Renesas RA Family

Reference System Design for Vision AI on EK-RA8D1

Introduction

The Renesas RA8 Series MCUs integrate Helium technology which boosts performance when running an Artificial Intelligence (AI) model. This document also describes other supporting hardware features and how they can be used in an AI application project. Porting a publicly available TensorFlow Lite (TFL) face detection model to RA8 MCU for inference operation is demonstrated. The design is architected to configure the system for best AI inference performance. The guidelines on porting a TFL AI model using open-source tools applies to all RA MCUs. The model size and the MCU memory footprint should be evaluated when considering porting an AI model to the RA MCUs.

The EK-RA8D1 is used for demonstrating running the face detection model. The OmniVision OV3640 camera, shipped with the EK-RA8D1 is used to collect the input image for inference. The Focus MIPI LCD shipped with the EK-RA8D1 is used to display the streamed image from the camera and provide information for the face detection result.

This release of the application project uses Renesas Flexible Software Package (FSP) v5.3.0 using LLVM Embedded Toolchain for Arm version 17.0.1 compiler.

Required Resources

Software and development tools

- e² studio IDE v2024-04
- Renesas Flexible Software Package (FSP) v5.3.0

The links to download the above software are available at https://github.com/renesas/fsp.

Hardware

- EK-RA8D1, Evaluation Kit for RA8D1 MCU Group (renesas.com/ra/ek-ra8d1)
- Workstation running Windows® 10 and the Tera Term console or similar application.
- One USB device cable (type-A male to micro-B male)

Prerequisites and Intended Audience

This application project assumes that the audience understands AI/ML design concepts and has good experience using 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 for guidance.

The intended audience is all users who are or will be developing AI applications using Renesas RA MCUs and the Tensor Flow Lite for Microcontrollers (TFLM) library using a pretrained and quantized TFL model. Model training and quantization are outside the scope of this application project.

Contents

1. RA8 MCU Device Features and IDE Support....4
1.1 CPU Features and IDE Supports....4
1.1.1 High-Performance Cortex-M85 Core for AI Application....4
1.1.2 Cortex-M85 Helium Technology....4
1.1.3 Using Helium Technology and CMSIS-NN Library for Quantized AI Models....4

1.1.4 Floating-Point Hardware Unit 5
1.1.5 Other CPU Features 6
1.2 Other Memory and Peripheral Features 7
1.2.1 User SRAM 7
1.2.2 Code Flash 8
1.2.3 External Bus Interface (SDRAM) 8
1.2.4 2D Drawing Engine (DRW) 8
1.2.5 Graphic LCD Controller (GLCDC) and MIPI DSI Interface 9
1.2.6 Capture Engine Unit (CEU) 9
1.2.7 Serial Communication Interface (SCI) 9
1.2.8 I2C Bus Interface (IIC) 9
1.2.9 General PWM Timer (GPT) 9

2. Deploy Open-Source AI Model using TFLM 9
2.1 Overview 9
2.2 Tensor Flow Lite for Microcontrollers 10

3. The Face Detection AI Application Architecture 10
3.1 System Architecture 10
3.1.1 Hardware Setup and FSP Driver Usage 10
3.1.2 Memory Allocation 12
3.1.3 Major Operations and Interactions 13
3.2 Yolo-Fastest Face Detection Model 14
3.3 OV3640 Camera Module Usage 17
3.3.1 Connect the OV3640 Camera Module with EK-RA8D1 17
3.3.2 OV3640 Color Format and Framerate Configuration 17
3.3.3 OV3640 Exposure Configuration 18
3.3.4 OV3640 Camera Zoom Control 18
3.3.5 OV3640 Output Size Configuration 19
3.4 Configure the Capture Engine Unit (CEU) 20
3.5 Process the Camera Image Data for AI Inference 20
3.6 Integrate and Execute the Face Detection Model 21
3.7 Post Inference Data Processing 21
3.8 Display the Camera Image on the MIPI LCD 21
3.9 Display the Face Detection Result on the MIPI LCD 22
3.10 Text Printing on the MIPI LCD 22

4. Running the Face Detection Project 23
4.1 Setting Up the Hardware and the IDE 23
4.2 Set up the e^2 studio and LLVM Embedded Toolchain 24
4.3 Compile and Run the Application 25
4.4 Evaluate the System Performance 28

4.4.1 Data Watchpoint and Trace (DWT) for Time Recording 28
4.4.2 J-Link Virtual Console as User Interface 28
4.4.3 Evaluating the Helium Usage 29

5. References 32

6. Website and Support 33

Revision History 34

1. RA8 MCU Device Features and IDE Support

This section focuses on some of the MCU features on Renesas RA8 MCU that can be used in a typical Vision AI application and how these features contribute to improved inference speed and an enhanced user visual experience. 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 this setup. Tools reference information for other IDEs and compilers is not included in this document.

1.1 CPU Features and IDE Supports

1.1.1 High-Performance Cortex-M85 Core for AI Application

The Renesas RA8 MCU Series integrates the Arm Cortex-M85 core, which implements the Armv8.1-M architecture profile and operates at a maximum frequency of 480MHz. Here is a summary of the core features that may enhance an AI application:

- Floating Point Unit (FPU), compliant with the ANSI/IEEE Std 754-2008 standard
- Helium technology M-profile Vector Extension (MVE)
- Armv8.1-M Security Extension
    - Armv8-M TrustZone Technology, Privileged control, and other security features. These features can be deployed for AI applications that require enhanced security.
    - This face detection application example does not utilize any of the security features on RA8D1. The entire application is developed with the CPU, the peripherals, and the on-chip memory regions in a secure state. The external memory SDRAM region, which is also used in this application, is defined as non-secure on the RA8D1 and is accessible by the CPU and peripherals.

1.1.2 Cortex-M85 Helium Technology

Cortex-M85 integrates Arm Helium technology to achieve the highest scalar, digital signal processing (DSP), and machine learning (ML) performance of the Cortex-M series of processors. Arm Helium technology is the M-Profile Vector Extension (MVE) for the Arm Cortex-M processor series. Helium is an extension of the Armv8.1-M architecture and delivers a significant performance boost for ML and DSP applications. The MVE supports Single Instruction Multiple Data (SIMD) operation, allowing the processing of 128-bit registers. The MVE-supported data types include:

• Integer
• Half-precision floating-point
• Single-precision floating-point

When using LLVM for ARM with RA Cortex-m85 MCU, using device optimization -O2 will enable Helium and Auto-vectorization. Refer to https://developer.arm.com/documentation/102095/0101/Auto-vectorization-and-Helium to learn more about Helium and Auto-vectorization.

In this example, the face detection application Helium is used for its integer type support to boost system performance on AI inference. Optimization Level -O2 is used in the project.

1.1.3 Using Helium Technology and CMSIS-NN Library for Quantized AI Models

The CMSIS-NN software library is a collection of efficient neural network kernels developed to maximize performance and minimize the memory footprint of neural networks on Arm Cortex-M processors. The CMSIS-NN library follows the int8 and int16 quantization specifications of TFLM. When using TFLM with quantized models, it is recommended that the CMSIS-NN library be used to perform neural network operations efficiently.

FSP provides a CMSIS-NN module that can be integrated into the application project.

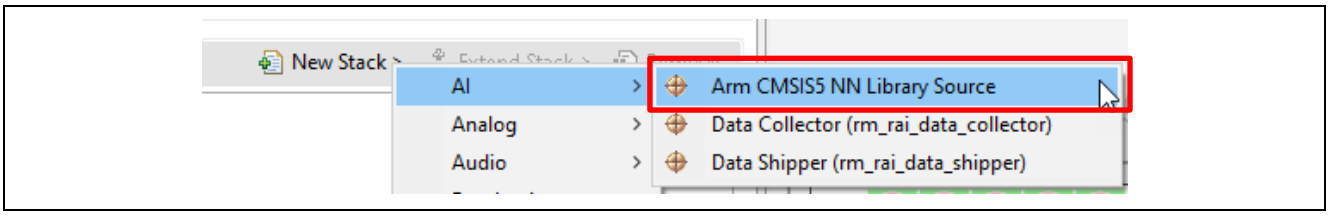


Figure 1. Use the Arm CMSIS NN Library

For Renesas RA Cortex-M85 MCUs, which support Arm M-Profile Vector Extension (MVE) Instructions, the "CMSIS_NN" preprocessor should be configured in the Compiler CPP option page in the IDE to enable the MVE instructions supported through the CMSIS-NN library, which is utilized by the TFLM library.

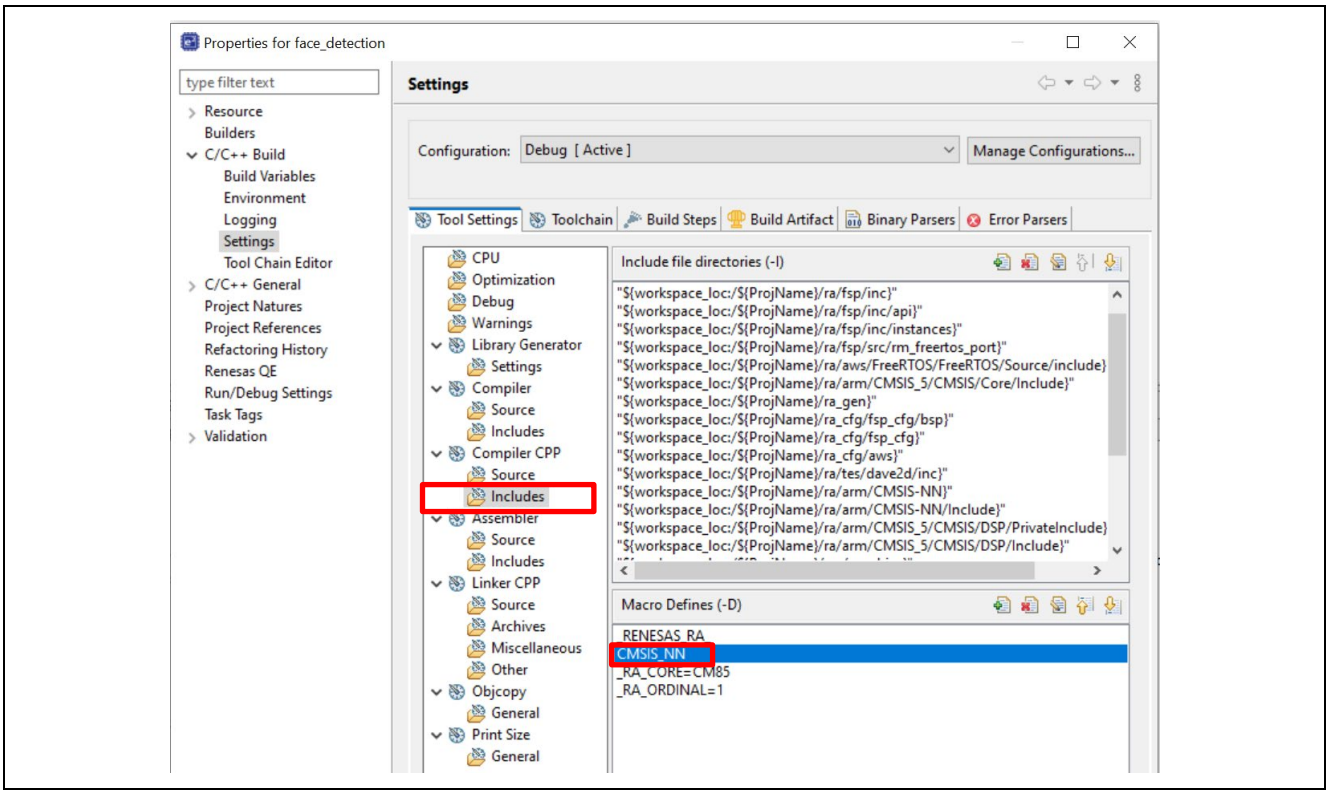


Figure 2. Enable TFLM Usage with Helium Support under Compiler setting

1.1.4 Floating-Point Hardware Unit

The FPU unit on RA8 is compliant with the ANSI/IEEE Std 754-2008 standard. The supported data types are scalar, half, single, and double-precision floating-point operations. The FPU hardware support is helpful if the AI application uses a float model. Although a float model requires more space and runs slower, it may achieve better model accuracy, which is desired in some AI applications. Although CMSIS-NN does not support float models, TFLM does support float models and can be used together with the FPU.

In this example face detection model, the 32-bit float model will not fit within the RA8D1 code flash, so an int8 quantized model is deployed. The FPU can be enabled through the IDE Property configuration. For RA8 MCUs, this option is enabled by default.

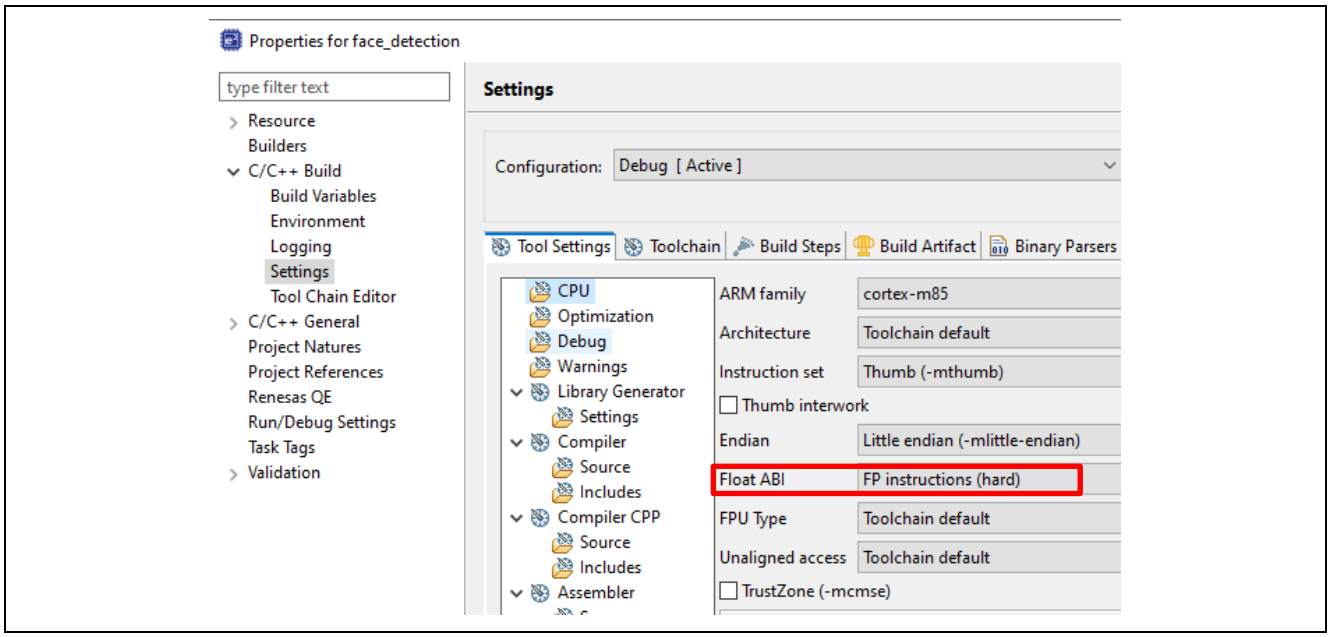


Figure 3. Enable Hardware Floating Point Unit

1.1.5 Other CPU Features

The RA8 MCUs possess other CPU features that can enhance the AI system performance, these include the Instruction Cache (I Cache) and Data Cache (D Cache), along with the Tightly Coupled Memory (Instruction TCM and Data TCM). This section focuses on the features used by this application project.

1.1.5.1 Instruction Cache and Data Cache

The Renesas RA8 MCUs implement16KB of Instruction Cache and 16 KB of Data Cache each with Error Correction Code (ECC) checking. With a cache hit, the access time is zero wait states. Since typical AI application neural network weights and sensor data for inference have local temporal locality, enabling I Cache, and D Cache greatly improves the AI application performance.

The Instruction Cache is enabled by the FSP Board Support Package (BSP) system_init function (\ra\fsp\src\bsp\cmsis\Device\RENESAS\Source\system.c). Additionally, branch prediction and branch cache are also enabled.

```
/***********************************************************************************************************************//**
 * Initialize the MCU and the runtime environment.
 **********************************************************************************************************************/
BSP_SECTION_FLASH_GAP void SystemInit (void)
{
#if defined(RENESAS_CORTEX_M85)

    /* Enable the instruction cache, branch prediction, and the branch cache (required for Low Overhead Branch (LOB) extension).
     * See sections 6.5, 6.6, and 6.7 in the Arm Cortex-M85 Processor Technical Reference Manual (Document ID: 101924_0002_05_en, Issue: 05)
     * See section D1.2.9 in the Armv8-M Architecture Reference Manual (Document number: DDI0553B.w, Document version: ID07072023) */
    SCB->CCR = (uint32_t) CCR_CACHE_ENABLE;
    __DSB();
    __ISB();
#endif
```

Figure 4. Enable CPU Instruction Cache, Branch Prediction and Branch Cache

The FSP BSP provides a configuration option to enable the Data Cache. In this face detection application, using the data cache significantly speeds up the image processing routines.

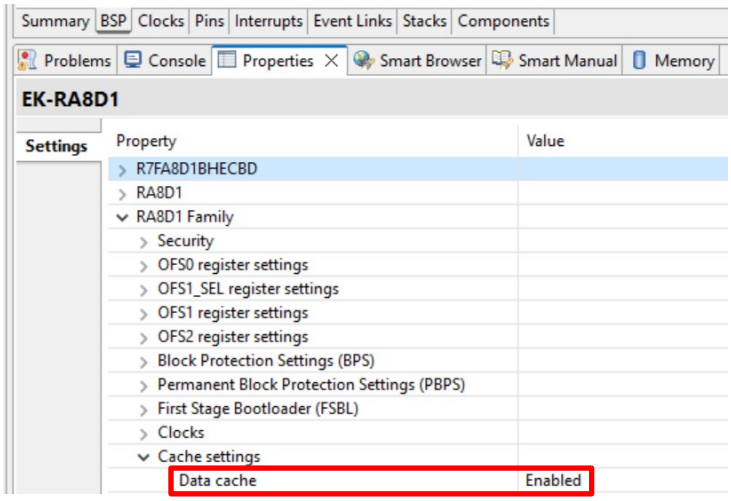


Figure 5. Enable CPU Data Cache

The automatic invalidation of the cache is enabled on RA8D1. This functionality will invalidate the D cache when it goes from an unpowered to a powered state. For a system where multiple bus masters write to a shared memory region, cache invalidation is needed prior to the CPU accessing the shared memory area. This can be achieved using the CMSIS API. Refer to the CMSIS library for details on the usage of the Cache APIs.

It is important to keep the Data Cache coherency during normal MCU execution. The Data Cache is enabled in this application project. The camera image storage area is written to by the CEU and is also accessed by the CPU for further processing, prior to serving the AI and MIPI LCD usage. This shared area is invalidated prior to the CPU accesses to keep the coherency of the Cache data using the CMSIS API SCB_InvalidateDCache_by_Addr.

1.1.5.2 Tightly Coupled Memory

The RA8 MCU integrates 128 KB of TCM RAM with 64 KB of Instruction TCM with ECC and 64 KB of Data TCM with ECC. The TCM RAM, being part of the CPU subsystem, provides 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, this can be used to improve access speed and determinism. In this example of a face detection system, the interrupt callbacks, which are defined by the application code, which are defined by the application code are allocated to the ITCM. As the camera image data and the intermediate buffers that store the processed images are much larger than the DTCM, the DTCM is not used in this example.

1.2 Other Memory and Peripheral 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 RA8D1. 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 example face detection application.

1.2.1 User SRAM

The RA8 MCU provides an on-chip, high-density user SRAM module with either parity-bit checking or ECC SRAM 0 is 384 KB with ECC, and SRAM 1 is 512 KB with parity check.

With this face detection application, the MCU is operated with ICLK at 240 MHz, and a one-cycle wait state is used. If the ICLK frequency is 120 MHz or less, a wait state is not required. This face detection application example uses about 90% of this on-chip SRAM.

1.2.2 Code Flash

The code flash memory stores instructions and operands. The RA8D1 MCU has up to 2MB of code flash memory, depending on the part number of the device. The EK-RA8D1 board features 2MB of code flash memory. The Flash Cache (FCACHE) speeds up read access from the bus master to the flash memory. On a cache hit, the access cycle is in a 0-wait state. On a cache miss, the wait cycle is the same as when the Flash Cache is disabled. The Flash Cache includes the following three caches. Refer to the RA8D1 Hardware User's Manual section "Feature of flash cache" for details on the definition and specification of these caches.

- FCACHE1, for CPU instruction fetches.
- FCACHE2, for CPU operand access and access from EDMAC.
- FLPF, for the prefetch access in CPU instruction fetches.

By default, FSP BSP enables Flash Cache, which is critical to enable the AI inference processing. The face detection application uses less than half of the code flash available on the EK- RA8D1.

```
/*******************************************************************************************************************//**
 * Initializes system clocks.  Makes no assumptions about current register settings.
 **********************************************************************************************************************/
void bsp_clock_init (void)
{
    /* Unlock CGC and LPM protection registers. */
#if BSP_FEATURE_TZ_VERSION == 2 && BSP_TZ_NONSECURE_BUILD == 1
    R_SYSTEM->PRCR_NS = (uint16_t) BSP_PRV_PRCR_UNLOCK;
#else
    R_SYSTEM->PRCR = (uint16_t) BSP_PRV_PRCR_UNLOCK;
#endif

#if BSP_FEATURE_BSP_FLASH_CACHE
 #if !BSP_CFG_USE_LOW_VOLTAGE_MODE && BSP_FEATURE_BSP_FLASH_CACHE_DISABLE_OPM

    /* Disable flash cache before modifying MEMWAIT, SOPCCR, or OPCCR. */
    R_BSP_FlashCacheDisable();
 #else

    /* Enable the flash cache and don't disable it while running from flash. On these MCUs, the flash cache does not
     * need to be disabled when adjusting the operating power mode. */
    R_BSP_FlashCacheEnable();
 #endif
#endif
```

Figure 6. Enable the Flash Cache

1.2.3 External Bus Interface (SDRAM)

The RA8D1 has external address space, which is divided into 8 CS areas, an SDRAM area (SDCS), and OSPI areas (CS0 and CS1). On the EK-RA8D1, an SDRAM with a 16-bit data bus is included, but an SDRAM with a 32-bit data bus can be preferable for higher performance throughput.

The on chip 986 KB of SRAM is not enough to support the entire face detection SRAM usage, so the GLCDC buffers and most of the camera images are stored in the SDRAM. The downside is the performance hit as accessing the SDRAM access is much slower than accessing the on-chip SRAM.

The EK-RA8D1 SDRAM is supported by an FSP BSP initialization function, bsp_sdram_init. Located in \ra\board\ra8d1_ek\board_sdram.c.

1.2.4 2D Drawing Engine (DRW)

The RA8D1 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, the DRW is used to copy a rectangular part of the source into the destination framebuffer, which will then be displayed on the LCD. Additionally, the DRW is used to render the face detection result by drawing a bounding box around the face detected. FSP driver r_drw can be used to access the DRW engine from the application.

1.2.5 Graphic LCD Controller (GLCDC) and MIPI DSI Interface

The GLCDC on the RA8D1 MCUs supports a single-color background and two graphics planes. A variety of pixel formats are supported. In this application, one graphic plane is used with the RGB565 format. The GLCDC peripheral works with several sub-peripherals to form a pixel data processing pipeline.

The MIPI PHY on the RA8D1 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 parallel interface is used, the GLCDC can be used without adding the MIPI stack.

In this application, the input to the GLCDC is RGB565, and the output to the MIPI DSI is RGB888. The MIPI DSI is operated in 2-lane mode to stream the camera image and rending the inference result by drawing bounding boxes around the detected faces.

1.2.6 Capture Engine Unit (CEU)

The CEU peripheral supports interfacing with external cameras by accepting timing and data signals to capture incoming data. A callback is invoked for each V-Sync event, frame of data accepted, or when certain errors occur. The CEU peripheral supports interfacing with external cameras using an 8 or 16-bit data bus with uncompressed image data (e.g., RGB565 or YUV422) or compressed image data (e.g., JPEG) with images of size up to 5 megapixels. The hardware interface to the external camera includes vertical sync, horizontal sync, pixel clock, and the data line.

In this application, the CEU is used to interface with the OV3640 camera module using an 8-bit data bus and capture RGB565 QVGA images (320x240).

1.2.7 Serial Communication Interface (SCI)

The 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 face detection application, Channel 9 is configured as UART to communicate with the Jlink OB USB Debug for status display on the Windows terminal (Tera Term).

1.2.8 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 face detection application, channel 1 is utilized in master mode to communicate with the I2C slave interface on the OV3640 control block.

1.2.9 General PWM Timer (GPT)

The GPT timer is a 32-bit timer with GPT32x8 and a 16-bit timer with GPT16x6 channels on the RA8D1. 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 face detection application, channel 3 is used to generate the input clock for the OV3640 using one of the output pins.

2. Deploy Open-Source AI Model using TFLM

This section provides an overview of the major steps to deploy an open-source AI model. Brief introduction to the Tensor Flow Lite for Microcontrollers (TFLM) is also provided.

2.1 Overview

The following is the typical operational flow for AI Application Design with an already trained Tensor Flow Lite for Micro model:

1. **Convert an open-source model for a model**:

1) Generate a small TensorFlow model that can fit your target device and contains supported operations (https://www.tensorflow.org/lite/microcontrollers/build_convert#operation_support).
    - The model should be optimized for size and accuracy.
2) Convert to a TensorFlow Lite model using the TensorFlow Lite converter (https://www.tensorflow.org/lite/microcontrollers/build_convert).

Embedded system has a small footprint, and the quantized model is preferred in most use cases. To understand more about the quantization of the TFLite model, refer to the following link. https://www.tensorflow.org/lite/performance/post_training_integer_quant.

3) Deployment to the embedded system

This is achieved by converting the Tensor Flow Lite Micro model to a C byte array using standard tools (https://www.tensorflow.org/lite/microcontrollers/build_convert#convert_to_a_c_array) to store it in read-only program memory on the device. Reference the next section for an example of using the xxd utility to convert the Face Detection model to a C-byte array.

- If you have Cygwin (https://www.cygwin.com/install.html) or MinGW (https://sourceforge.net/projects/mingw/) installed, they come with the xxd utility.
- Otherwise, Vim (https://www.vim.org/download.php) can be a good option to acquire the xxd utility.

2. **Run inference** on the device using the Tensor Flow Lite for Micro library C++ library (https://www.tensorflow.org/lite/microcontrollers/library).
3. Process the results. The inference performance can be optimized by the hardware architecture and the usage of CMSIS-NN.

2.2 Tensor Flow Lite for Microcontrollers

TensorFlow Lite for Microcontrollers is designed to run machine learning models on microcontrollers. It is written in C++ 17 and requires a 32-bit platform.

To create an application using TensorFlow Lite for Micro library in e^2 studio, create a C++ project with the intended compiler. This will enable the C++ compiler to be supported by the IDEs. This application project can be used as a reference design to use the Tensor Flow Lite for Micro library.

Refer to the URL below for the details of TFLM.
• Tensorflow Lite for Microcontrollers https://www.tensorflow.org/lite/microcontrollers

3. The Face Detection AI Application Architecture

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

- Decouple the AI inference operation path and MIPI display operation path.
- Utilize the Core and CPU features (MVE instructions, Data Cache, etc.) to optimize the system performance.
- Utilize SRAM access performance advantage over SDRAM.
- Configure the Graphics system for optimal camera data capture and camera image streaming effect.

3.1 System Architecture

This section introduces the hardware, software, and memory usage of the face detection application.

3.1.1 Hardware Setup and FSP Driver Usage

This application uses the hardware system described in Figure 7.

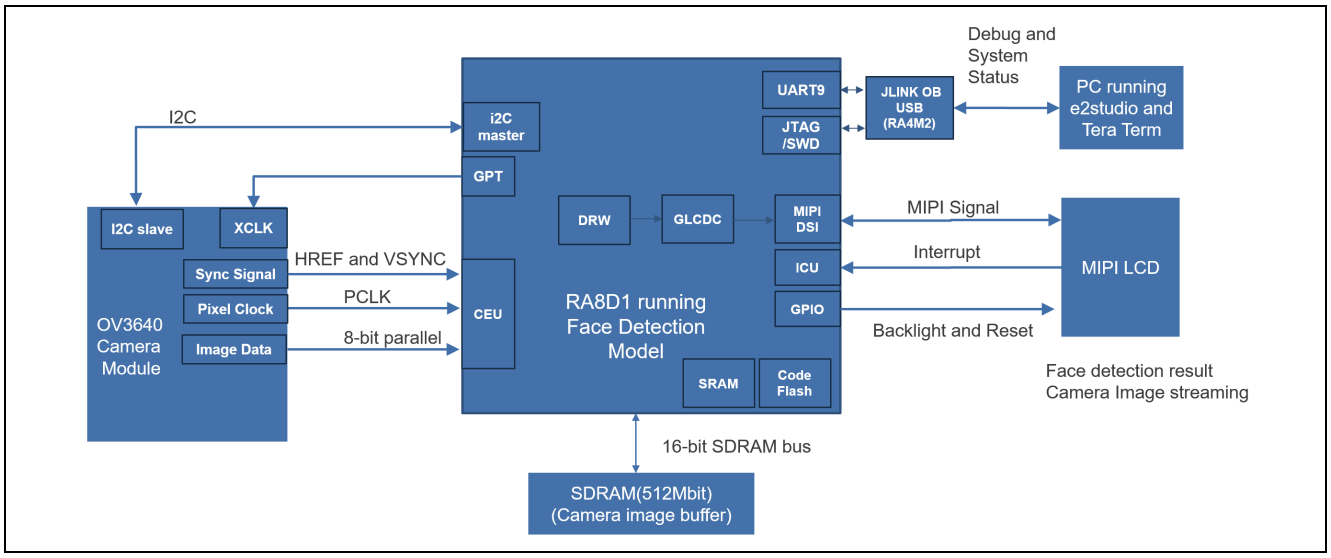


Figure 7. Hardware System Block Diagram

The following FSP v5.3.0 drivers are used in the software development of the application.

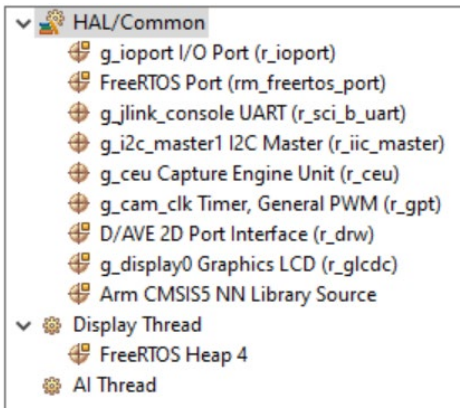


Figure 8. FSP Drivers Used

The following screen capture includes the versions of the third-party libraries that are integrated with FSP v5.3.0. Later, FSP versions may use different versions of the third-party libraries. This application project needs to be reevaluated when migrating to a new FSP version.

Board Support Package Common Files	v5.3.0
I/O Port	v5.3.0
FreeRTOS	v10.6.1+fsp.5.3.0
Arm CMSIS Version 5 - Core (M)	v5.9.0+renesas.1.fsp.5.3.0
FreeRTOS Port	v5.3.0
Board support package for R7FA8D1BHECBD	v5.3.0
Board support package for RA8D1	v5.3.0
Board support package for RA8D1 - FSP Data	v5.3.0
Board support package for RA8D1 - Events	v5.3.0
RA8D1-EK Board Support Files	v5.3.0
Capture Engine Unit (r_ceu)	v5.3.0
TES D/AVE 2D Port	v5.3.0
Graphics LCD Controller	v5.3.0
General PWM Timer	v5.3.0
I2C Master Interface	v5.3.0
MIPI DSI Host	v5.3.0
MIPI PHY Host	v5.3.0
SCI UART	v5.3.0
FreeRTOS - Memory Management - Heap 4	v10.6.1+fsp.5.3.0
Arm DSP Library Source	v5.9.0+renesas.1.fsp.5.3.0
Arm NN Library Source	v4.1.0+fsp.5.3.0
TES DAVE 2D Drawing Engine	v3.8.0+fsp.5.3.0

Figure 9. Third-Party Library Versions for FSP v5.3.0

3.1.2 Memory Allocation

The following tables detail the memory usage of the system when using FSP v5.3.0. The BSP, Thread Stack, and Heap are allocated in the e^2 studio Stack view, while other usages are allocated by the compiler based on the application code.

The RA8D1 on EK-RA8D1 features a 2MB code flash. The face detection application utilizes about 36% of the code flash.

Table 1 Code Flash Memory Usage

Category	Description	Size in KB
AI Model	**const** uint8_t nn_model[]	500
Application code	Application code	228

The RA8D1 has 896 KB of on-chip SRAM. The face detection application utilizes approximately 83% of the SRAM. The following table provides details on the major SRAM usages.

Table 2 SRAM Memory Usage

Category	Description		Size in KB
Stack and Heap	BSP Main Stack		4
	BSP Heap		16
	Display Thread Stack		4
	AI Thread Stack		8
	Thread Heap		16
AI specific usage	Tensor Arena	uint8_t tensorArena [ ]	512
	Model input buffer	model_buffer_int8[ ]	36
MIPI Display usage	Rotated CEU image	camera_out_rot_buffer565 [ ]	150

Due to insufficient SRAM for the GLCDC buffers and camera image buffers, SDRAM is utilized to store the following buffers.

Table 3 SDRAM Memory Usage

Category	Description	Code Reference Symbol	Size in KB
SDRAM	Camera Output Storage (double buffer)	camera_out_buffer[2][ ]	300
	GLCDC framebuffer	fb_background[2][ ]	800

3.1.3 Major Operations and Interactions

The following diagram depicts the major operations in this application. For details on the specific blocks, please refer to the corresponding sections.

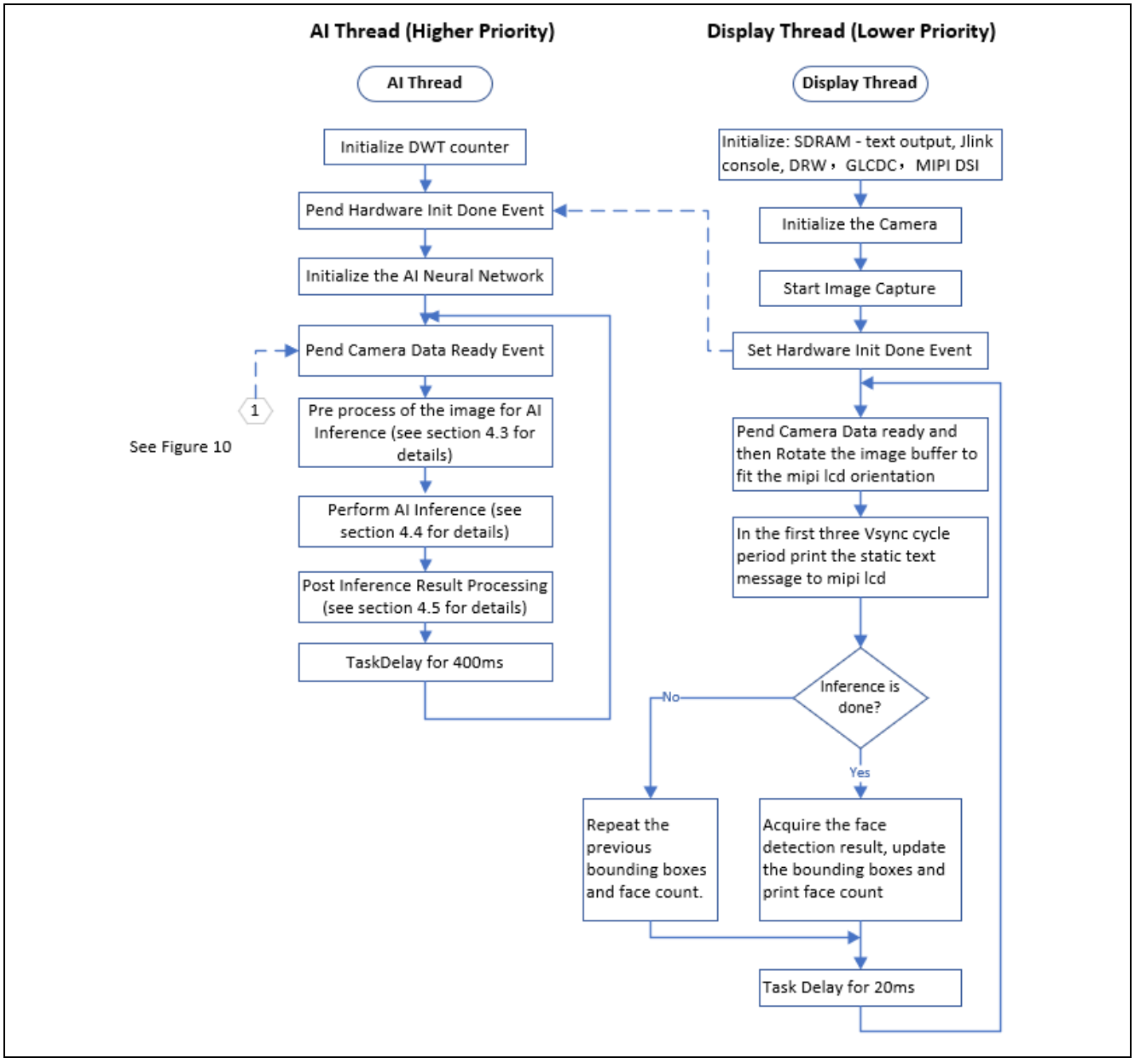


Figure 10. Operational Flow

There are several asynchronous interrupt service routine callbacks in the system, which are described in the following graph.

- CEU ISR Callback: An interrupt is triggered when a complete camera image frame is captured. The system switches to the other buffer in a double-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: An interrupt is triggered when a MIPI command transfer finishes. The system signals the display thread to send another MIPI command. This sequence repeats until the end of the MIPI command array.

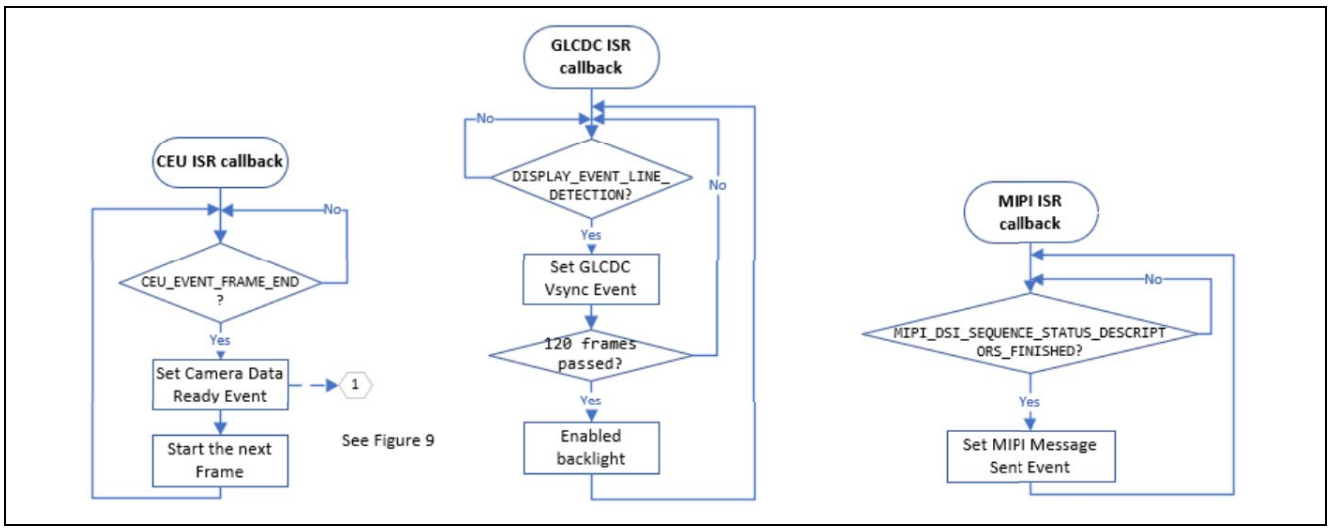


Figure 11. Interrupt Service Routine Callbacks

3.2 Yolo-Fastest Face Detection Model

This face detection project uses Yolo-Fastest for face detection. Yolo-Fastest is based on YoLo's ultra-lightweight CNN-based universal target detection algorithm. https://github.com/dog-qiuqiu/Yolo-Fastest.

There are several versions of this algorithm; the model used in this example project is custom-trained by Emza Visual Sense (https://github.com/emza-vs) based on the Yolo-Fastest model version v1.1.0 (http://doi.org/10.5281/zenodo.5131532) using private and public datasets (including http://shuoyang1213.me/WIDERFACE/). The training process of this model is outside the scope of this write-up. The guidelines for training the original Yolo-Fastest v1.1.0 model can be found at the first link provided in this section. There are many web pages that describe how to do custom training based on YoLo-Fastest, which can be easily located through a Google search.

This model is under Apache-2.0 license and can be downloaded from ModelZoo/object_detection (https://github.com/emza-vs/ModelZoo/tree/v1.0/object_detection). The model is already quantized to int8, which can fit in the RA8D1 MCU.

For the RA8D1 MCU, choose to download the **yolo_fastest_192_face_v4.tflite**.

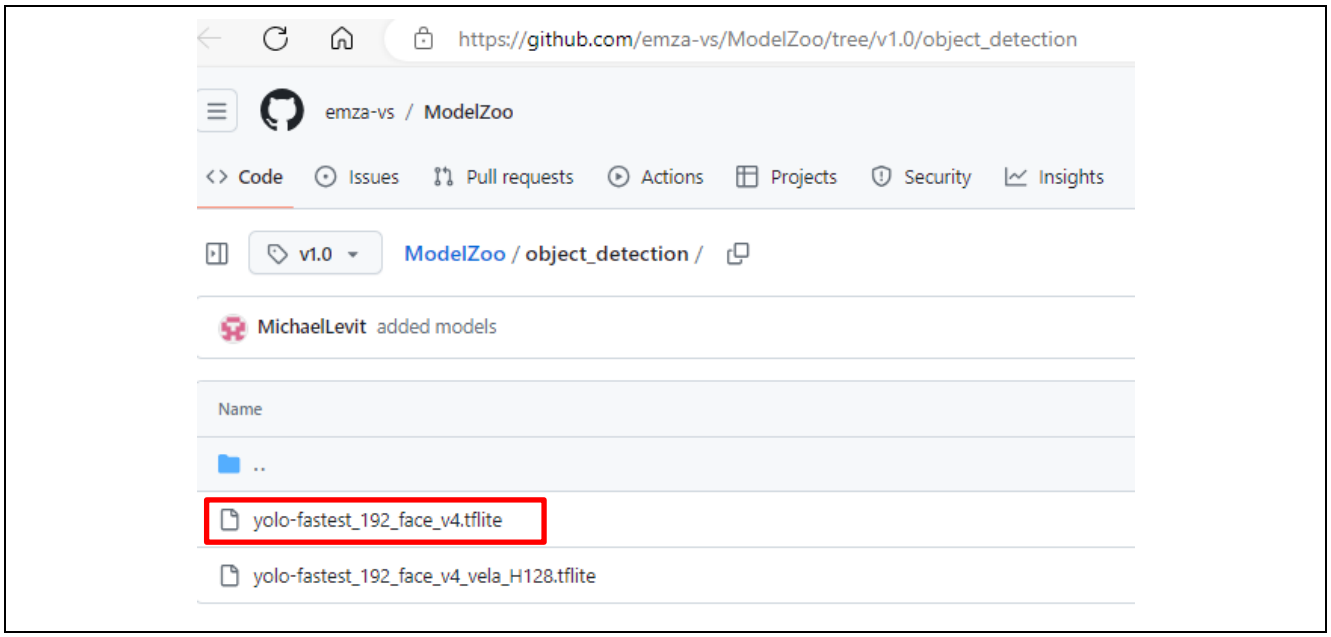


Figure 12. The Face Detection Tensor Flow Lite for Micro Model

This model has over 100 neural network layers, including many Conv2D and DepthwiseConv2D as well as ReLU layers. The NETRON tool (https://netron.app/) can be used to visualize the included layers and the input/output of each layer. Figure 13 is a snippet of the layers starting from the AI inference input layer (192x192x1). Using the NETRON tool is very easy; we only need to provide the .tflite file as input for this tool. Explaining the functionality of each layer is outside the scope of this application project. There is an abundance of resources online that explain the different operations in these layers. The keywords to these layers (e.g., Conv2D, DepthwiseConv2D can bring up lists of resources in a web search).

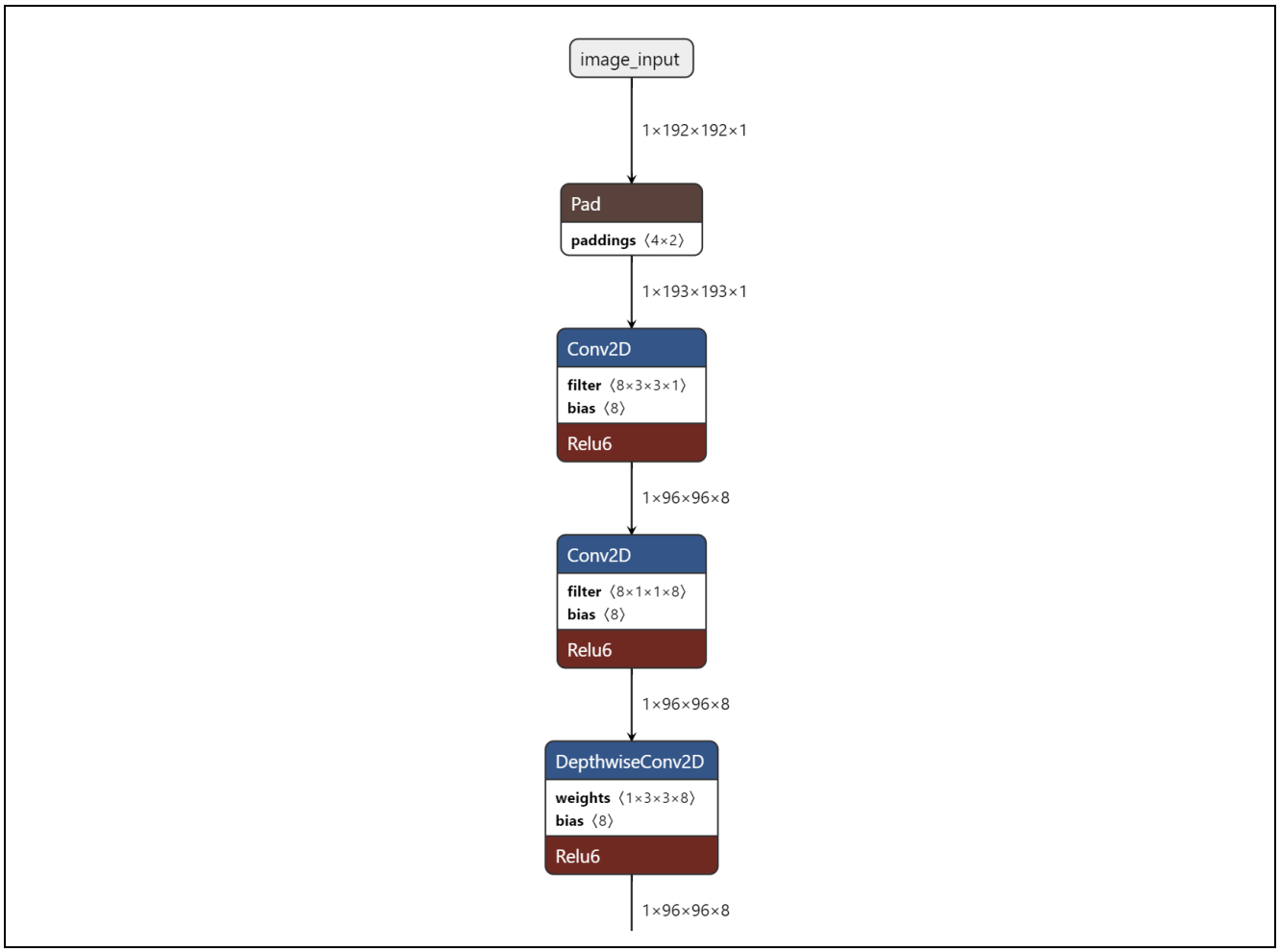


Figure 13. The Face Detection Neural Network Layers (snippet)

In this example project, the xxd utility included with the Vim system is used to generate the model data. A previously generated buffer is already included in the example project. It is not necessary to perform this step to evaluate the face detection application, but a new model buffer generated using the following steps can be used by replacing the provided model. Note that the model buffer generation process incorporates some randomness as part of the process; the result will not be identical to the included model data.

- Navigate to the Vim installation folder where the xxd utility is installed and open a command prompt to change the current directory to this folder.
- Copy the downloaded .tflite file to this folder.
- Execute the following script to generate the model as a flat buffer.

  ```
  xxd -i yolo-fastest_192_face_v4.tflite > c:\AI\model_data.cc
  ```

The result of this command is an array that holds the model data and a variable that holds the length of this array.

```
unsigned char yolo_fastest_192_face_v4_tflite[] = {
  0x1c, 0x00, 0x00, 0x00, 0x54, 0x46, 0x4c, 0x33, 0x14, 0x00, 0x20, 0x00,
  0x1c, 0x00, 0x18, 0x00, 0x14, 0x00, 0x10, 0x00, 0x0c, 0x00, 0x00, 0x00,
  0x08, 0x00, 0x04, 0x00, 0x14, 0x00, 0x00, 0x00, 0x1c, 0x00, 0x00, 0x00,
unsigned int yolo_fastest_192_face_v4_tflite_len = 510280;
```

Figure 14. Tensor Flow Lite for Micro Face Detection Model Flat Buffer

- This buffer can be integrated into the application project and used by the TFLM library to perform inference. In this application project, this flat buffer is used in the buffer `static const uint8_t nn_model[],` which is located in \src\ai\face_detection\yolo-fastest_192_face_v4.tflite.cpp.

3.3 OV3640 Camera Module Usage

This application project utilizes the OV3640 camera included in the EK-RA8D1 kit to gather 8-bit camera data for the face detection application. On the EK-RA8D1 board, the image sensor input clock is provided by the GPT timer channel 3. This clock is used by the OV3640 to enable image sensing. The OV3640 outputs the pixel clock PCLK, which will be used by the CEU for image data collection.

3.3.1 Connect the OV3640 Camera Module with EK-RA8D1

Note that on the EK-RA8D1, the camera module pin numbers are reversed from those on the module. So, when connecting the camera module with EK-RA8D1, Pin 1 on the EK-RA8D1 socket must align with Pin 2 on the camera module. Refer to Figure 21 for the correct orientation of the camera module.

Table 4 Camera Module and MCU Interconnect

Camera Module and MCU Interconnect

Camera Module pin number	Camera Module pin name	MCU pin number and signal name	Comment
1	VCC	3.3 V	3.3 V
2	GND	GND	GND
3	SCL	P512 (SCL 1)	two-wire serial interface clock
4	SDATA	P511 (SDA 1)	Two-wire serial interface data
5	VSYNC	P710 (CAM_VD)	Active high, frame valid
6	HREF	P709 (CAM_HD)	Active high, line/data valid
7	PCLK	P708 (CAM_CLK)	Pixel Clock output from OV3640
8	XCLK	P403 (CAM_XCLK)	Master Clock input for OV3640
9	DOUT9	P703 (CAM_D7)	MSB of the sensor data
10	DOUT8	P702 (CAM_D6)	Bit 7 of the sensor data
11	DOUT7	P701 (CAM_D5)	Bit 6 of the sensor data
12	DOUT6	P700 (CAM_D4)	Bit 5 of the sensor data
13	DOUT5	P406 (CAM_D3)	Bit 4 of the sensor data
14	DOUT4	P405 (CAM_D2)	Bit 3 of the sensor data
15	DOUT3	P401 (CAM_D1)	Bit 2 of the sensor data
16	DOUT2	P400 (CAM_D0)	LSB of the sensor data
17	PWDN	P704 (CAM_D8)	Power down pin. Active Low.
18	NC	P705 (CAM_D9)	Not used
19	DOUT1	P706 (CAM_D10)	Not used
20	DOUT0	P415 (CAM_D12)	Not used

3.3.2 OV3640 Color Format and Framerate Configuration

The OV3640 has an active array size of 2048x1536. The supported output formats are YUV (422/420) and YCbCr422, RGB565/555/444, 8-bit compression data. This application project uses RGB565 output to match the GLCDC input data format. The output format of the OV3640 is configured through a set of registers via the I2C interface. For more details on the OV3640 specification and the descriptions on the control registers, please contact the manufacturer. Only a few groups of key register settings used are described in this section. Renesas is not responsible for any issue generated when referencing the register descriptions provided in this write-up.

Frame rate control of the OV3640 output

Based on the OV3640 data sheet, the maximum image transfer rates are the following:

- 15 fps for QXGA (2048x1536) and any size scaling down from QXGA
- 30 fps for XGA (1024x768) and any size scaling down from XGA

Given the maximum image transfer rate, the following factors can be configured for the OV3640 to operate at a slower framerate:

- The camera module's internal PLL setting is proportional to the frame rate. This is controlled by OV3640 register 0x3011.
- The frame rate is proportional to the input clock (XCLK) to the camera module.
- The speed at which the MCU triggers the OV3640 image sensing and the image storage bandwidth affects the frame rate. The evaluation results indicate that the current trigger scheme can achieve the maximum transfer rate of the OV3640. The image is stored in SDRAM, which has a bandwidth fast enough to enable the camera's maximum image transfer rate.

The OV3640 default configuration uses QXGA mode. For the face detection application, QXGA mode enables a broader view and multiple face detection with a reasonable framerate. This configuration is used in this application project. To use the OV3640 in XGA mode, please contact the manufacturer for support.

Table 5 OV3640 Color Format and Framerate Control

Register number	Description	Default setting (QXGA)	Face detection app setting (QVGA)
0x3012	Framerate Control Bit[6 :4] : Sensor array resolution	0x00 QXGA mode	0x00 QXGA mode
0x3400	Format Multiplexing Control Bit[2 :0] : ISP format selection	0x03 ISP RAW	0x01 ISP RGB
0x3404	Format Control Bit[5 :0] : Color format selection	0x18 Raw	0x11 RGB565

3.3.3 OV3640 Exposure Configuration

It is important to be aware that the OV3640 is configured to work in an office environment. The camera may not perform well when exposed to other lighting conditions.

The following settings are used in this application project. They are based on the Average-based Algorithm default exposure level setting. Please contact the manufacturer to understand these settings.

Table 6 OV3640 Exposure Configuration

Register number	Description	Default setting (QXGA)	Face detection app setting (QVGA)
0x3018	Bit[7 :0] : Luminance Signal/Histogram High Range for AEC/AGC operation	0x78	0x38
0x3019	Bit[7 :0] : Luminance Signal/Histogram Low Range for AEC/AGC operation	0x68	0x30
0x301a	Bit[7 :4] : High threshold Bit[3 :0] : Low threshold Fast Mode Large Step Range Thresholds	0xD4	0x61

3.3.4 OV3640 Camera Zoom Control

The OV3640 features an image scaling circuit. The image first enters the Windowing circuit, which decides the horizontal and vertical starting point and the length of the sensor array. The image within this selected window of the sensor array then enters the Image Signal Processing (ISP) scaling control, which decides the zoom-out size. By default, the OV3640 uses the QXGA window size as the scaling input and output VGA (640 x 480) image.

The face detection model requires a 192x192x1 image for inference; it would be sensible to use a camera output resolution that, it would be sensible to use a camera output resolution which is close to this resolution. In addition, considering the efficient scaling capability of the DRW, this application selected an output resolution of QVGA RGB565 (320x240) from the OV3640.

To have the best field of view from the camera, the zoom output should also be set to 320x240 RGB565 QVGA, as the final camera output needs to be equal or smaller than the scaling zoom output. To learn more about the OV3640 scaling circuitry, please contact the manufacturer for the reference document.

Figure 15. Camera Internal Processing (inside the dotted box)

Table 7 Output QVGA from the Scaling Circuit

Register number	Description	Default setting (QXGA)	Face detection app setting (QVGA)
0x3362	SIZE_OUT_MISC Bit [6 :4] : Vsize_Out[10 :8] (use with 0x3361[7 :0] to construct the Vertical input size) Hsize_Out[11 :8] (used with 0x3360[7 :0] to construct the Horizontal input size)	0x68	0x01
0x3363	HSIZE_OUT_L Bit[7 :0] : Hsize_Out[7 :0] (use with 0x0x3362[3 :0])	0x08	0x48
0x3364	VSIZE_OUT_L Bit[7 :0] : Vsize_Out[7 :0] (use with 0x3362[6 :4])	0x04	0xF4

3.3.5 OV3640 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 face detection application, the same size is used for both zooming out and the image output of the camera, which is QVGA.

Table 8 Configure the Camera Output

Register number	Description	Default setting (QXGA)	Face detection app setting (QVGA)
0x3088	Image Signal Processing (ISP) X-direction Output Size [15 :8] Bit [7 :4] : Not used Bit [3 :0] : X-direction Output size[11 :8]	0x08	0x01
0x3089	ISP X-direction Output Size [7 :0]	0x00	0x48*
0x308a	ISP Y-direction Output Size [15 :8] Bit [7 :3] : Not used Bit [2 :0] : Y-direction Output size [10 :8]	0x06	0x00
0x308b	ISP Y-direction Output Size [7 :0]	0x00	0xF0

Note *: The edge of the camera image is corrupted, so collect 8 more pixels and use CEU to crop this portion off.

3.4 Configure the Capture Engine Unit (CEU)

The CEU is configured through the FSP CEU stack. Its configuration should match the camera output (Table 8) in terms of the image format and size.

In this example project, the CEU is used to fetch RGB565 images in 256-byte units from the OV3640 and sequentially write the image data to the memory. An interrupt is triggered when a complete camera image frame is captured. The following are the key stack configurations for this example usage. For the rest of the CEU configuration, the FSP default settings are used. Users can refer to section Capture Engine Unit (r_ceu) in the FSP User's Manual to understand the other CEU configurations.

Table 9 Configure the CEU

Module Property Path and Identifier: **g_ceu Capture Engine Unit (r_ceu) > Settings > Property > Module g_ceu_vga Capture Engine Unit (r_ceu)**	**Default Value**	**Project Configuration**	**Comment**
> Input > Capture Specifications > Horizontal capture resolution	640	320	Match OV3640 output
> Input > Capture Specifications > Vertical capture resolution	480	240	Match OV3640 output
> Input > Capture Specifications > Horizontal pixel offset	0	8	This is to clip off some corruption on the edge of the camera image
> Output > Byte Swapping > Swap 32-bit units	☐	☑	The output from the OV3640 is big endian while the CEU is little endian.
> Output > Byte Swapping > Swap 16-bit units	☐	☑	16bpp image needs 16-bit swap as well
> Interrupts > Callback	g_ceu0_user_callback	g_ceu_user_callback	Restart capture upon reception of event: CEU_EVENT_FRAME_END

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 buffer is doubled buffered to allow parallel processing during image collection. These buffers are stored in the SDRAM.

3.5 Process the Camera Image Data for AI Inference

In this application project, the image collected from the CEU is in a resolution of 320x240 with RGB565 format. This image goes through the following processing before being fed to the AI model.

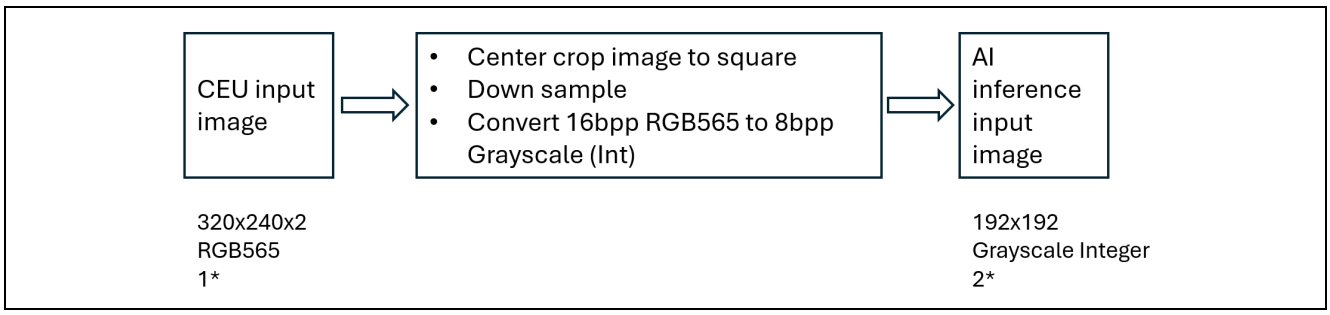


Figure 16. Process the Image Prior to AI Inference

Note 1* : camera_out_buffer[2][ ]

Note 2* : model_buffer_int8[]

The following routine is used for the image processing. Note that this routine is targeted to process 16bpp images to grayscale 8bpp integer images; it should be updated if other input and output formats are used.

```
/*****************************************************************************************************************
 *  Crop square, downsample and convert RGB565 to 8bpp grayscale (integer).
 *  @param[IN]   const void * inbuf: input 320x240 RGB565 image buffer
 *  @param[IN]   void * outbuf: output 192x192 int8 image buffer
 *  @param[IN]   uint16_t width: width of the input buffer
 *  @param[IN]   uint16_t height: height of the input buffer
 *  @param[IN]   uint16_t width: width of the output buffer
 *  @param[IN]   uint16_t height: height of the output buffer
 *****************************************************************************************************************/

face_det_err_t image_rgb565_to_int8 (const void * inbuf, void * outbuf, uint16_t in_width, uint16_t in_height, uint16_t out_width, uint16_t out_height)
```

Figure 17. Software API to process camera image data for inference

3.6 Integrate and Execute the Face Detection Model

As the face detection model's flat buffer and the tensor arena memory usage exceeds the SRAM size for the RA8D1 part used, they cannot both be put in the SRAM region. The tensor arena area is accessed more frequently and more sensitive to access latency, so it is put in the SRAM area. The face detection model is put in the code flash with reasonable access latency.

The face detection model is configured to detect a maximum of 20 faces, with a detection threshold of 0.5. Update the following macro definition to change the number of faces to be detected (\src\common_util.h)

/* This definition selects how many faces will be presented in the detection result */

#define MAX_DETECTION_NUMS 20

3.7 Post Inference Data Processing

The AI system maintains a list of inference results. The face detection inference uses a set of bounding boxes to indicate the inference result.

- After the inference is performed, the list of detections is sorted based on the detection result.
- Next, non-max suppression (NMS) is applied to removing redundant or overlapping bounding boxes.
- The top twenty bounding boxes with the highest probability are then updated to an array.
- The display thread subsequently accesses this array to render the location of the face on the MIPI display using the bounding boxes and to display the number of faces on the MIPI LCD.

3.8 Display the Camera Image on the MIPI LCD

The MIPI LCD configuration is initiated in the function display_mipi_focuslcd_init(). Once the MIPI commands are all executed, the MIPI transitions to video mode to stream the camera image buffer accessed via the DRW r_drw driver. The MIPI LCD has touch capability, but this functionality is not used in this application.

The OV3640 and CEU generate images in landscape format, whereas the MIPI LCD operates in portrait mode. Hence, before displaying the camera image on the MIPI LCD, it must be rotated clockwise 90 degrees. Rotation is a resource-intensive operation if both the source and destination buffer are stored in SDRAM. In this example implementation, the destination buffer is stored in SRAM, so the rotation is fast and only a single buffer is used to store the rotated image. Rotation can be either achieved using either software or the DMA unit on the RA8D1. For the image size of QVGA, software rotation is more efficient, hence it is used in this application project.

The image rotation is handled by the function **rotate_16bit_image_90_clockwise** (ceu_ctl_and_data_processing.c). The scaling from 320x240 to 640x480 is done using the scaling function of the DRW.

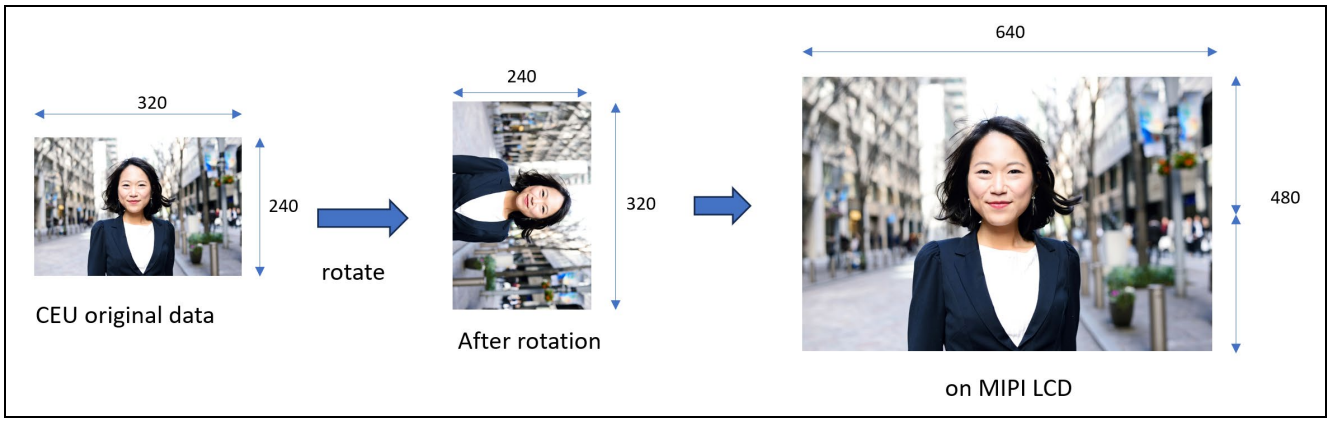


Figure 18. Display the Camera Image on the MIPI LCD

3.9 Display the Face Detection Result on the MIPI LCD

The display thread accesses the AI model inference result for rending the bounding boxes for the faces detected on the MIPI LCD.

The AI inference output has a different coordinate and scaling system than the MIPI LCD. It needs to be reprocessed to generate the coordinates of the top right corner and the bottom left corner of the bounding box using the MIPI LCD coordinate system. The DRW is then used to draw four lines to form the bounding box.

The following figure depicts the three steps to convert the AI inference result to the MIPI display coordinate. The notation matches the variable definitions in the source code. Once the coordinates are acquired, the DRW is used to draw the bounding box.

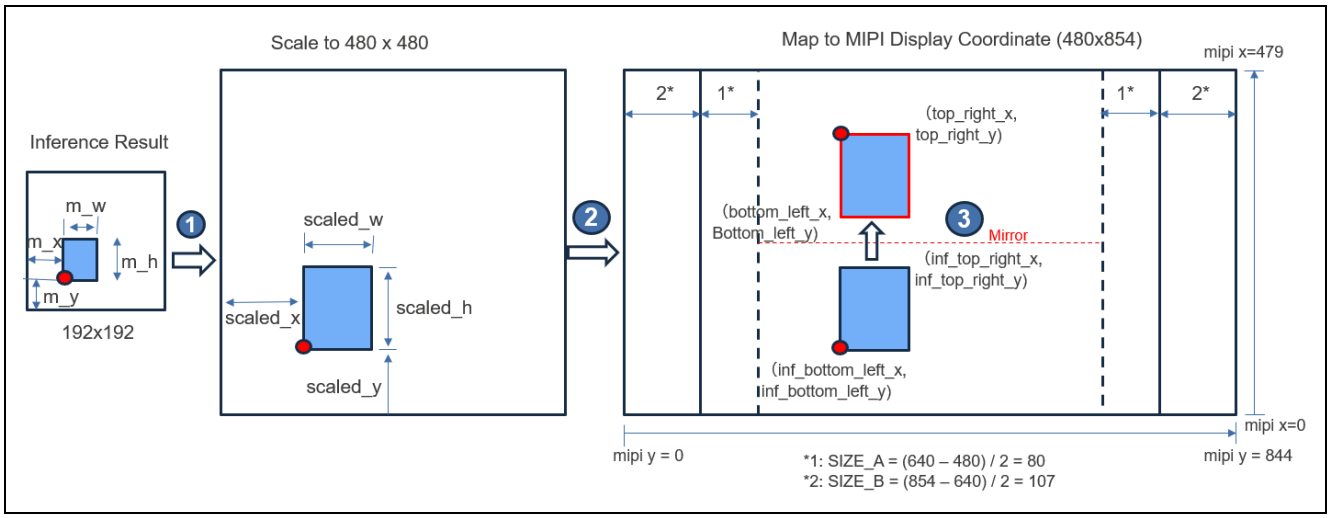


Figure 19. Derive the Top Right and Bottom Left Coordinate of the Bounding Box

Figure 19 can be used to understand the following routines in the example project. The literals are kept in the source project to match the dimensions depicted here.

3.10 Text Printing on the MIPI 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 following API prints a string to the MIPI 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 20. Print Text to MIPI LCD

4. Running the Face Detection Project

This section describes how to run and evaluate the face detection application.

4.1 Setting Up the Hardware and the IDE

The EK-RA8D1 package includes all the hardware components needed to run this face detection application.

- EK-RA8D1 board
- Micro USB device cable (type-A made for micro-B male). This is used to connect with the Debug USB and the PC
- Camera Expansion Board
- MIPI Graphics Expansion Board
- Nylon mounting stands and nylon screws. These are used to mount the MIPI LCD with the EK-RA8D1 board.

Prior to running this face detection application, default jumper, and switch settings must be utilized. Refer to the EK-RA8D1 user's manual for the default jumper and switch settings. Ensure SW1 and the camera module are setup as in Figure 21. Please pay special attention to the orientation of the camera module, ensuring correct orientation and correct connection on all the pins.

It is highly recommended to secure the MIPI Graphic Expansion board using the Nylon stands and screws if they are available.

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

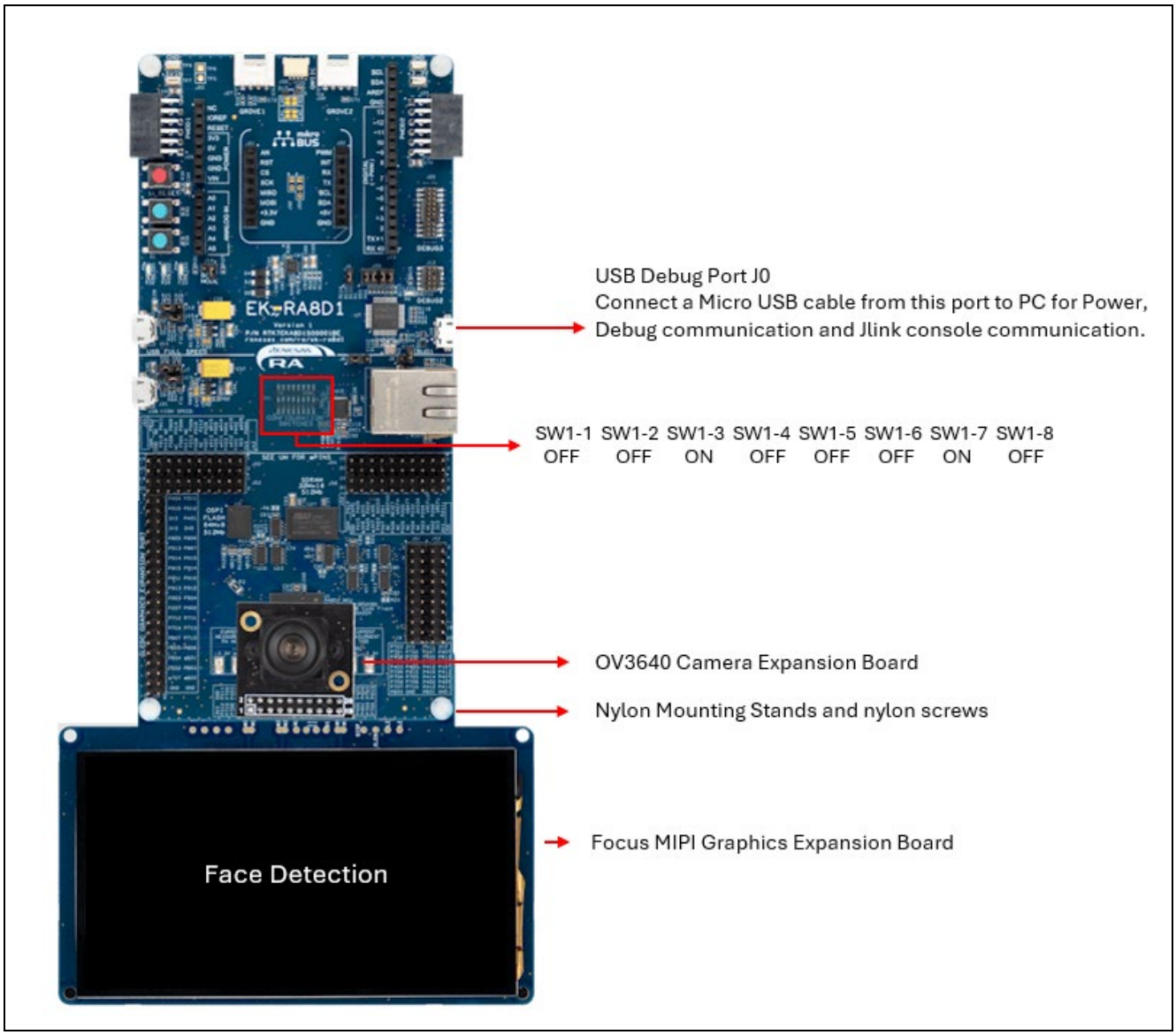


Figure 21. Assemble the Hardware System

4.2 Set up the e² studio and LLVM Embedded Toolchain

The included example project uses the LLVM embedded toolchain for Arm version 17.0.1. The LLVM toolchain can be downloaded from LLVM Embedded Toolchain for Arm: Release 17.0.1 (https://github.com/ARM-software/LLVM-embedded-toolchain-for-Arm/releases/tag/release-17.0.1).

Once the LLVM is downloaded, follow the below procedure to integrate LLVM intoto e² studio.

- Extract the downloaded zip file to a location on the PC.
- Launch e² studio.
- From the e² studio top menu items, click Help -> Add Renesas Toolchains -> Renesas Toolchain Management
- Select the entry called LLVM Embedded Toolchain for Arm and browse to the location where the LLVM embedded toolchain for Arm is unzipped.

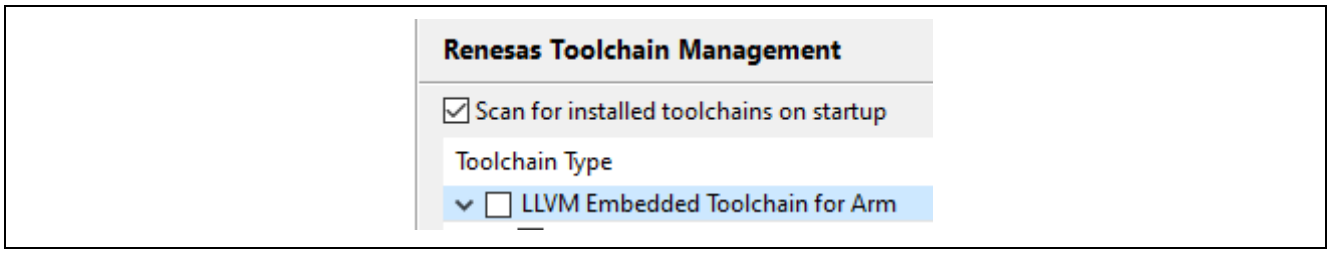


Figure 22. Integrate LLVM Embedded Toolchain for Arm to e² studio

- Now, the LLVM should be available for normal use within the e² studio.

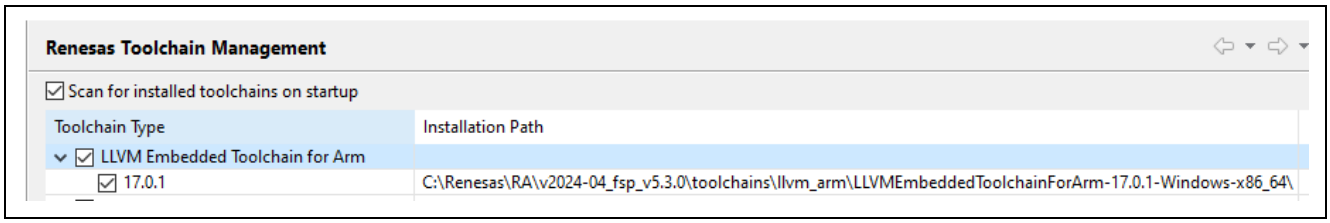


Figure 23. LLVM Embedded Toolchain for Arm is integrated into e² studio

When working with LLVM for C++ code, the exceptions are not handled. This is achieved by using the -fno-exceptions configuration. This configuration should be set up in the following three places on the project Properties page. Open the **Properties** page and select **Settings**.

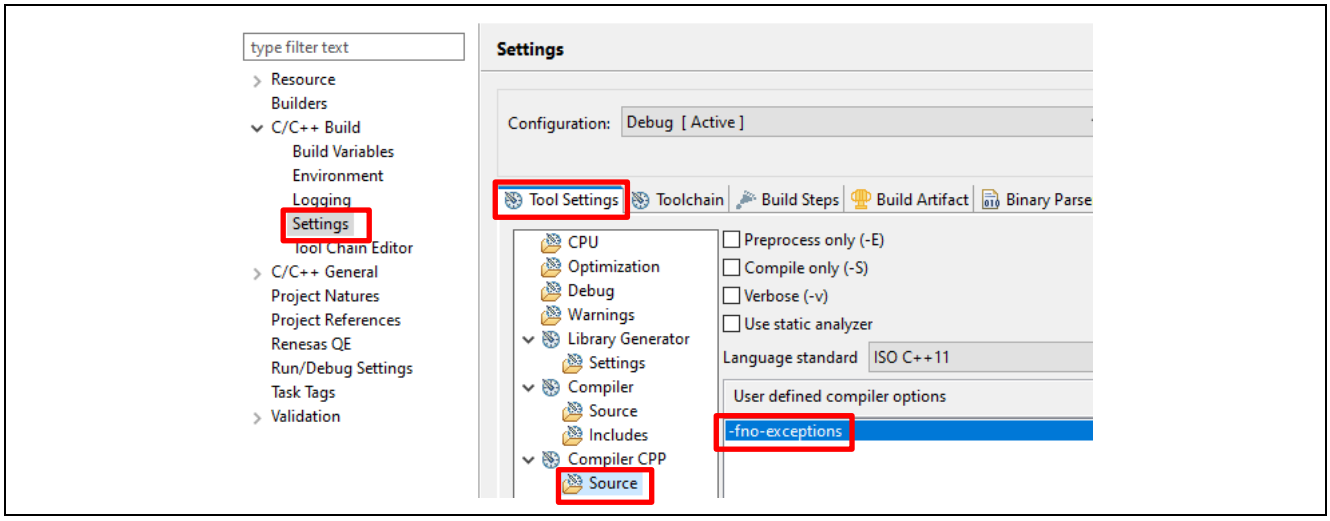


Figure 24. Disable Exception Handling

4.3 Compile and Run the Application

After the LLVM for Arm is integrated, follow the “Importing an Existing Project into e² studio” section in the FSP User’s Manual to import the face detection project into a new workspace. For details on understanding developing RA applications and how to use the various drivers using Renesas-supported IDE and FSP, refer to the FSP User’s Manual section “Starting Development” and sections on the corresponding drivers.

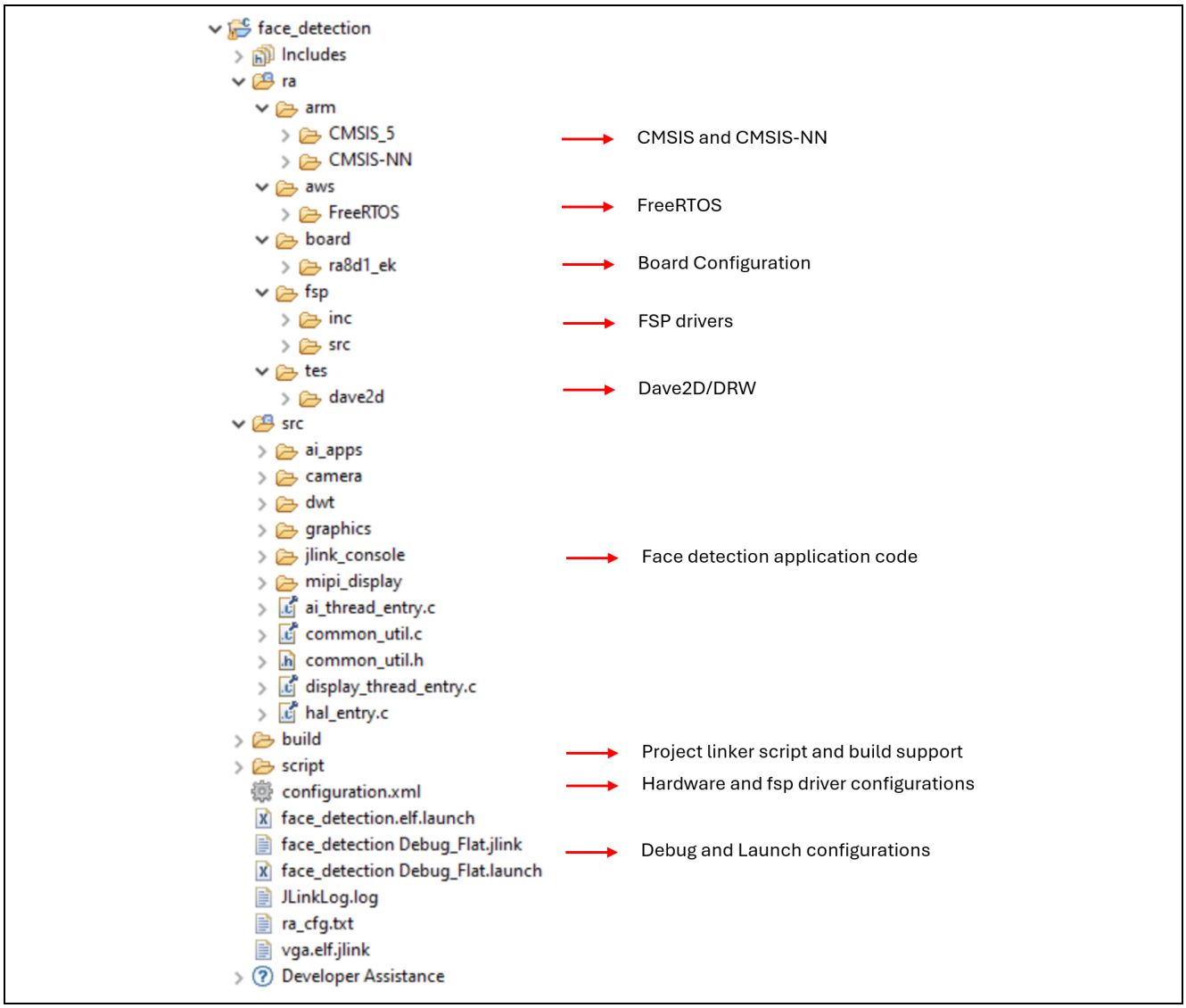


Figure 25. Face Detection Software Structure

The following are some additional details on the application code.

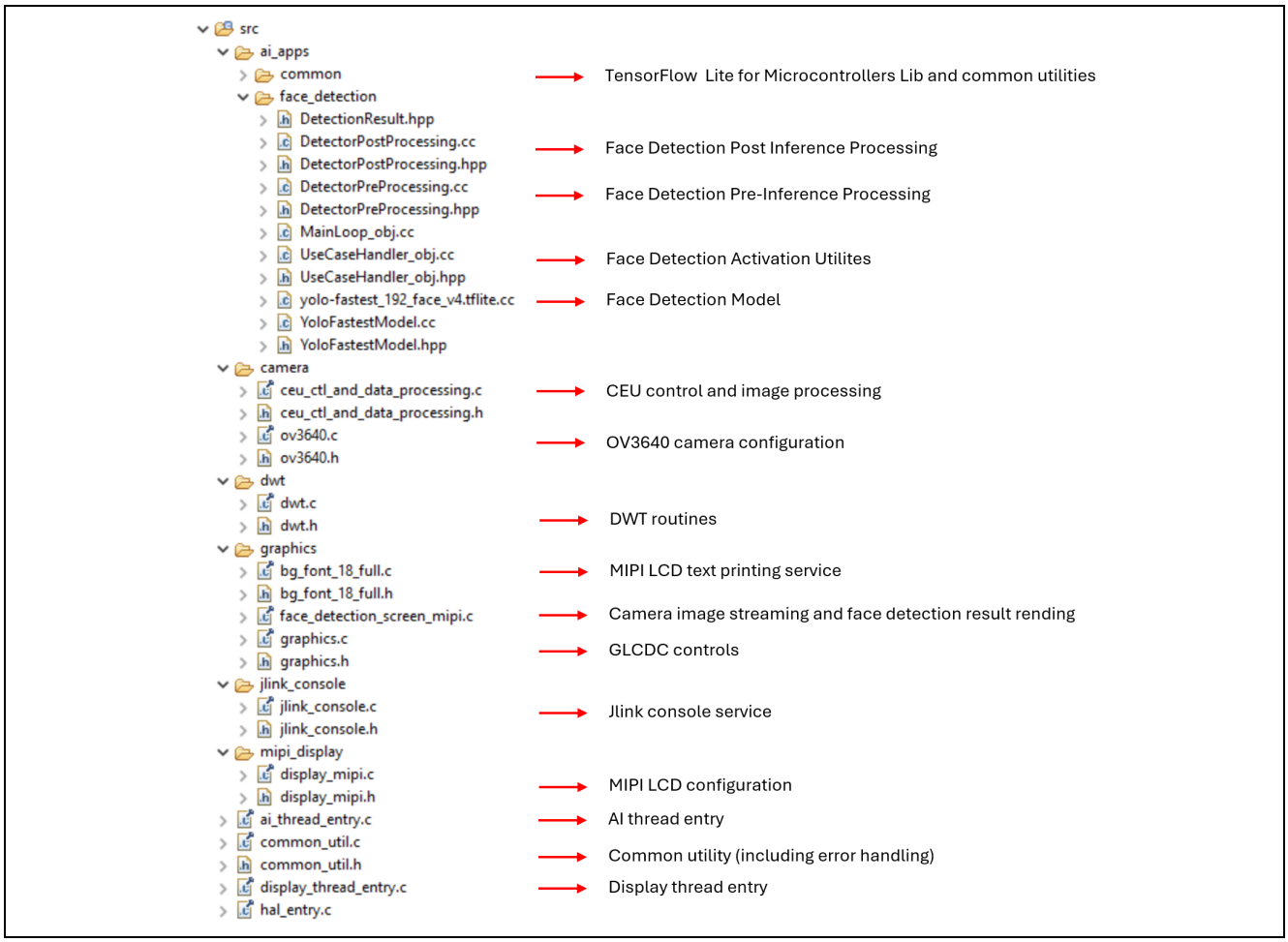
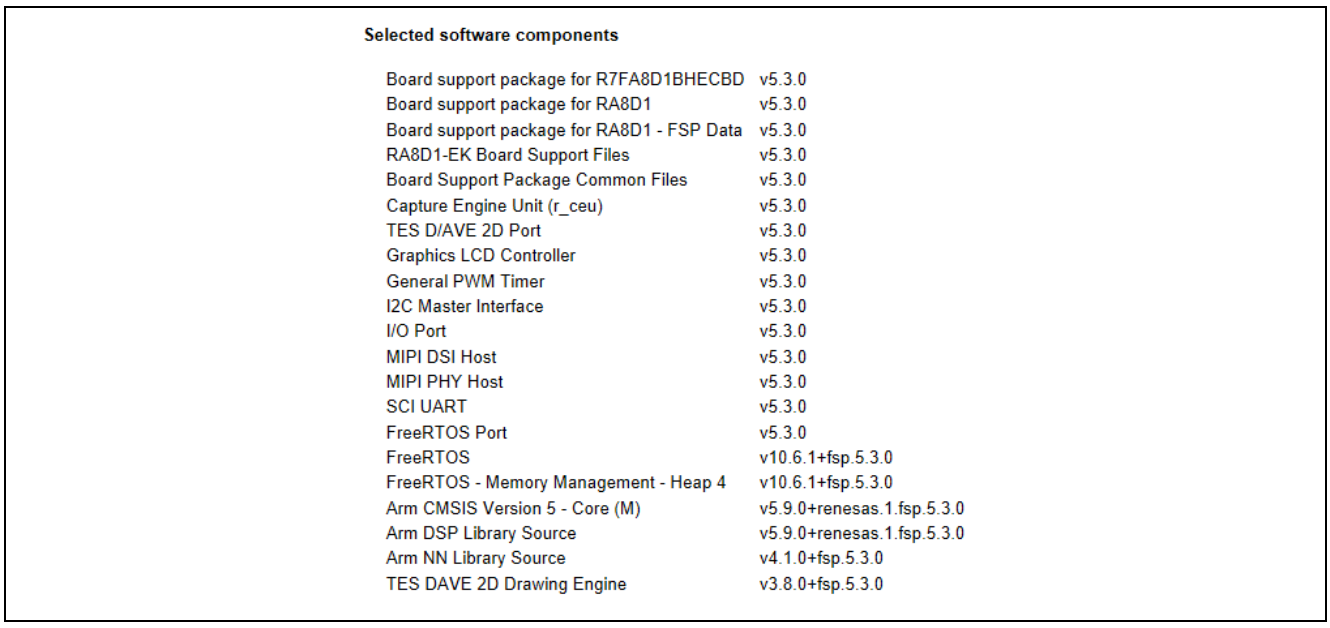


Figure 26. Face Detection Application Code

After the project is imported, double-click **configuration.xml** to open the RA configurator. Navigate to the Summary page to observe the FSP components used in this example project.



Selected software components

Board support package for R7FA8D1BHECBD	v5.3.0
Board support package for RA8D1	v5.3.0
Board support package for RA8D1 - FSP Data	v5.3.0
RA8D1-EK Board Support Files	v5.3.0
Board Support Package Common Files	v5.3.0
Capture Engine Unit (r_ceu)	v5.3.0
TES D/AVE 2D Port	v5.3.0
Graphics LCD Controller	v5.3.0
General PWM Timer	v5.3.0
I2C Master Interface	v5.3.0
I/O Port	v5.3.0
MIPI DSI Host	v5.3.0
MIPI PHY Host	v5.3.0
SCI UART	v5.3.0
FreeRTOS Port	v5.3.0
FreeRTOS	v10.6.1+fsp.5.3.0
FreeRTOS - Memory Management - Heap 4	v10.6.1+fsp.5.3.0
Arm CMSIS Version 5 - Core (M)	v5.9.0+renesas.1.fsp.5.3.0
Arm DSP Library Source	v5.9.0+renesas.1.fsp.5.3.0
Arm NN Library Source	v4.1.0+fsp.5.3.0
TES DAVE 2D Drawing Engine	v3.8.0+fsp.5.3.0

Figure 27. FSP Components

Next, Click Generate Project Component to extract the pin, interrupt, and thread configuration data.

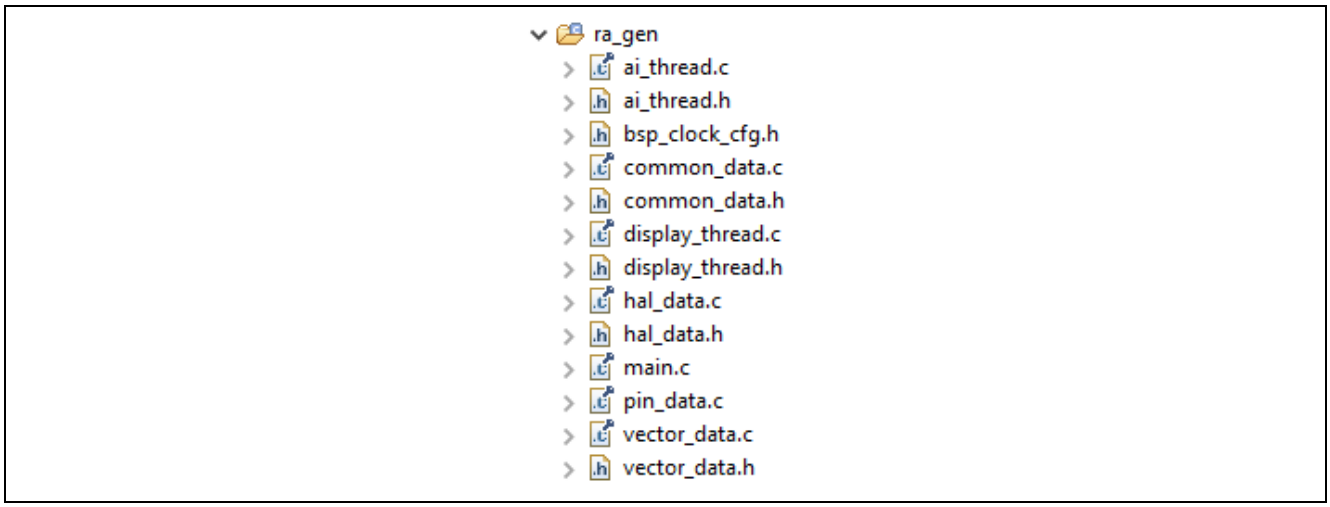


Figure 28. ra_gen folder

Now click the hammer icon to compile the project. The project should be compiled with no errors. It takes several minutes to compile the project. There are a large number of warnings from the usage of third-party software.

Next, right, click on the project folder face_detection, then select Debug As->Renesas GDB Hardware Debug to run the application. After the debug session is launched, click Resume twice to run the application. Finally, look into the camera to observe the face detection in action. If the camera image is blurry, rotate the camera focus near the lens to focus the image.

Note that as only the middle 240x240 of the original QVGA image is sent to the AI inference engine, the face detection will be effective in the middle 480x480 of the MIPI display screen.

4.4 Evaluate the System Performance

4.4.1 Data Watchpoint and Trace (DWT) for Time Recording

The Cycle counting capability of the DWT is used to measure the inference time usage, MIPI LCD refreshing, and CEU image capture framerate. Simple APIs are provided to enable DWT and acquire the current cycle count. The APIs are provided in \src\dwt\dwt.h. The APIs are very easy to understand; they are not duplicated here.

4.4.2 J-Link Virtual Console as User Interface

The jlink console using UART9 on PA15 and PA14 is used to display the system information configured with baudrate 115200. The jlink 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 (Jlink CDC UART Port).

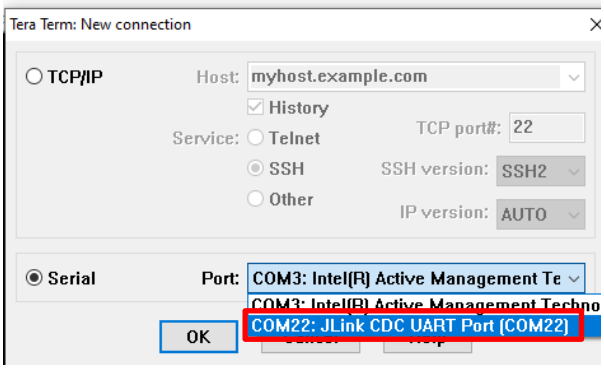


Figure 29. Select the JLink Console Connection

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

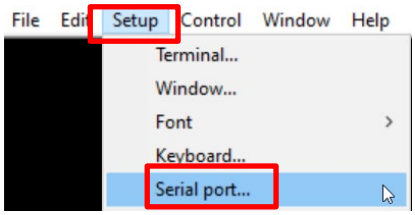


Figure 30. Open the "Serial port" interface

Update the Speed to 115200 and click New setting to commit the update.

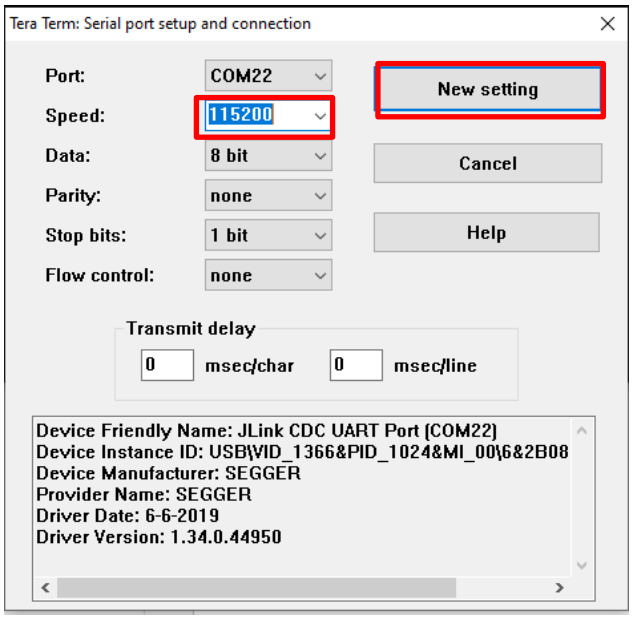


Figure 31. Configure the Tera Term

By default, the output on the Jlink console is enabled. Comment out the following line to disable the statisitc output.

#define ENABLE_STATISTICS_OUTPUT

The time reported here includes the influence of the software design and the SDRAM being accessed by multiple peripherals. They can not be used to accurately time these operations. This utility is provided to help with system debugging.

```
pre Inference Image Processing takes 2 ms; image rotation time takes 2 ms
mipi lcd display is refreshed at 30 fps;  camera capture is at 18 fps
```

Figure 32. Statistic Output

4.4.3 Evaluating the Helium Usage

The default configuration of this application project sets the optimization level to -O2, which enables Helium as well as Auto-Vectorization.

To disable Helium, navigate to the CPU property, change the **ARM family** configuration to "**Toolchain default**" and add the following configuration in the project: **-mcpu=Cortex-M85+nomve**.

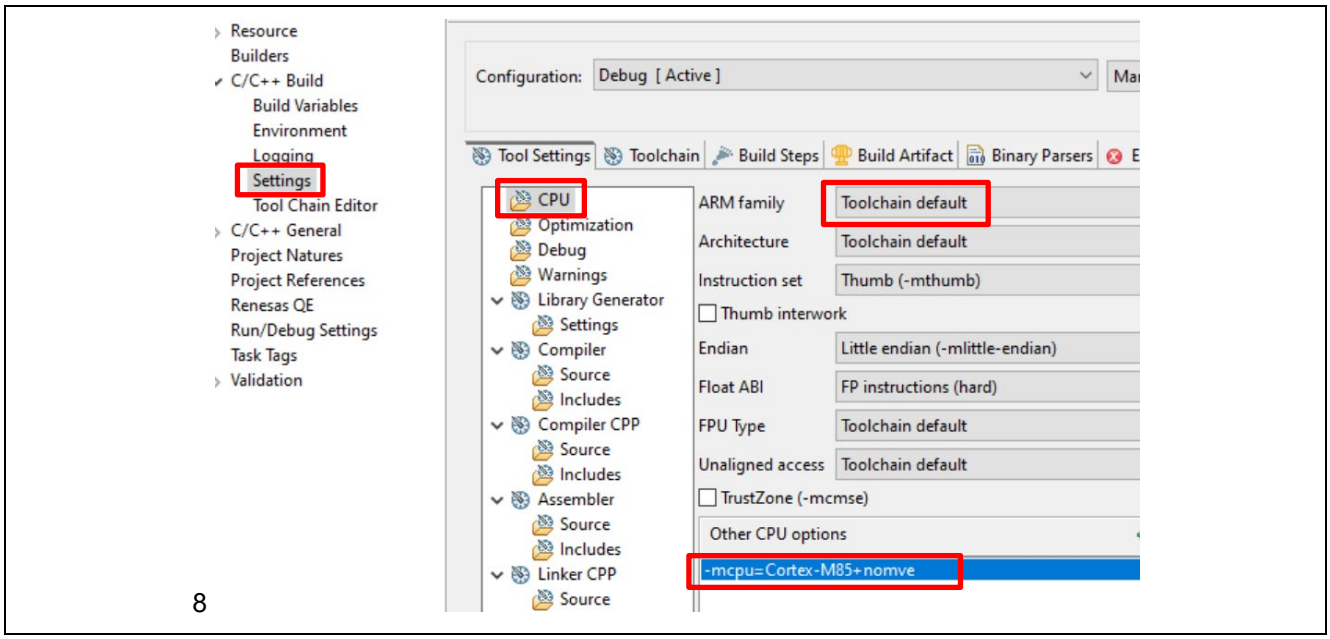


Figure 33. Disable Helium

To ensure a clean build of the project on a configuration change, the project is provisioned with a Pre-build step to remove the .elf file upon a build command: rm -f ${ProjName}.elf.

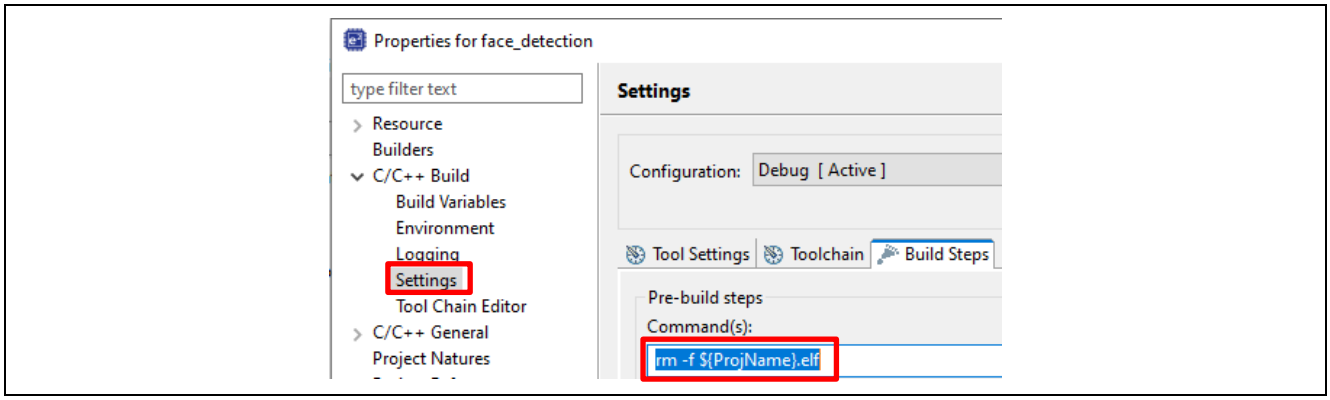


Figure 34. Pre-build Step

The following is the inference time comparison when Helium is enabled and disabled with Data Cache enabled.

- The first set of data is collected using the default project, with the Display thread having a higher priority (priority 6) than the AI thread (priority 5).

Table 10 Face Detection Application Usage with Helium and Data Cache (FSP v5.3.0)

Helium and Data Cache Configuration		Inference time (ms)
Data Cache Enabled	Helium Enabled	135
	Helium Disabled	432

- The second set of data is collected after swapping the thread priority between the Display thread and the AI thread (so the Display thread is at priority 5 and the AI thread is at priority 6).

Table 11 Face Detection Application Usage with Helium and Data Cache (FSP v5.3.0)

Helium and Data Cache Configuration		Inference time (ms)
Data Cache Enabled	Helium Enabled	126
	Helium Disabled	403

5. References

Renesas RA Family RA8D1 Develop Graphic Application using MIPI on RA8D1 (R01AN7362).

Renesas RA FSP User's Manual

Renesas RA Family RA8D1 User's Manual: Hardware (R01UH0995)

6. Website and Support

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

EK-RA8D1 Resources	renesas.com/ra/ek-ra8d1
RA8D1 Resources	renesas.com/ra/ra8d1
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
1.00	07/08/24	—	Initial release

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/.