	CPU Inference (KB)
Stack and Heap	BSP Main Stack		4	4
	BSP Heap		12	12
	Display Thread Stack		8	8
	Camera Thread Stack		6	6
	AI Inference Thread Stack		8	8
	Thread Heap		16	14
Display Related	Camera output image		450	
	GLCDC framebuffer (fb_background)		600	
	Model input image	model_buffer_int8[ ]	150	
	Copy of model input image	model_input[]	Allocated from the tensor Arena area	150
Application Code			11	11

This model requires a large Tensor Arena that does not fit in SRAM, so it is allocated in SDRAM. The AI model is larger than MRAM capacity, so it is stored in OSPI flash before application execution. When the system comes out of reset, the FSP initialization routine copies the model from OSPI to SDRAM before the user application starts. The GLCDC foreground framebuffer (Layer 2) is also allocated in SDRAM.

Table 8. SDRAM Memory Usage (EfficientNet Lite0)

Description	Ethos-U Inference (KB)	CPU Inference (KB)
AI Model	4570	4586
Tensor Arena	1470	1678
GLCDC framebuffer (fb_foreground)	350	

3.5 Hardware Configurations

3.5.1 MIPI-CSI and Camera Connections

Table 9 is the MIPI-CSI and Camera Pin connection used in the Application Project. This connection is established by the ribbon cable.

Table 9. MIPI-CSI and Camera Module Pin Configuration

MCU pin number	Signal Name	Comment
P108	MIPI_IF_EN	Initial State High, Transition to Low to enable MIPI-CSI Interface
MIPI_CLN, MIPI_CLP	MIPI_CLN, MIPI_CLP	Differential MIPI Clock Signal Pair
MIPI_DL0N, MIPI_DL0P	MIPI_DL0N, MIPI_DL0P	Differential MIPI Data Lane 0
MIPI_DL1N, MIPI_DL1P	MIPI_DL1N, MIPI_DL1P	Differential MIPI Data Lane 1
P709	CAM_RST	Camera Reset: Initial Low, Transition to High to enable Camera
P501	CAM_XCLK	Output from GPT Channel 12 for Camera Input Clock
P511	SDA1	IIC Channel 1 Data Line for Camera and Parallel LCD Touch Configuration
P512	SDA1	IIC Channel 1 Clock Line for Camera and Parallel LCD Touch Configuration

3.5.2 Parallel Graphics LCD Connections

Table 10 is the parallel graphics LCD control pin connection used in this application project. The clock, vertical and horizontal sync lines, and the data lines are not listed. Please refer to the stack configuration to understand all the data lines.

Table 10. Parallel Graphics LCD Pin Description

MCU pin number	Signal Name	Comment
P111	Touch_INT	LCD panel touch IRQ (IRQ19). Note that this line is only used in the Dual Core Cortex-M33 project
P606	DISP_RESET	Initialized to Low -> Transition to High to release the LCD
P514	DISP_BLEN	Initialized to Low -> Transition to High to enable the LCD Backlight
P108	MIPI_IF_EN	Initialized to Low -> Transition to Low to enable MIPI Interface

3.5.3 SDRAM Usage

The SDRAM can be enabled using the BSP stack configuration. After that is done, the SDRAM address and signal lines will be configured through the pin configuration and FSP driver.

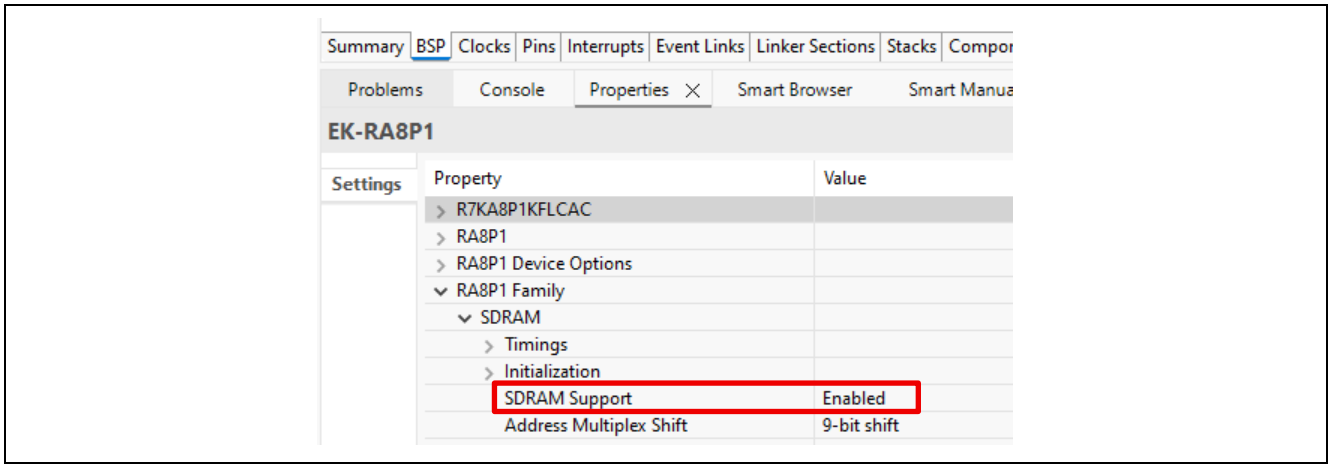

Figure 5. Enable SDRAM Usage from BSP Stack

3.5.3.1 Using the MobileNet V1 0.25 Model

When using the MobileNet V1 0.25 model, the VIN output buffers are allocated to the .sdram_noinit section.

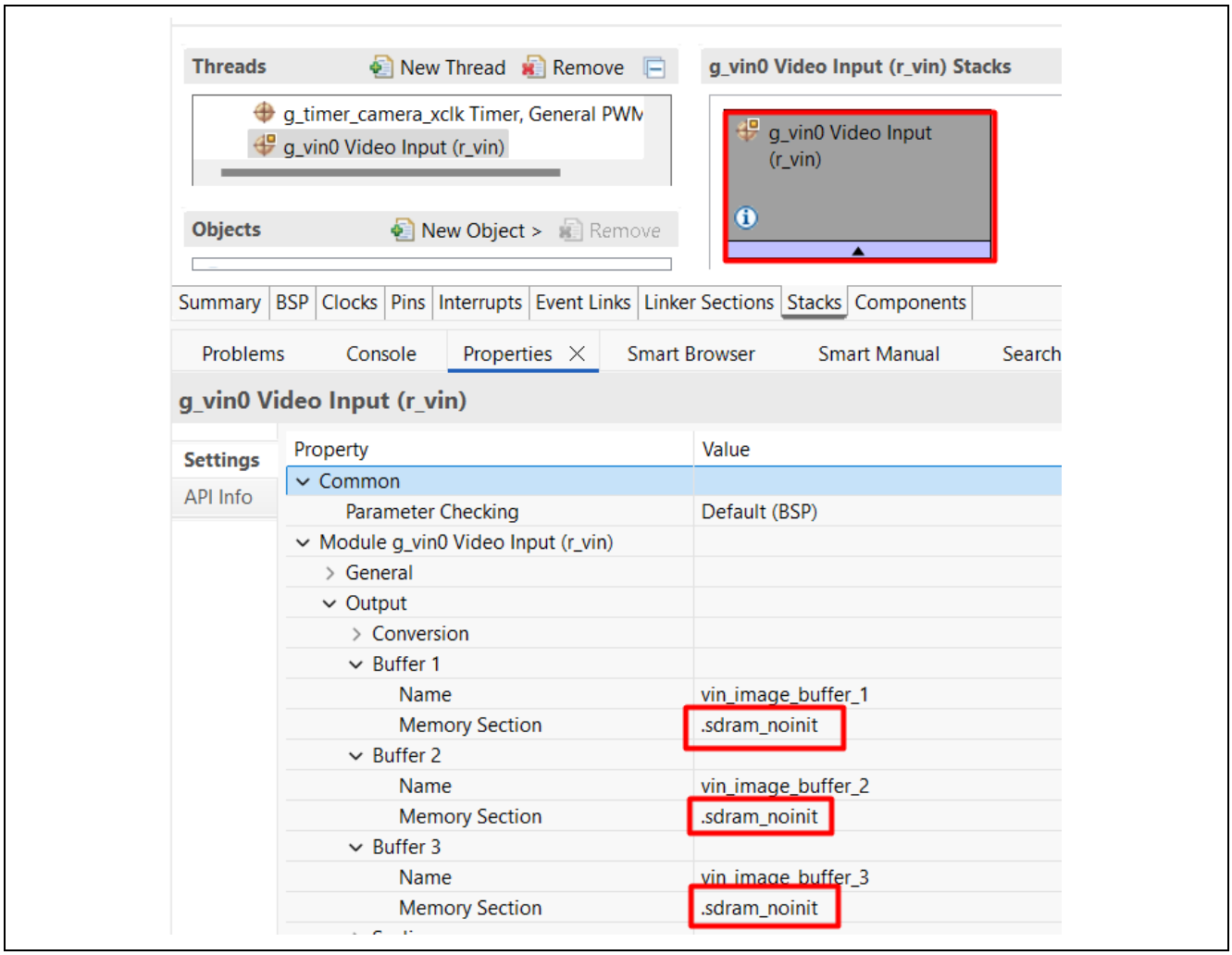

Figure 6. Allocate the VIN Output to SDRAM

The LCD framebuffers are also allocated to the SDRAM region. Although the fb_background framebuffer is used in the initialization stage based on the current GLCDC driver requirement; the VIN output buffers are directly displayed in layer 1 afterwards to reduce access frequencies to the SDRAM area.

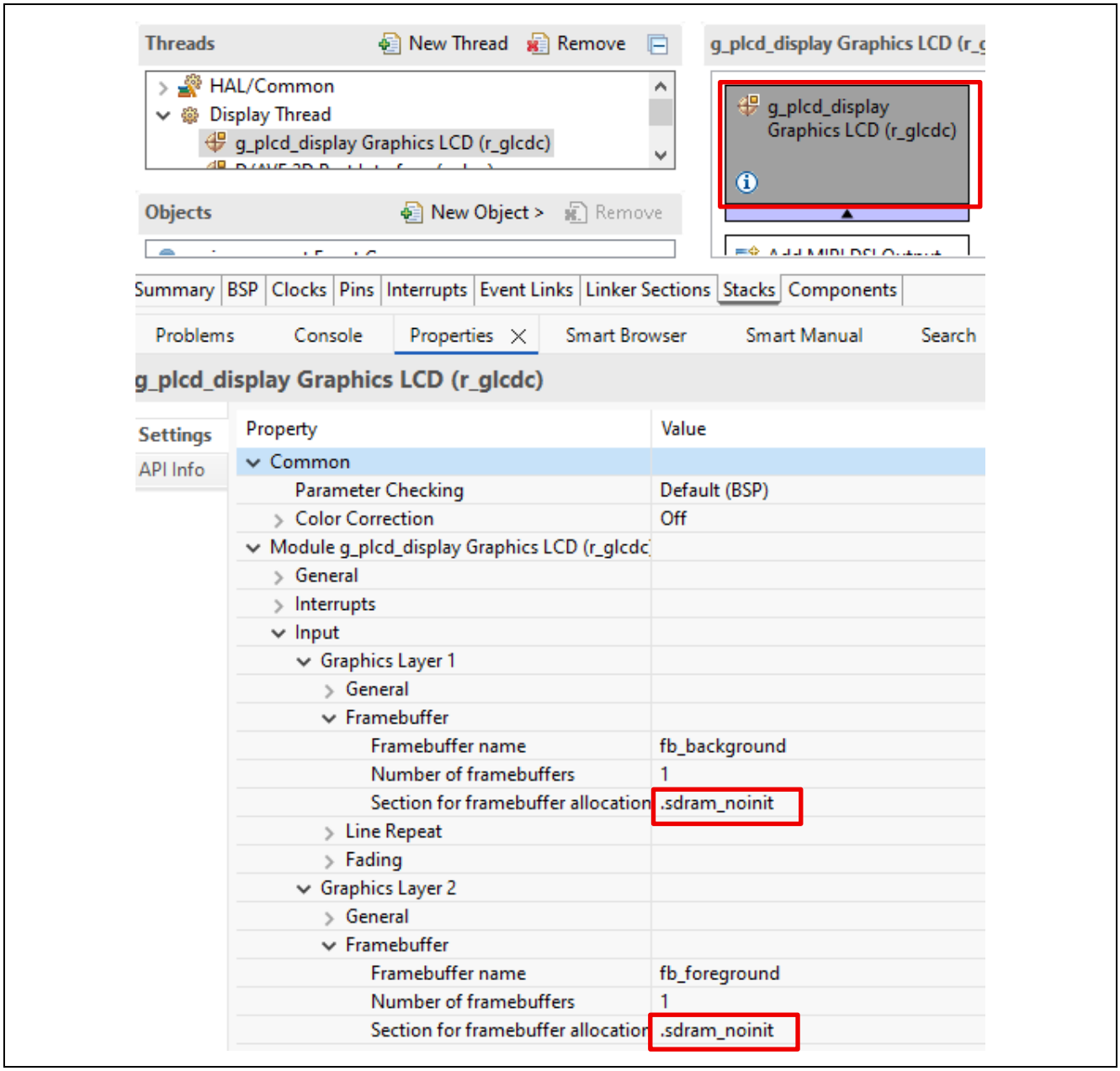

Figure 7. Allocate the LCD Framebuffers to the SDRAM Region

3.5.3.2 Using the EfficientNet Lite0 Model

Inference using the Ethos-U NPU

When compiling for Ethos-U NPU inference using the EfficientNet Lite0 model, the Tensor Arena area and the AI model storage memory allocated to the .sdram section. The following code is generated from the RUHMI compilation process. The RUHMI compiler can evaluate the Tensor Arena area needed and perform the memory allocation.

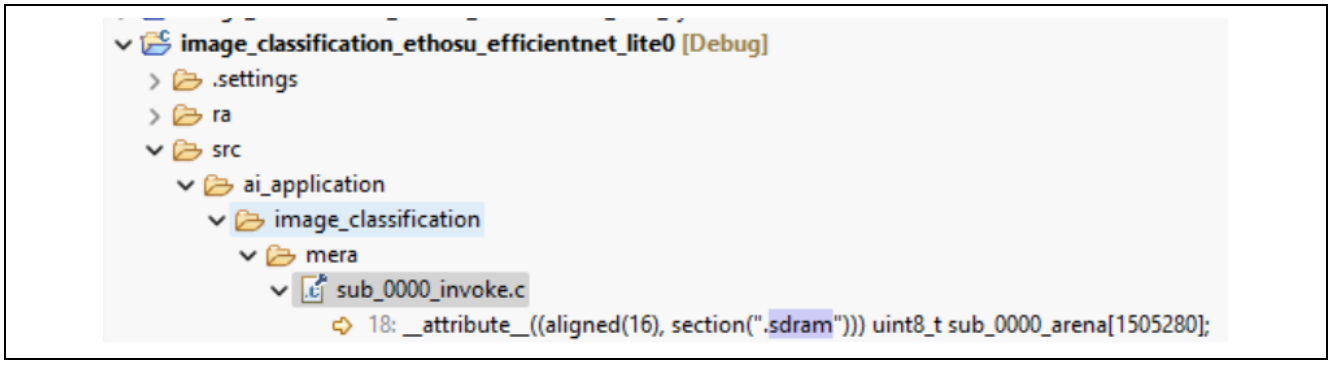

Figure 8. Allocate the Tensor Arena to the SDRAM Region for Ethos-U Inference

The AI model is programmed to the OSPI and then initialized to the SDRAM by FSP prior to the execution of the user application program. The RUHMI compiler can allocate the AI model during the compilation process and assign the model data to the appropriate memory section.

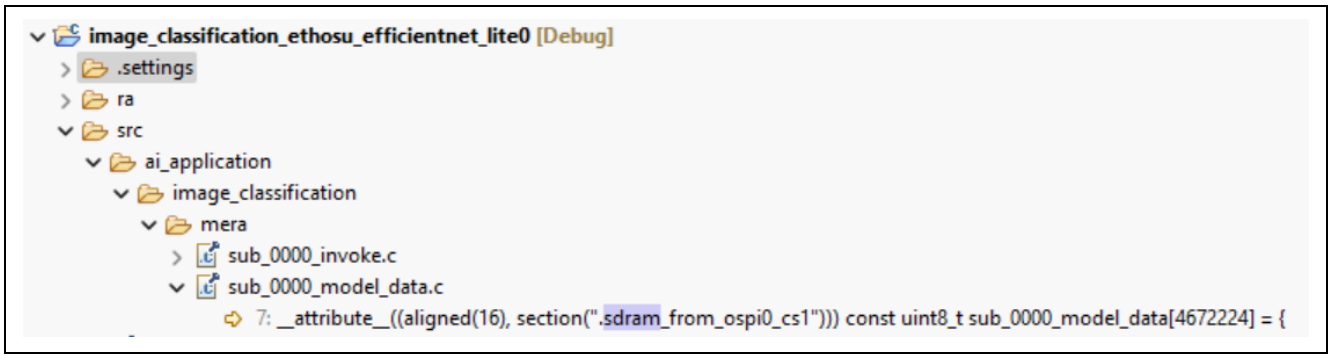

Figure 9. Allocate the AI Model to OSPI for Runtime Copying to SDRAM for Ethos-U Inference

The GLCDC first layer fb_background is allocated the SRAM region and the second layer fb_foreground buffer is allocated to the SDRAM region.

3.5.4 J-Link Virtual Console

Table 11 is the J-Link Virtual Console pin connection used in the Application Project.

Table 11. J-Link Virtual Console Pin Description

MCU pin number	Signal Name	Comment
PD03	RXD8	Reception pin
PD02	TXD8	Transmit pin

3.6 The Image Classification Model

The image classification model used in this application project is a quantized MobileNet V1 0.25 model. Refer to the Appendix section to understand how to access the float model and quantize the model for the RUHMI Framework. The dataset used for training this model is the ImageNet dataset. Refer to the following page to understand this dataset.

https://www.image-net.org/download.php

Reference the Renesas RUHMI Framework Quick Start Guide and the RUHMI Github landing page to understand how to use the RUHMI framework to compile the generated integer mode. The compilation results are included in the \ai_application\image_classification\mera folder.

3.7 OV5640 Camera Module Usage

This application project utilizes the OV5640 camera included in the EK-RA8P1 kit to gather camera data via the MIPI-CSI interface. On the EK-RA8P1 board, the image sensor input clock is provided by the GPT timer channel 12 from the MCU. This clock is used by OV5640 to enable image sensing.

Color Format and Framerate Configuration

The OV5640 has an active array size of 2592x1944. The supported output formats are YUV (422/420) and YCbCr422, RGB565/555/444, and 8-bit compression data. This application project uses YUV422 output from the camera and converts to RGB565 by the MCU VIN module to achieve reasonable throughput on the SDRAM traffic. The output format of the OV5640 is configured through a set of registers via the I2C interface. For more details on the OV5640 specification and the descriptions of the control registers, please contact the manufacturer. Only the usage related to this example project is described. Renesas is not responsible for any issues generated when referencing the register descriptions provided in this write-up.

The maximum frame rate of the OV5640 depends on the resolution of the image data collected. This application project uses the default register setting, which has auto-exposure enabled. This application project uses the default register setting, which does not scale the camera image prior to feeding it to the MCU through the MIPI-CSI interface.

Output Size Configuration

The following group of parameters controls the final camera output image size. This size should be no larger than the scaling output size. In this image classification application, the same size is used for both the Ethos-U Inference and the CPU Inference projects.

Table 12. Configure the Camera Output (IMAGE_WIDTH 640; IMAGE_HEIGHT 480)

Register number	Description	Default setting (QXGA)	Application Setting (VGA)
0x3808	DVP output horizontal width[11:8]	0x0A	(uint8_t)(IMAGE_WIDTH>>8))
0x3809	DVP output horizontal width[7:0]	0x20	(uint8_t)IMAGE_WIDTH)
0x380a	DVP output vertical height[10:8]	0x07	(uint8_t)(IMAGE_HEIGHT>>8))
0x380b	DVP output vertical height[7:0]	0x00	(uint8_t)IMAGE_HEIGHT)

Users can reference `\src\camera_layer\camera_control.c` to understand the details of the camera configuration.

3.8 MIPI-CSI and VIN Interface

The MIPI-CSI and VIN Interface is configured through the FSP stack. Its configuration should match the camera output (Table 12) in terms of the image format and size.

In this example project, the camera is used to fetch RGB565 images with 640x480 resolution. An interrupt is triggered when a complete camera image frame is captured. The following are the key stack configurations for this example usage. Refer to the section VIN in the FSP User's Manual to understand the other VIN configurations. By default, the VIN stack uses a triple-output buffer configuration. This setting is used in this application project.

Data Storage of the Camera Image

With the SRAM primarily allocated for the AI tensorArena, the thread stack and heap usage, the camera image output buffers are allocated in the SDRAM.

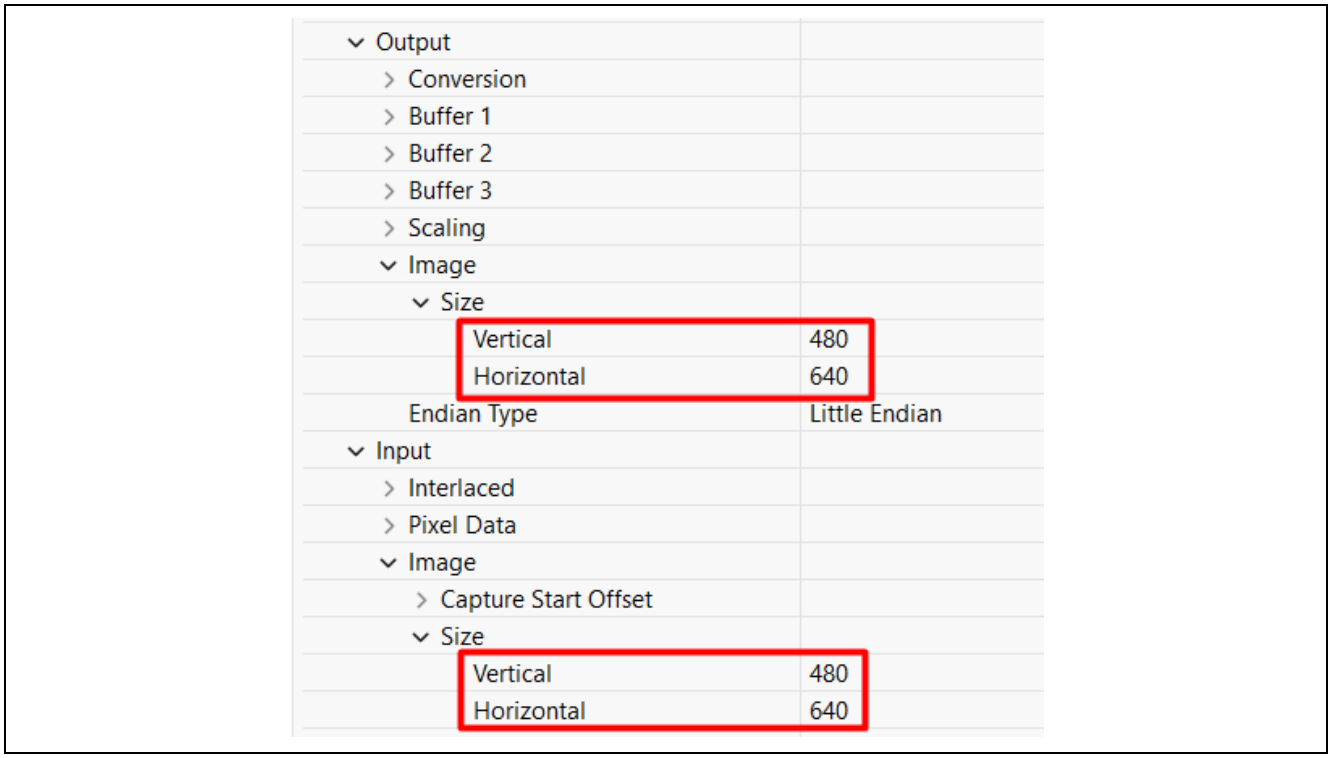

Figure 10. Video Input Module Configurations

3.9 AI Inference Pre-Processing and Post-Processing

The following diagram captures the Pre-Processing and Post-Processing of the AI Inference. For performance consideration, the processed AI inference image is stored in SRAM when using both the MobileNet V1 0.25 model and the EfficientNet Lite0 model.

3.9.1 Pre-Processing and Post-Processing for MobileNet V1 0.25

The following is the pre-processing and post-processing when using MobileNet v1 0.25. The VIN output for this implementation is 640x480. Only the middle 480x480 of the original VGA image is sent to the AI inference engine, the image classification will be effective in the middle 480x480 of the active window.

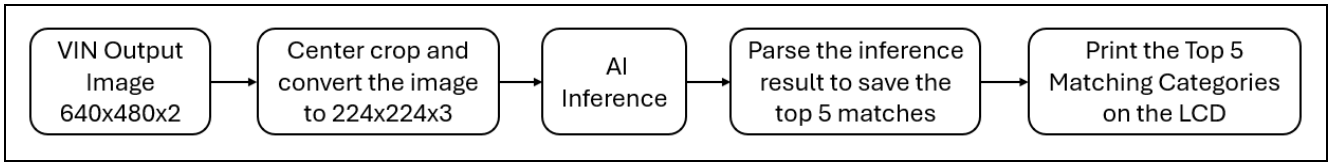

Figure 11. Pre and Post-Processing for AI Inference for Mobilenet V1 0.25

The following routine is used for image processing. Note that this routine is targeted to process 16bpp images to 24bpp images; it should be updated if other input and output formats are used.

```
/*******************************************************************************************************************
 *  Crop square, downsample and convert 640x480 RGB565 to 224x224 RGB888 image.
 *  @param[IN]   const void * p_input_image_buff: input RGB565 image
 *  @param[IN]   void * p_output_image_buff: output RGB888 image
 *  @param[IN]   uint16_t in_width: width of the input buffer
 *  @param[IN]   uint16_t in_height: height of the input buffer
 *  @param[IN]   uint16_t out_width: width of the output buffer
 *  @param[IN]   uint16_t out_height: height of the output buffer
 ******************************************************************************************************************/
vision_ai_app_err_t image_rgb565_to_rgb888(const void *p_input_image_buff, void *p_output_image_buff,
                                           uint16_t in_width, uint16_t in_height,
                                           uint16_t out_width, uint16_t out_height) {
```

Figure 12. Software API to process camera image data for inference

3.9.2 Pre-Processing and Post-Processing for EfficientNet Lite0

The following is the pre-processing and post-processing when using EfficientNet Lite0. The VIN output for this implementation is 320x240. Only the middle 240x240 of the original VGA image is sent to the AI inference engine. DRW is used scale the image from 320x240 to 640x480 prior to displaying on the LCD. When viewing the image on the LCD, the image classification will be effective in the middle 480x480 of the active window.

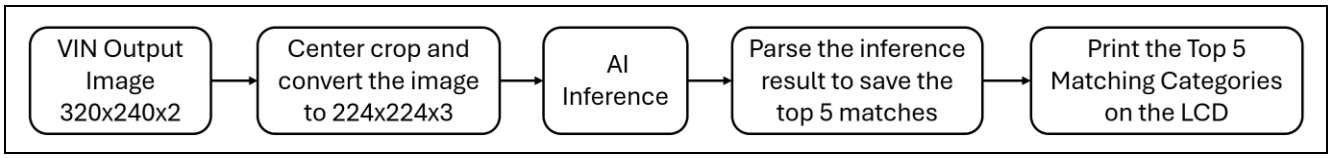

Figure 13. Pre- and Post-Processing for AI Inference for EfficientNet Lite0

3.10 Displaying the Camera Image and Inference Results on the Parallel Graphics LCD

For the MobileNet V1 0.25 model, the camera output saved by the VIN module at 640x480 RGB565 is directly displayed on the LCD without any scaling or cropping.

For the EfficientNet Lite0 model, the camera output is scaled by the DRW from 320x240 RGB565 to 640x480 RGB565 prior to display on the LCD.

The project includes a text printing service capable of printing English alphabet characters from 'a' to 'z' in both lowercase and uppercase, as well as numeric digits from '0' to '9'. The service is also able to print several special characters, such as %, -. The text image buffers are generated using the open-source GIMP tool and are allocated to array ascii_table_flash in the code flash (refer to \src\graphics\bg_font_18_full.c). The display thread accesses the AI model inference result and displays the result on the parallel graphics LCD using this text printing service.

The following API prints a string to the parallel graphics LCD using DRW:

```
/*****************************************************************************************************************************
 * Use DAVE/2D to print a string onto the mipi lcd.
 *  @param[IN]   d2_device *handle: acquired with d2_opendevice
 *  @param[IN]   d2_point _xs: coordinate of the character on the horizontal line
 *  @param[IN]   d2_point _ys: coordinate of the character on the vertical line
 *  @param[IN]   char * str: the string to be printed on the mipi lcd
 *  @retval      None
 *****************************************************************************************************************************/

void print_bg_font_18(d2_device *handle, d2_point _xs, d2_point _ys, char *_str)
```

Figure 14. Print Text to Parallel Graphics LCD

There are several asynchronous interrupt service routine callbacks in the system. Some critical system synchronizations are handled in these interrupt routines.

- VIN ISR Callback: An interrupt is triggered when a complete camera image frame is captured. The system switches to the next buffer in a triple-buffer system to collect a new image.
- GLCDC ISR Callback: An interrupt is triggered when a Vertical Sync display signal arrives. The system signals the display thread to render a new camera image.
- MIPI Command Transfer ISR Callback: This interrupt facilitates the MIPI-CSI data collection from the OV5640. There is no significant system handling in this application project in this ISR. Additional error handling and debugging information can be added to this ISR in case the system runs into any issues with MIPI-CSI communication.

4. Running the Image Classification Project

This section describes how to run and evaluate the image classification application.

4.1 Setting Up the Hardware and the IDE

The EK-RA8P1 package includes all the hardware components needed to run this image classification application.

- EK-RA8P1 board
- USB Type-C cable. This is used to connect with the Debug USB port on the EK-RA8P1 and the PC's USB port. We recommend using a USB Type-C to Type-C cable for a reliable power supply to the system.
- Camera module and ribbon cable
- Parallel Graphics LCD Expansion Board
- Display mounting hardware (spacers and fixing screws). Prior to running this image classification application, the default jumper and switch settings must be utilized. Refer to the EK-RA8P1 user's manual for the default jumper and switch settings.
- To run the EfficientNet Lite0 projects, ensure SW4-3 is set to OFF.

Connect the OV5640 Camera Module with EK-RA8P1

To connect the camera module with the EK-RA8P1 kit, follow the steps:

1. Connect one end of the ribbon cable included to the bottom of the EK-RA8P1 kit: Camera Expansion Port J35.
2. Connect the other end of the ribbon cable to the camera module.
3. It is highly recommended to use the Nylon stub included in the EK-RA8P1 kit to secure the camera from movement.

Be careful not to apply excessive force to the ribbon cable, and make sure to protect the camera lens.

Connect the Parallel Graphics LCD with the EK-RA8P1

The parallel graphics display board features a 1024 x 600 TFT LCD with a capacitive touchscreen connected to the RA MCU using the Parallel Graphics Expansion Port (J1). It is highly recommended to secure the LCD Graphic Expansion board using the Nylon stands and screws if they are available. Reference Figure 15 for the orientation of the LCD.

Once the camera and parallel graphics LCD are assembled, connect J10 to the development PC to provide power, debug communication, and J-Link console communication with the PC.

Figure 15 shows the components used in the application:

1. EK-RA8P1 v1 board

2. USB-C to USB-A cable

3. Parallel Graphics Expansion Board 1

4. Camera FFC cable

5. Camera Expansion Board

6. Display mounting hardware (spacers and fixing screws)

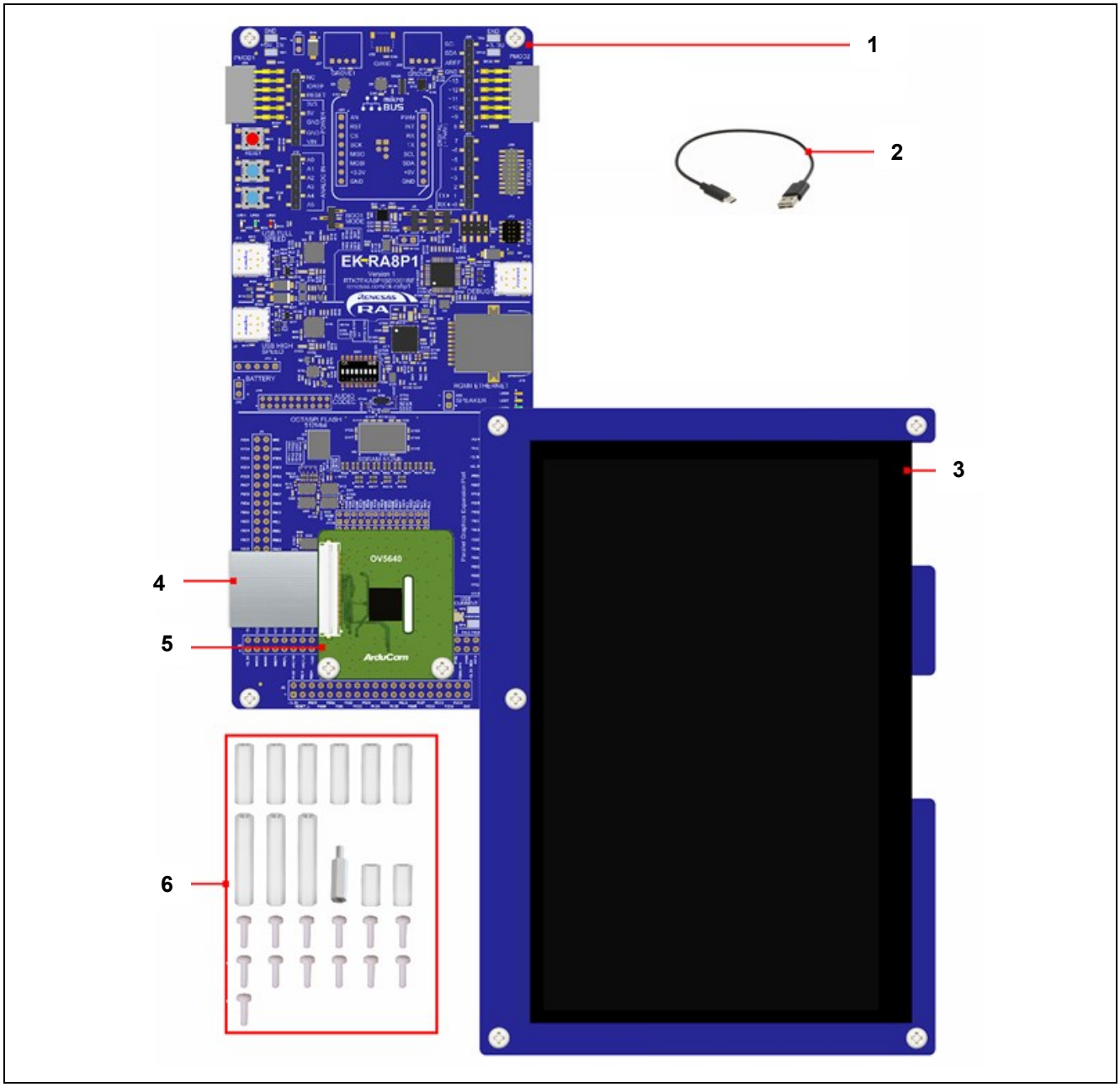

Figure 15. Assemble the Hardware System

Set up the IDE

Set up the development environment following the section "**Software and development tools"** on the first page.

4.2 Importing the Projects

Follow the "Importing an Existing Project into e² studio" section in the FSP User's Manual to import the application projects.

Please Note the following:

- Place the project folder near the root of the drive (e.g., C:\Project) to avoid exceeding the maximum path length limit on Windows.
- Use a separate e2studio workspace for the EfficientNet projects and the MobileNet V1 projects as the dual core projects use same project names. This will be improved in a future release.

4.2.1 Dual Core Projects for the MobileNet V1 0.25 Model Overview

There should be three projects imported into the workspace.

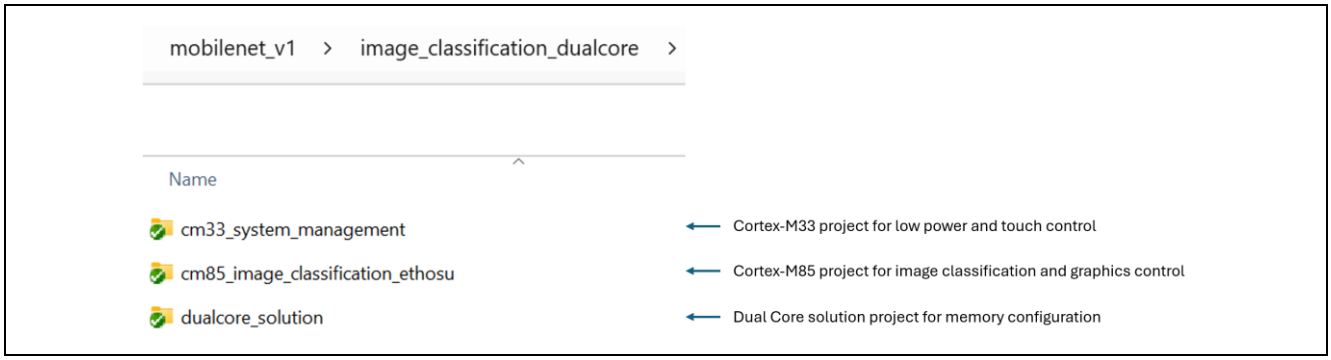

Figure 16. Image Classification Dual Core Embedded Code for MobileNet V1 0.25

The following are some additional details on the application code.

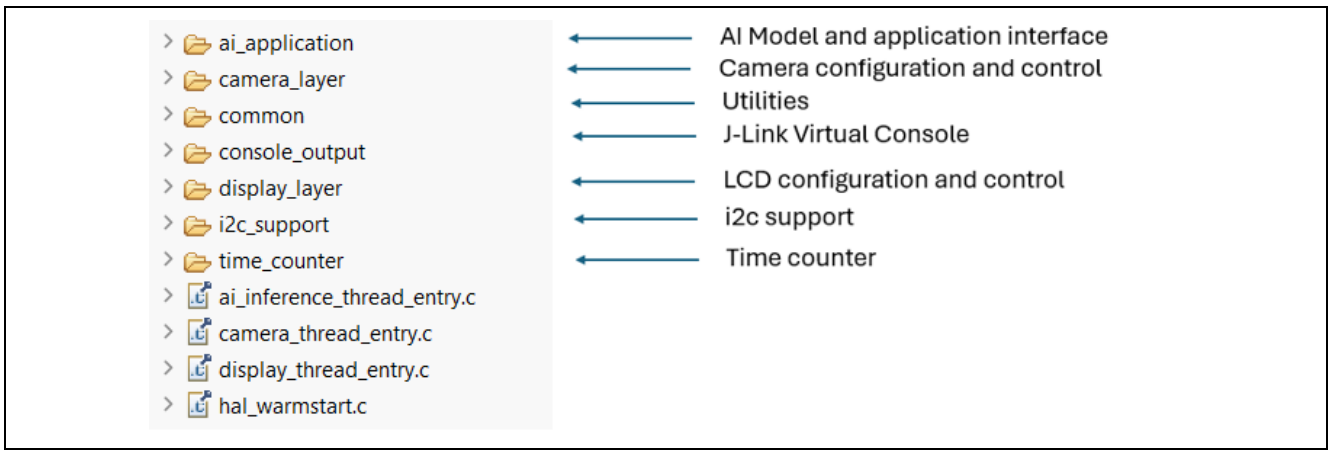

Figure 17. Image Classification Application Code for Cortex-M85

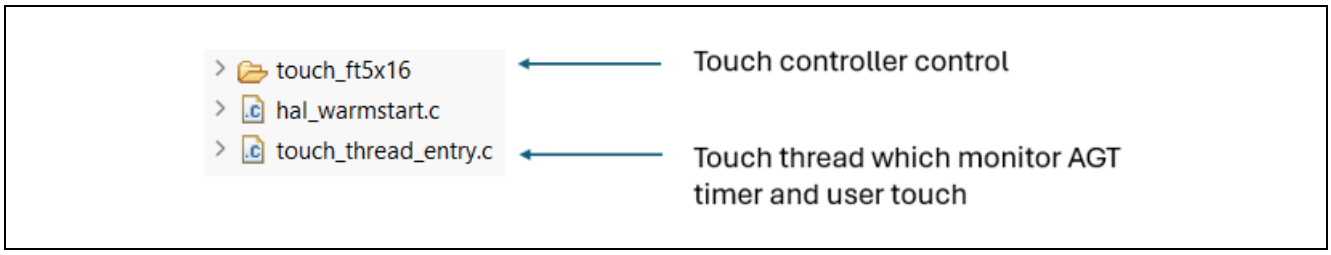

Figure 18. Image Classification Application Code for Cortex-M33

4.2.2 Dual Core Projects for the EfficientNet Lite0 Model Overview

There should be three projects imported into the workspace.

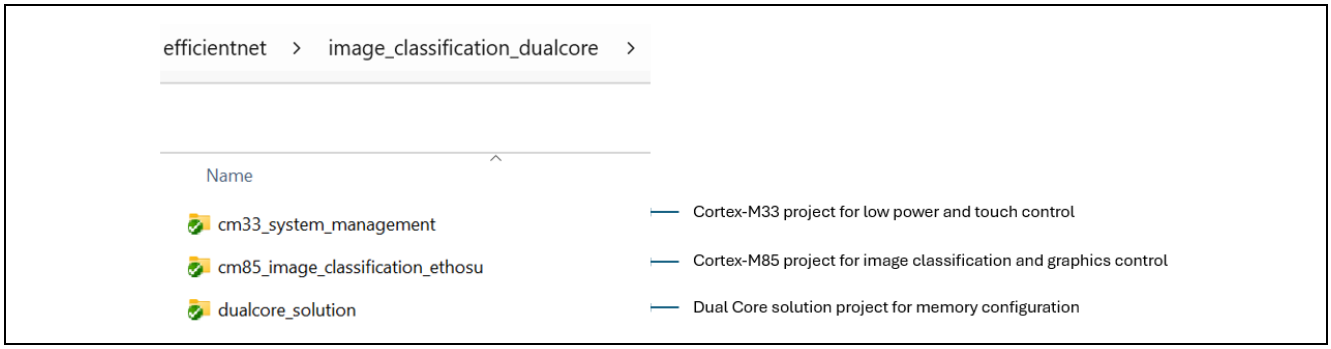

Figure 19. Image Classification Dual Core Embedded Code for EfficientNet Lite0

The following are some additional details on the application code.

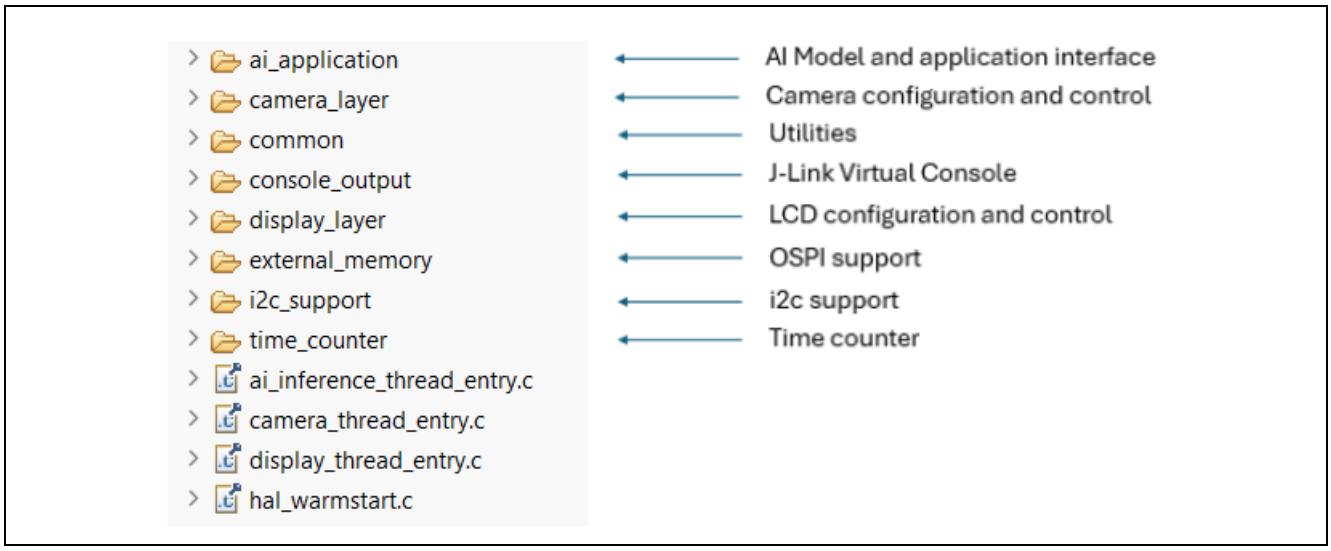

Figure 20. Image Classification Application Code for Cortex-M85

The same code for Cortex-M33 is used for the EfficientNet Lite0 and MobileNet V1 0.25 model.

4.3 Initialize the MCU

It is important to note that prior to running any of the application projects, the MCU must be initialized. This can be achieved by using the Renesas Device Partition Manager (RDPM), which is a built-in tool of the e^2 studio IDE.

Launch e^2 studio, select Run -> Renesas Debug Tools -> Renesas Device Partition Manager. Use the following settings to initialize the MCU. Ensure the operation is successful.

In case the dual core projects are imported, or you have other dual core projects in the workspace, the RDPM should be launched by highlighting a Cortex-M85 or a Cortex-M33 project rather than the dual core solution project; otherwise, the RDPM might not launch.

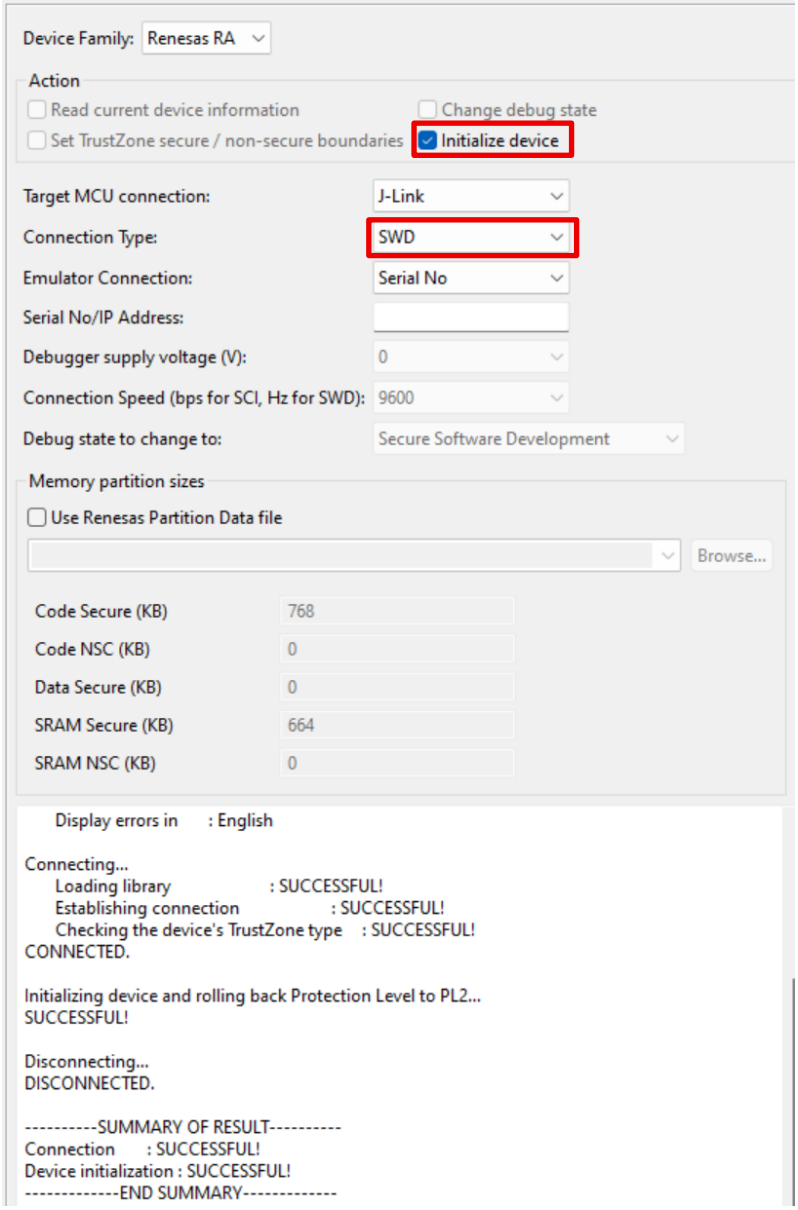

Figure 21. Initialize the MCU

4.4 Compile and Run the Dual-Core Ethos-U Inference Application Projects

The dual-core projects are only provided for the Ethos-U55 inference use case.

Note: There is a stack error message in the Cortex-M33 project. This error message will be resolved in the future FSP version; it can be ignored. This is not a functioning issue for the system setup.

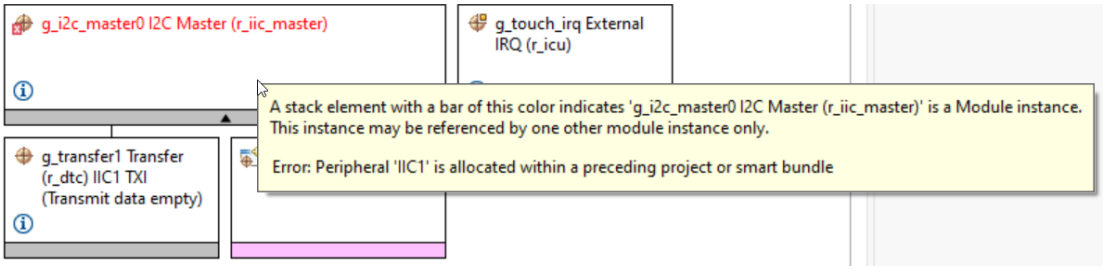

Figure 22. IIC Stack

After the projects are imported, select the **dualcore_solution** project, right-click, and choose Build Project. Wait until both the Cortex-M85 and Cortex-M33 projects are built. This can take about 5 minutes for the first time. There are warnings from third-party software. The screenshots in this section use the MobileNet V1 projects as examples; the general idea also applies to the EfficientNet projectsNext, open the **Debug Configurations** page and select to debug from the Group Launch.

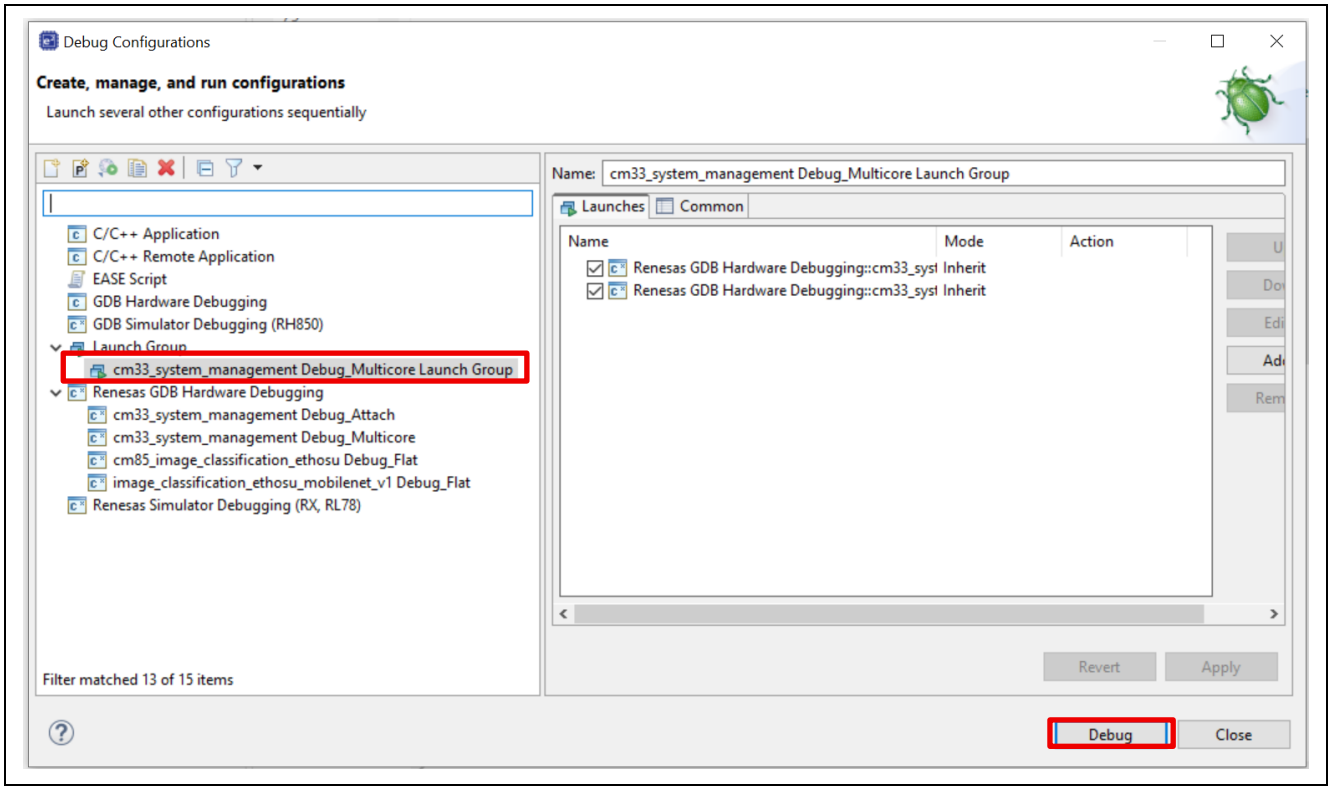

Figure 23. Debug the Dual Core Project

After the debug session is launched, the Cortex-M85 will be released from Reset first and start executing the application code. Click Resume twice to run the Cortex-M85 application.

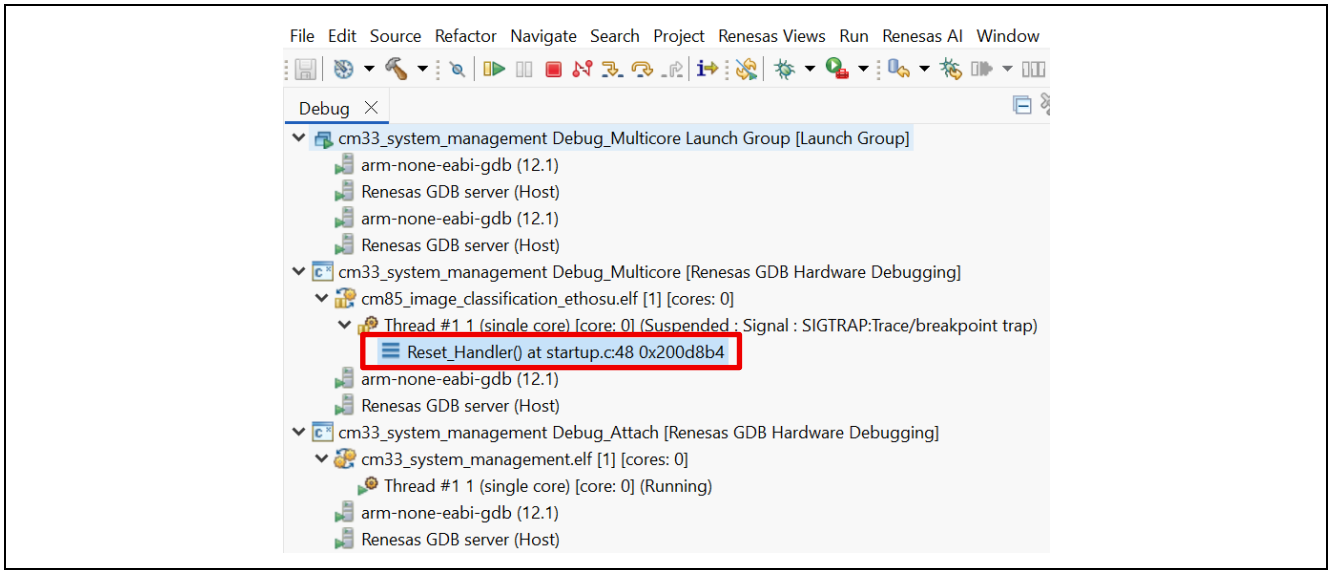

Figure 24. Release the Cortex-M85 from Reset_Handler

Once the Reset_Handler of the Cortex-M33 project is highlighted by the debug session, click Resume to run the Cortex-M33 application.

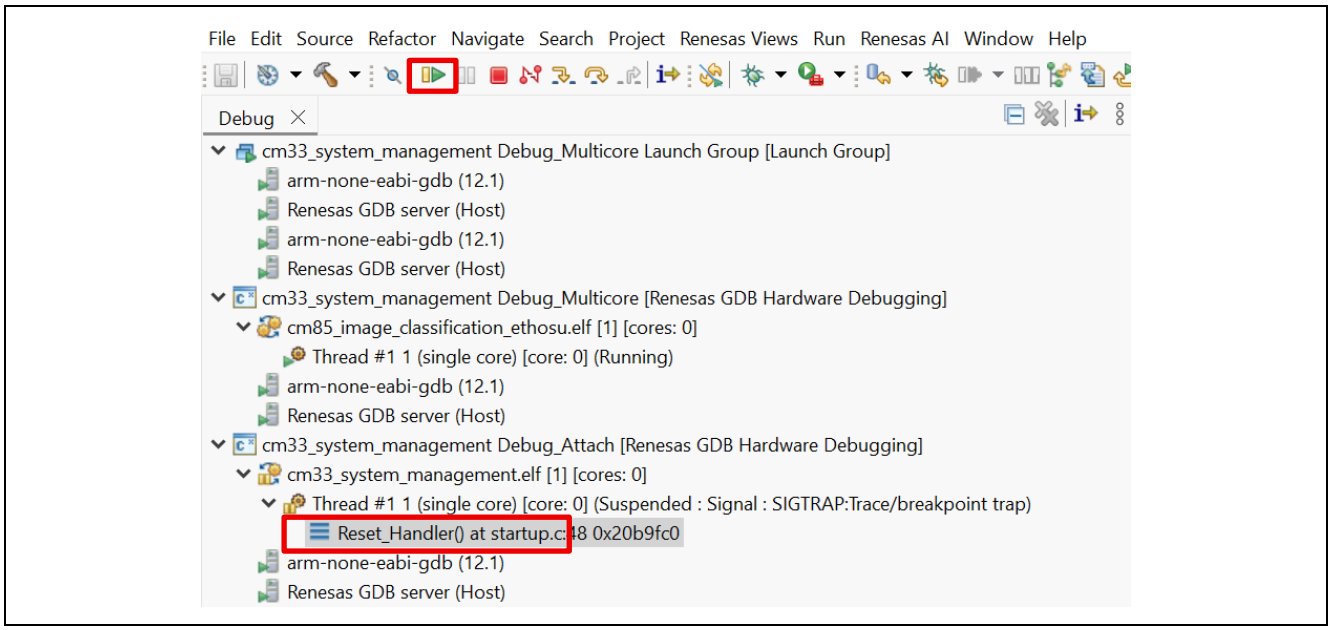

Figure 25. Release the Cortex-M33 from Reset_Handler

Refer to section 4.6 to understand the limitations on launching the debug session when running the EfficientNet projects.

Please refer to section 4.7 to evaluate the inference performance of the system.

4.5 Compile and Run the Single-Core Ethos-U and CPU Inference on Cortex-M85

The architecture of the single core projects is very similar to the Cortex-M85 projects in the dual core system. They are not repeated here.

The procedure to compile and run the single-core Ethos-U and CPU inference projects is similar regardless of which model is used.

After the projects are imported, select one of the following projects

- image_classification_cpu_mobilenet_v1
- image_classification_ethosu_mobilenet_v1
- image_classification_cpu_efficientnet_lite0
- image_classification_ethosu_efficientnet_lite0

Right-click and choose Build Project.

Once the project is compiled, right-click on the project and select Debug As -> Renesas GDB Hardware Debugging to launch the debug session. Reference section 4.5 to evaluate the performance of the Ethos-U inference.

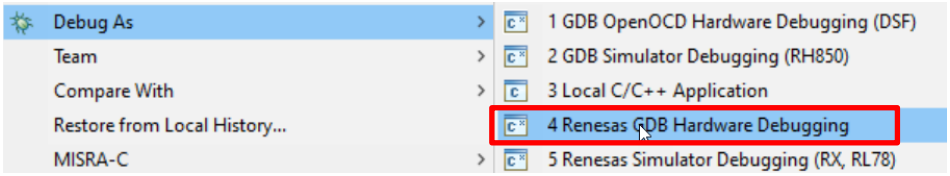

Figure 26. Launch the Ethos-U Inference-based Image Classification

4.6 Limitations When Downloading the Application

When downloading the applications using the EfficientNet model, the debugger will time out with the following error message for both the single core and dual core projects. This is expected and will be fixed in

future tools update. To resolve the error, wait until the download finishes, and then start the debug session a second time; the launch will proceed as expected. The screenshot in **Figure 27** uses the dual core project as an example.

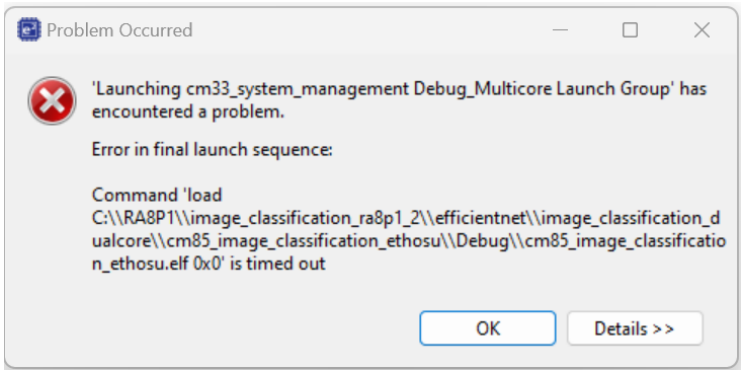

Figure 27. Launch the Ethos-U Inference-based Image Classification

4.7 Observe the System Performance using the LCD Output

Observe the LCD image classification result display and present an object or picture from one of the options in \image_classification\labels.c to see the image classification in action. Some of the objects that have been tested are fish, fruits, and musical instruments.

Note that every object presented in the middle 480 x 480 pixels of the camera output/LCD active screen will be classified. For a classification focused on one particular object, use a clean background and present that particular object to the camera.

By default, after 30 seconds, the LCD screen will go black as the CM85 enters Deep Sleep mode. The AGT timer period can be updated to adjust this operational period. Meanwhile, the debug connection of both Cortex-M85 and Cortex-M33 will be terminated.

Touching the parallel graphics LCD panel will wake up the Cortex-M85 to resume the image classification.

4.8 Observe the System Statistics using the J-Link Virtual Console

The J-Llink console using UART8 on PD02 and PD03 is used to display the system information configured with a baud rate of 230400. The J-Link console is accessed through the same USB Debug connection from the PC. It can be used concurrently during a debug session or in a standalone session. Tera Term terminal version 4.106 is used during the project evaluation.

Launch **Tera Term** and select the enumerated COM port (J-Link CDC UART Port).

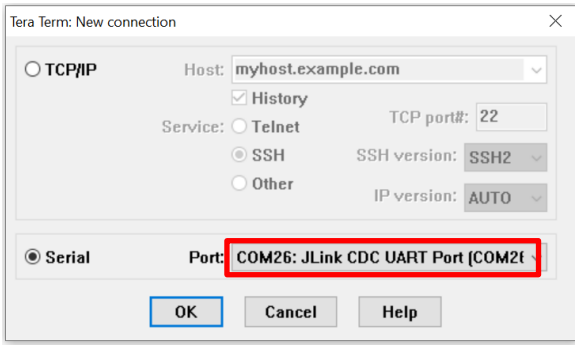

Figure 28. Select the JLink Console Connection

Once the COM port is open, navigate to the **Setup** tab and select **Serial port**.

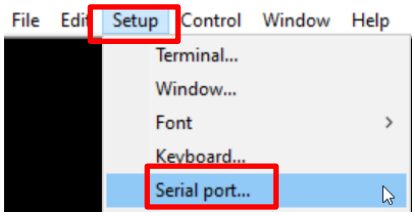

Figure 29. Open the "Serial port" interface

Update the Speed to 230400 and click **New setting** to commit the update.

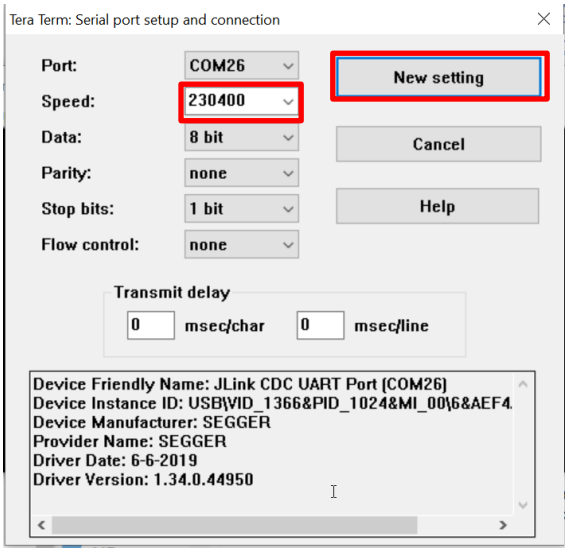

Figure 30. Configure the Tera Term

The time reported here includes the influence of the software design and the SDRAM being accessed by multiple peripherals. This utility is provided to assist with system debugging.

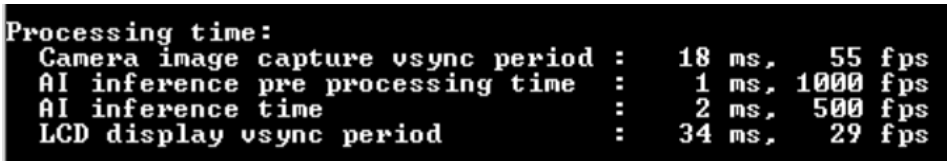

Figure 31. Typical Statistical Output for Ethos-U Inference when using MobileNet V1 0.25

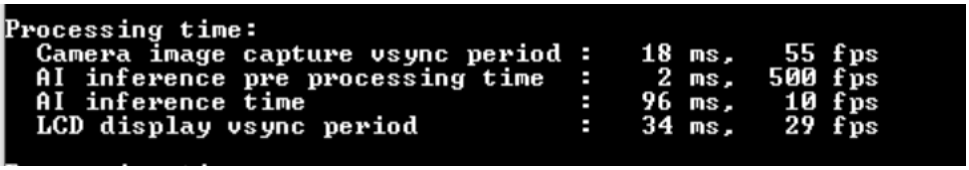

Figure 32. Typical Statistical Output for CPU Inference when using MobileNet V1 0.25

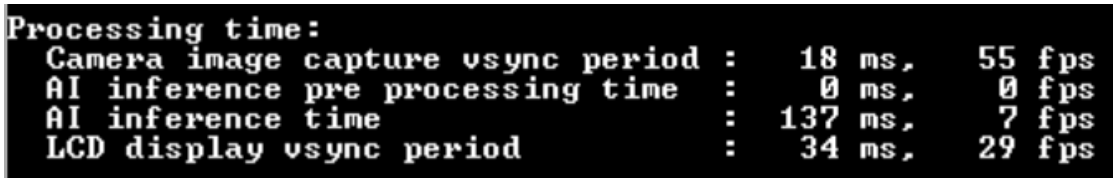

Figure 33. Typical Statistical Output for Ethos-U Inference when using EfficientNet Lite0

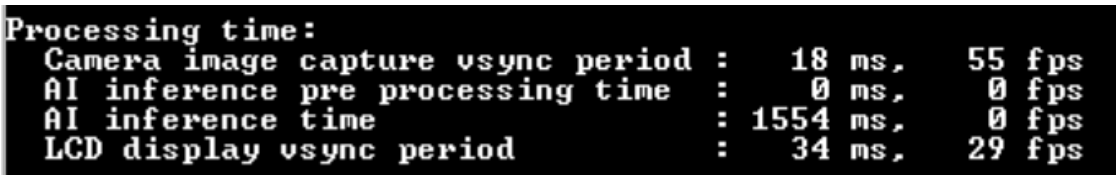

Figure 34. Typical Statistical Output for CPU Inference when using EfficientNet Lite0

4.9 Summarize the Ethos-U Inference Performance Uplifting

The following table captures the system performance with inferencing performed by the Ethos-U55 and the CPU.

For details on memory allocation usage, please refer to section 3.4. This information may change when a different model or a different MCU is used in an image classification. The CPU-based inference benefits from data caching due to frequent access to weights and intermediate tensors. In contrast, Ethos-U accesses memory through tightly managed DMA or AXI interfaces, and its performance is more dependent on external memory bandwidth and not directly affected by the CPU's data cache. In all cases, using the Ethos-U NPU brings in a significant performance uplift compared with the CPU inference.

Table 13. Memory Usage and System Performance using MobileNet V1 0.25

Application	Core Frequencies (CPU /Ethos-U55)	MRAM (KB)	SDRAM (KB)	SRAM (KB)	MACs/Inference (based on output from RUHMI compilation result)	Inference time (ms)		Frames /second (Display)	Frames /second (MIPI CSI)
RA8P1 (M85 w/Helium, inference on Ethos-U55)	1GHz /500MHz	See Table 3	See Table 5	See Table 4	41947984	1-2 (regardless of whether the data cache is enabled)		29	55
RA8P1 (M85 only w/Helium, inference using CPU)	1GHz /500MHz					data cache enabled	98	29	55
						data cache disabled	1885		

Table 14. Memory Usage and System Performance using EfficientNet Lite0

Application	Core Frequencies (CPU /Ethos-U55)	MRAM (KB)	SDRAM (KB)	SRAM (KB)	MACs/Inference (based on output from RUHMI compilation result)	Inference time (ms)	Frames /second (Display)	Frames /second (MIPI CSI)
RA8P1 (M85 w/Helium,	1GHz /500MHz	See Table 6	See Table 8	See Table 7	41947984	137 (regardless of whether the data cache is enabled)	29	55

inference on Ethos-U55)									
RA8P1 (M85 only w/Helium, inference using CPU)	1GHz /500MHz					data cache enabled	1553	29	55
						data cache disabled	19741		

4.10 Low Power Feature Summary

When the Cortex-M85 is in deep sleep mode, the core is halted. The application code also puts the Ethos-U and the MIPI-CSI into stop mode and disables the GLCDC output. When the Cortex-M33 is in sleep mode, the core is halted while the peripherals remain active. This leads to power saving in a real-world application with only the Cortex-M33 monitoring user actions.

The EK-RA8P1 board provides precision 5 mΩ resistors (Yageo, part number PS0612FKE070R005L) for current measurement of the main 3.3 V MCU power and the 3.3 V USB MCU power. Measure the voltage drop across these resistors and use Ohm's Law to calculate the current.

Note that these power consumptions are based on this application's project hardware configuration and software implementation with data cache enabled. Other applications may exhibit different power consumption characteristics.

The EfficientNet Lite0 model inference is stored in OSPI, uses less Code MRAM, runs at a much slower Frame Per Second and consumes less current.

Table 15. Power Consumption at Room Temperature (MobileNet V1 0.25)

	Voltage across R3 Accessible via TP1 and TP3 (matching current value)
Cortex-M85 in deep sleep mode Cortex-M33 in sleep mode	290 Micro Volt (58 mA)
Normal mode	380 Micro Volt (76 mA)

Table 16. Power Consumption at Room Temperature (EfficientNet Lite0)

	Voltage across R3 Accessible via TP1 and TP3 (matching current value)
Cortex-M85 in deep sleep mode Cortex-M33 in sleep mode	284 Micro Volt (57 mA)
Normal mode	356 Micro Volt (71 mA)

5. Appendix: Compile the MobileNet V1 0.25 Model for Embedded AI Inference

5.1 Quantize the MobileNet V1 0.25 Float Model

Within this application project, there is a Google Colab notebook file included: Quantize_MobileNet_v1_0_25_Model.ipynb. It can be executed in a Colab environment to generate a quantized Mobilenet_V1_0.25 int8 model.

The following steps to run this Colab Notebook:

1. Ensure the PC is connected to the Internet.
2. Open the Colab page: https://colab.research.google.com/
3. Choose File -> Open notebook -> Upload -> Browse and upload this notebook to the Colab environment.
4. Once the notebook is opened, click the array to run the notebook.

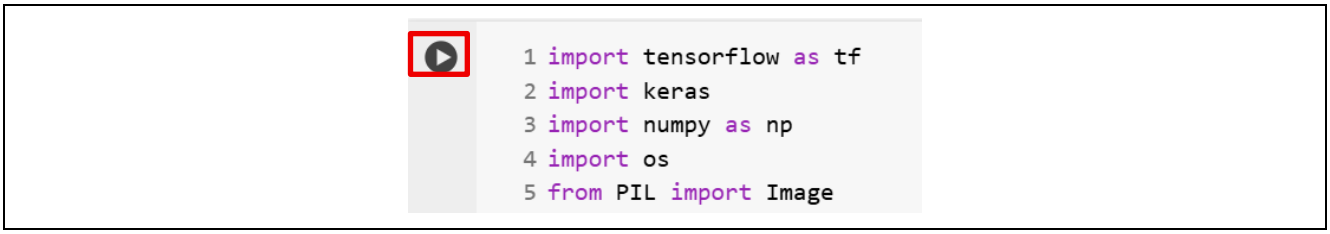

Figure 35. Run the Colab notebook Code

5. There will be a green tick mark on the left of the array once the execution is successfully executed.

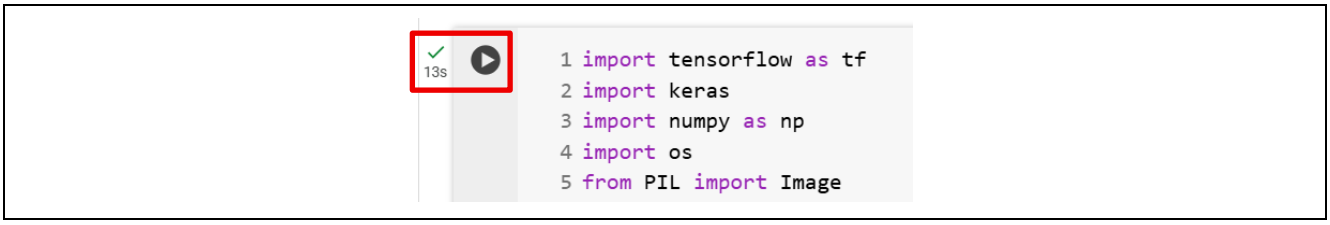

Figure 36. Successful Execution

6. Click the folder icon on the left, the generated mobilenet_v1.tflite will appear as shown below.

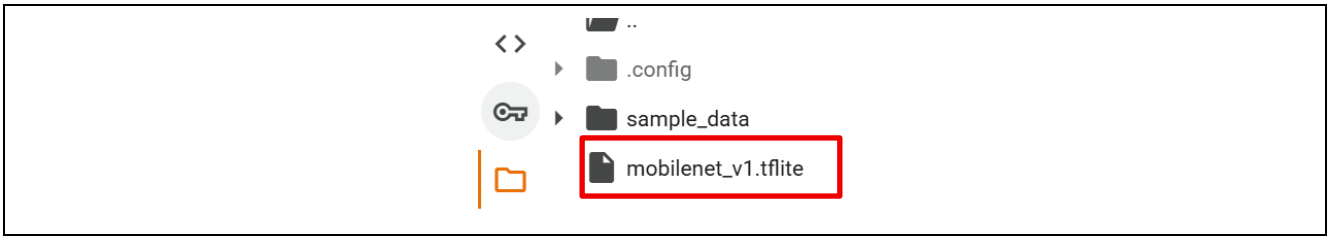

Figure 37. Successful Execution

5.2 Compile the EfficientNet Lite0 Model

The **EfficientNet-Lite0** quantized model used in this application is available at: https://github.com/joonb14/TFLiteClassification/tree/main. A couple of test images are also provided on that page.

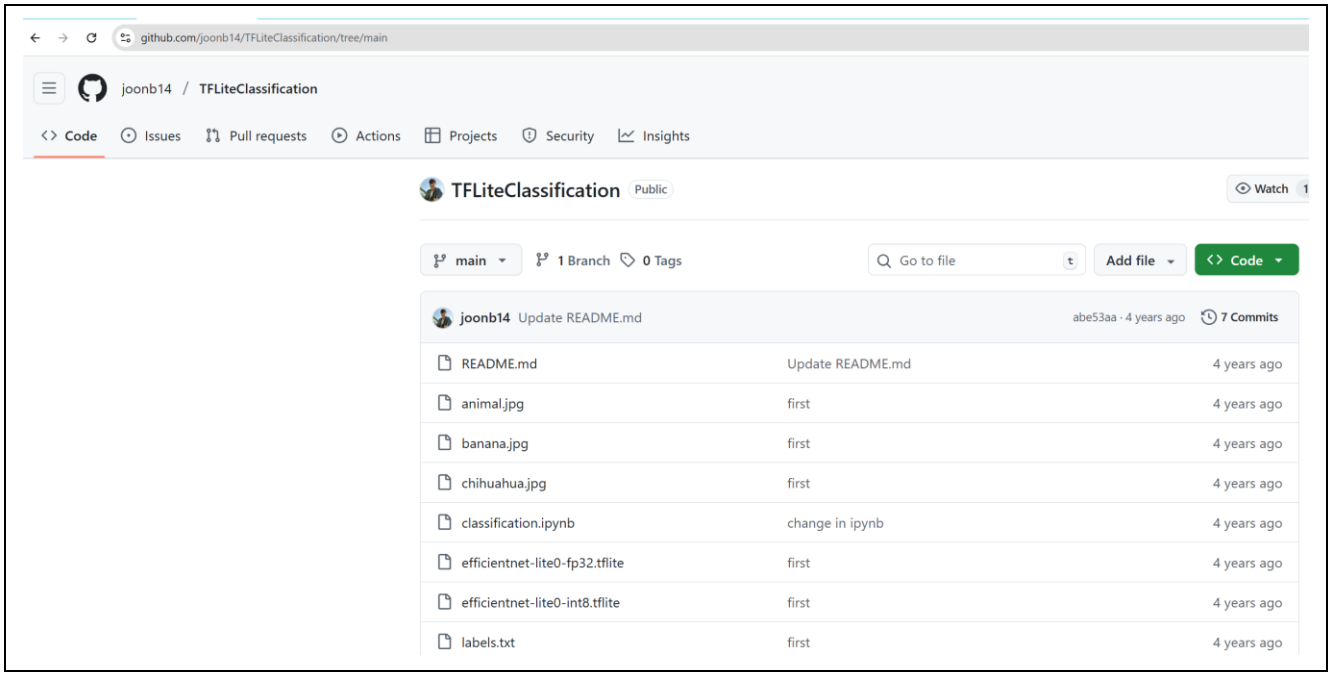

Figure 38. Download the EfficientNet Lite0 int8 Model

5.3 Use the Renesas RUHMI Framework to Compile the Model for Ethos-U and CPU Deployment

Refer to https://github.com/renesas/ruhmi-framework-mcu/tree/main for instructions on how to set up RUHMI and compile the generated mobilenet_v1.tflite mode for Ethos-U and CPU Inference.

The compilation result for Ethos-U NPU Inference is located under this folder using the Dual Core project as reference: \cm85_image_classification_ethosu\src\ai_application\image_classification\mera\.

The compilation result for CPU Inference is located under this folder:\image_classification_cpu_mobilenet_v1\src\ai_application\image_classification\mera.

5.4 Utilizing Cortex®-M85 Core Data Cache

In the default configuration for RA8 devices, the FSP enables the Cortex®-M85 Instruction Cache (I-Cache). The Cortex®-M85 Data Cache (D-Cache) is provided as an optional feature within the BSP configuration and is disabled by default.

To enable the D-Cache, open FSP Configurator by double click to **configuration.xml** and navigate to the **BSP** tab. Under **RA8P1 Family > Cache Settings**, enable the Data Cache option. Figure 39 shows an example of enabling the Data Cache in the CM85 project.

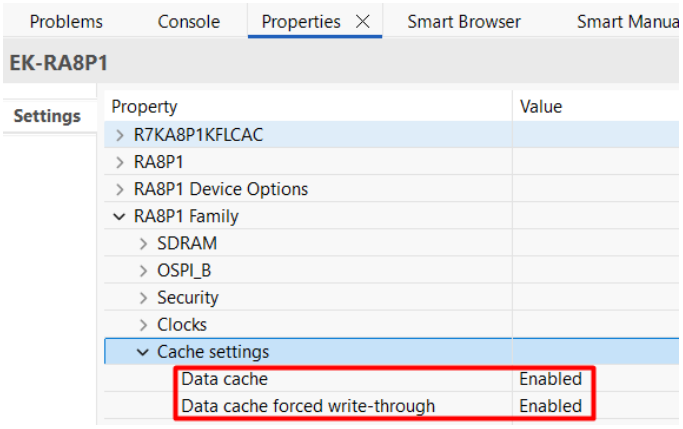

Figure 39. Enable Cache in CM85 Project.

References

Renesas RA FSP User's Manual

Renesas RA Family RA8P1 User's Manual: Hardware (R01UH1064)

Renesas RUHMI Framework Quick Start Guide (R11QS0065)

Renesas RUHMI Framework GitHub

Using Ethos-U NPU with RA MCUs (R01AN7712)

Guidelines for Using the S Cache on the System Bus (R11AN0538)

Developing with RA8 Dual Core MCU (R01AN7881)

Getting Started with RA8 Memory Architecture, Configurations and Topologies (R01AN7880)

Reference System Design for Vision AI on EK-RA8D1 (R11AN0756)

6. Website and Support

Visit the following URLs to learn about the RA family of microcontrollers, download tools and documentation, and get support.

EK-RA8P1 Resources	renesas.com/ra/ek-ra8p1
RA8P1 Resources	renesas.com/ra/ra8p1
RA Product Information	renesas.com/ra
Flexible Software Package (FSP)	renesas.com/ra/fsp
RA Product Support Forum	renesas.com/ra/forum
Renesas Support	renesas.com/support

Revision History

Rev.	Date	Description	
		Page	Summary
1.0.0	Jul.15.25	-	Initial release
1.1.0	Oct.08.25	-	Add EfficientNet Lite0 model support
1.2.0	Apr.29.26	-	Update to FSPv6.4.0

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan

www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.