Renesas RA Family

RA8 CEU Webcam Getting Started Guide

Introduction

This application project demonstrates how the Renesas RA8 MCU, featuring an integrated Camera Engine Unit (CEU) and advanced networking capabilities, can be used to implement an embedded webcam solution. This solution is designed for applications that require continuous image capture and updates to a standard web browser. With Arm® Cortex®-M85 core and networking capabilities, RA8 provides an efficient way to combine imaging and connectivity in an MCU.

This application note guides through the steps to implement a basic webcam-like application using RA8D1, including:

- Application demonstration.
- System overview.
- Software and Hardware configuration.
- Application highlights.

Required Resources

The following resources are referenced throughout this application note.

Development Tools and Software

- e² studio version: 2025-12 (25.12.0)
- LLVM Embedded Toolchain for ARM v21.1.1
- Renesas Flexible Software Package (FSP) v6.4.0
- SEGGER J-Link RTT-Viewer version and 8.74 Tera Term version 4.99.
- Microsoft Edge Web Browser Application version 144.0.3719.82.

Hardware

- Renesas RA8D1 Evaluation Kit (RA8D1 MCU Group) (RA8D1 Group User's Manual: Hardware)
- OV3640 camera included in the EK-RA8D1 package.
- Ethernet Router to provide DHCP service and other network functionalities.
- Host PC for programming, debugging, and status display on the Terminal.

Contents

1. Application Project Requirements

The following items are required to build and run the RA8 CEU Webcam Getting Started Guide.

1.1 Required Development Tools

The following tools are required for building the application project. The tools are available for downloading from Renesas.com:

- e^2 studio version: 2025-12 (25.12.0).
- LLVM Embedded Toolchain for ARM v21.1.1.
- Renesas Flexible Software Package (FSP) v6.4.0.

Please refer to this link https://www.renesas.com/en/software-tool/ra-flexible-software-package-fsp or search for the keyword “RA FSP Download” on the internet to download the package.

1.2 Required Hardware

The following items are required to operate the application project:

- RA8D1 Evaluation Kit.
- OV3640 Image Sensor.
- Ethernet Router.
- PC or Laptop.

Additionally, the following items are required for the hardware connection:

- 1 x USB Debug Cable.
- 2 x Ethernet Cable.

1.3 Required Software

The following software items are required for the application project operation:

- SEGGER RTT-Viewer version 8.74.
- Tera Term version 4.99.
- Web Browser application.

2. CEU Webcam Getting Started Guide

2.1 Project Build

After importing the application project, the user can build, download, and run the project. There are two terminal instances provided in this application for user interface, each using different terminal software. The user should select the desired instance before building the project.

Step 1: Navigate to the top-left corner of e² studio.

Select **"File"** → **"Import"** → **"General"** → **"Existing Project into Workspace"**.

Then select the root directory and browse to the workspace or the application directory. Check the **"Search for nested projects"** option to import the entire application project, including nested files. Click **"Finish"** and wait until the project import completes.

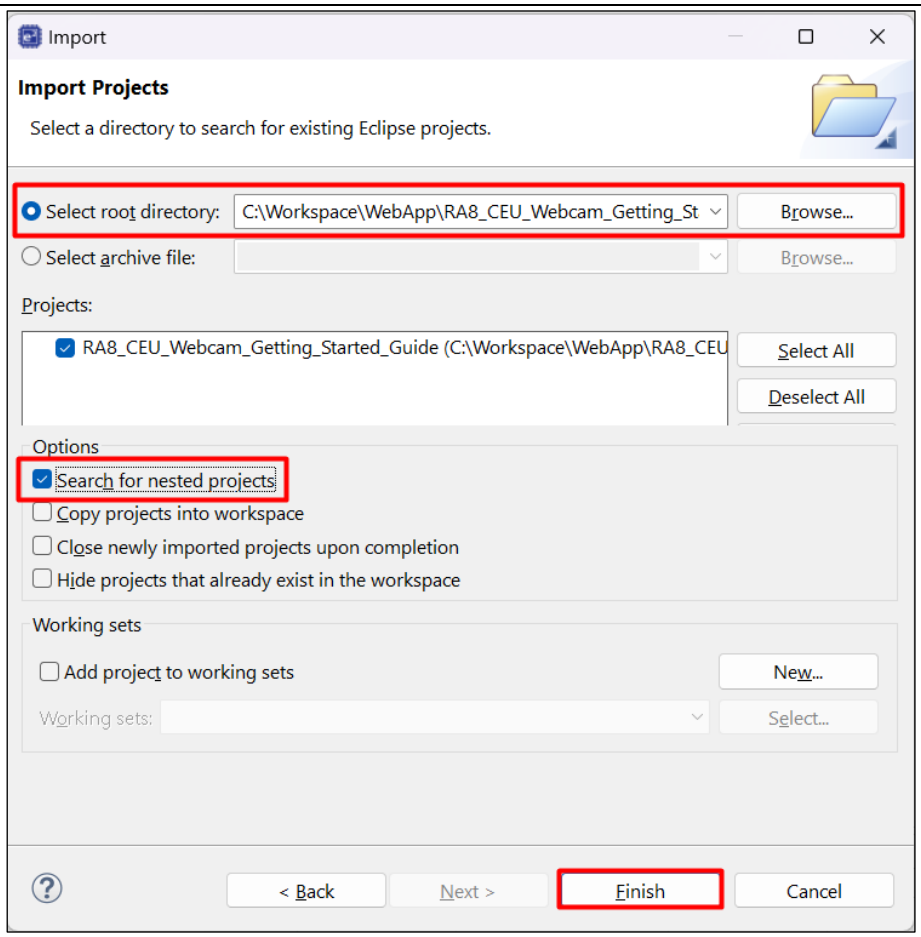

Figure 1. Import project to e² studio

In the **"Project Explorer"** window, expand project tree and double-click the **"configuration.xml"** file to open the FSP Configuration perspective. The user can view the application information in the **"Summary"** tab.

Step 2: Next, the user should select the terminal instance to run. If users connect to the wrong terminal instance configured, there will be nothing shown on the Terminal view.

Right-click the project folder **"RA8_CEU_Webcam_Getting_Started_Guide"** → then click **"Properties"** → expand **"C/C++ Build"** section → select **"Settings"** → then open the **"Tool Settings"** tab → under **"Compiler"** configuration, select **"Includes"** section.

In the **"Macro Defines (-D)"** box, double-click the **"USE_VIRTUAL_COM"** macro and modify its value to select terminal instance as below.

- **USE_VIRTUAL_COM** = 0 → Run the application with **SEGGER RTT Viewer v8.74**.
- **USE_VIRTUAL_COM** = 1 → Run the application with **Tera Term v4.99**.

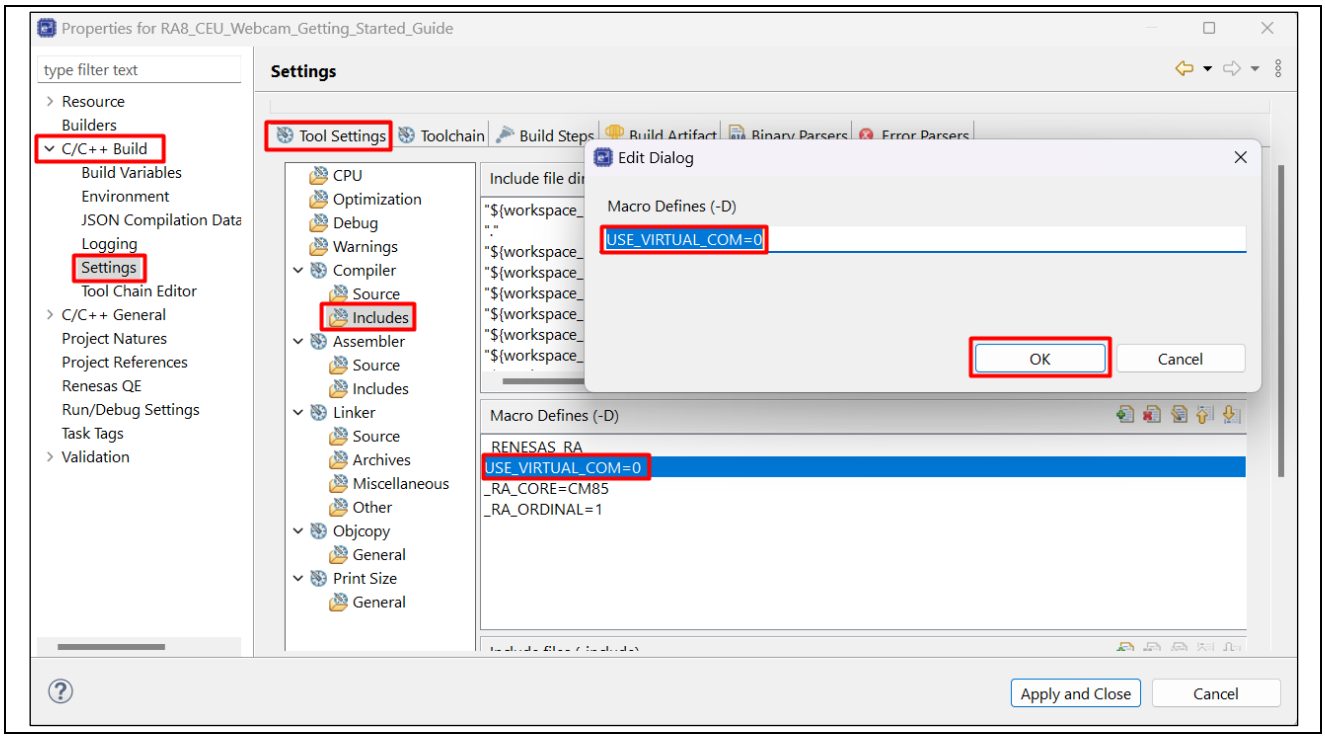

Figure 2. Change Terminal instance

Next, click **"Apply and Close"**.

Step 3: Finally, select the application folder name, then at the top left corner of the e² studio, click the **"Build"** icon "  " to start building the project. Wait until the build process completes without errors.

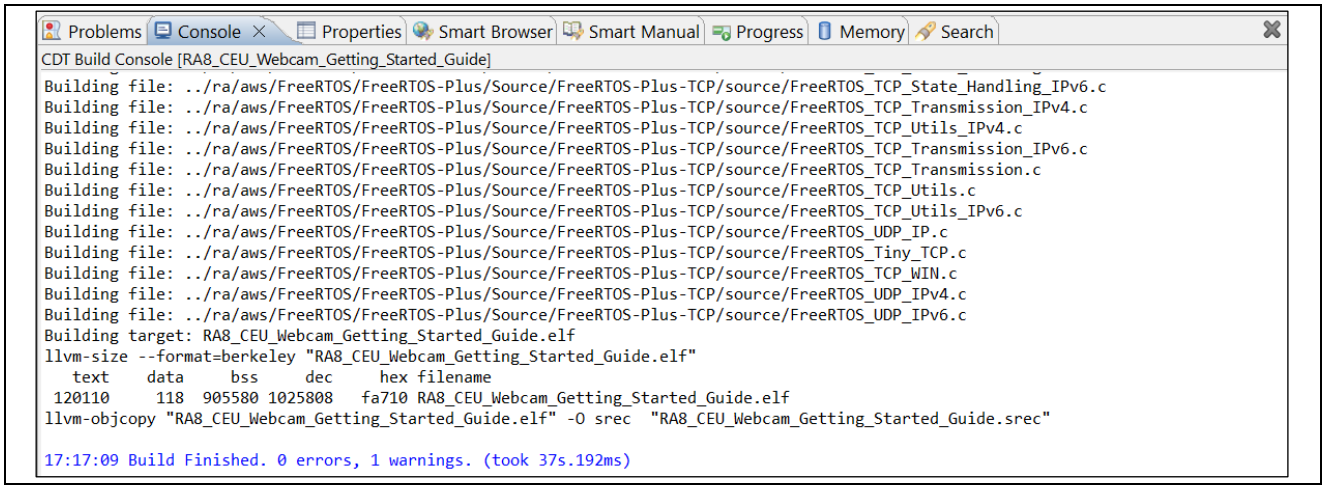

Figure 3. The project finished building without errors

2.2 Hardware Connection

Firstly, connect the OV3640 image sensor to the EK-RA8D1 board via the Camera Expansion Port on the evaluation kit, as shown in the hardware connection Figure 4. Carefully align pin 1 of the OV3640 module with the 3.3V pin of the Camera Expansion Port.

Next, connect the EK-RA8D1 to the host PC or laptop using a Micro USB cable for debugging and logging.

Then, use Ethernet cables to connect both the EK-RA8D1 and the host PC to the router for network establishment. Ensure that both devices are located on the same network.

Finally, to configure the system for operation, adjust the configuration switches on the EK-RA8D1. Turn **SW1-3 ON** to activate the camera for image capture, and **SW1-4 ON** to enable Ethernet A for network

connectivity. Once these switches are set, power on the device and proceed with the next steps for network initialization and real-time operation as described in the demonstration workflow.

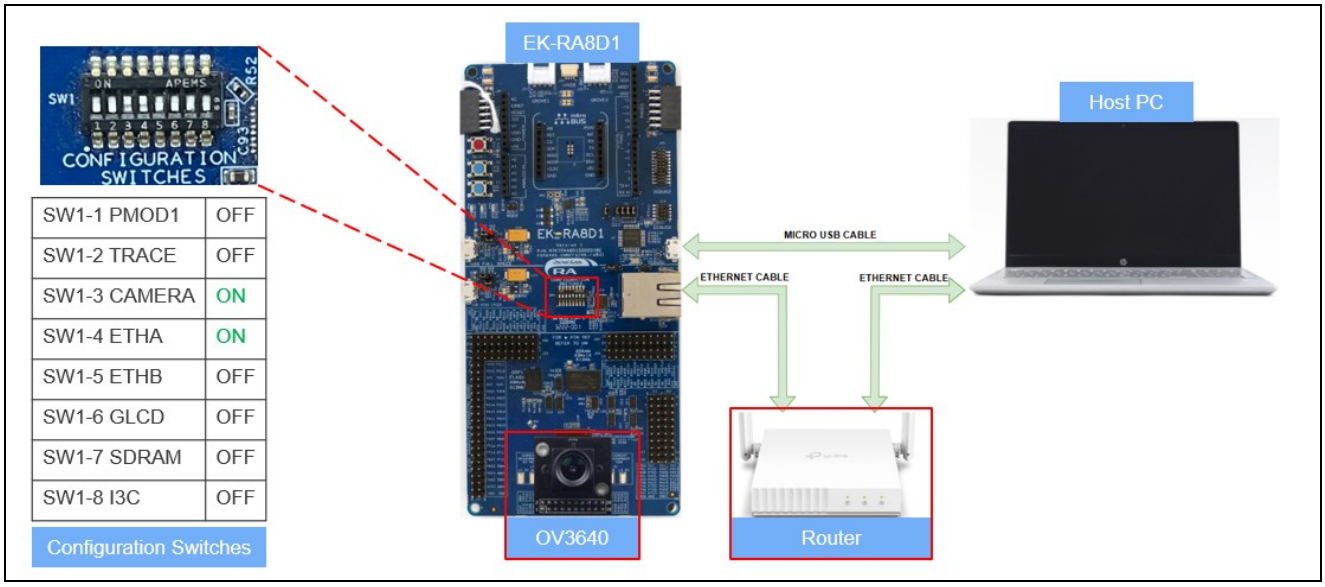

Figure 4. Hardware Connection

Note: Turn **OFF** all other switches to avoid any configuration conflicts.

2.3 Running RA8 CEU Webcam Application

2.3.1 Setup to running with SEGGER RTT-Viewer

Navigate to the project **"Debug"** folder and open **"RA8_CEU_Webcam_Getting_Started_Guide.map"** file with text editor to obtain the RTT Control Block address as Figure 5.

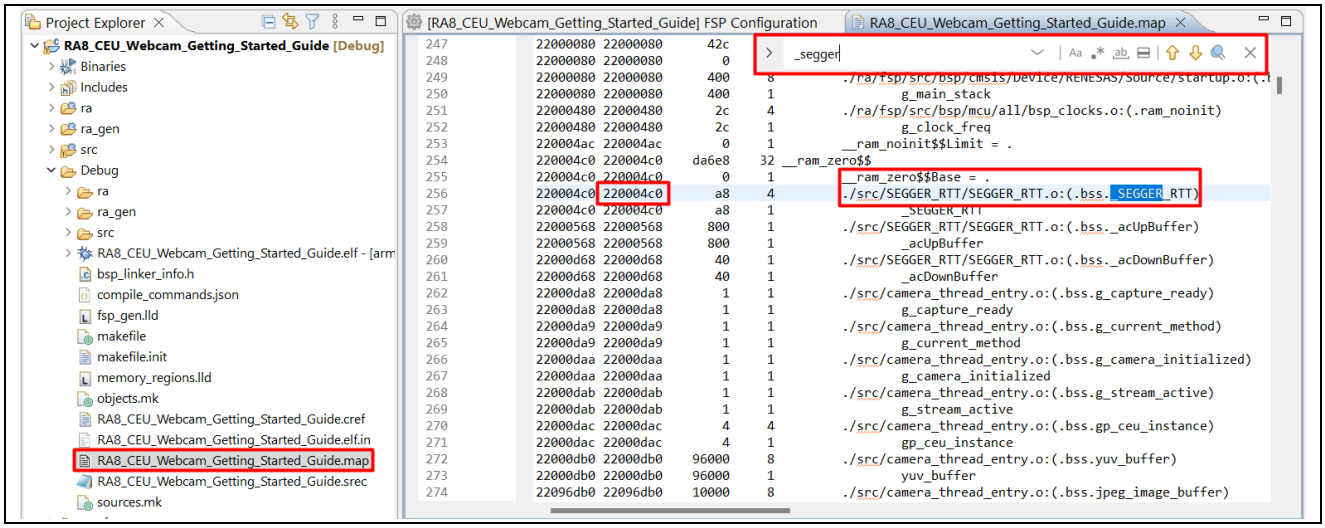

Figure 5. Obtain Control Block Address in .map file

After connecting the board to the host PC, open the **SEGGER RTT Viewer** application to complete the connection.

Click **"File"** → **"Connect"**, and the **"RTT Viewer Configuration"** window will appear. Complete the configuration shown in Figure 6, ensure to fill the **"RTT Control Block"** field with the obtained RTT Control Block address, and click **"OK"**.

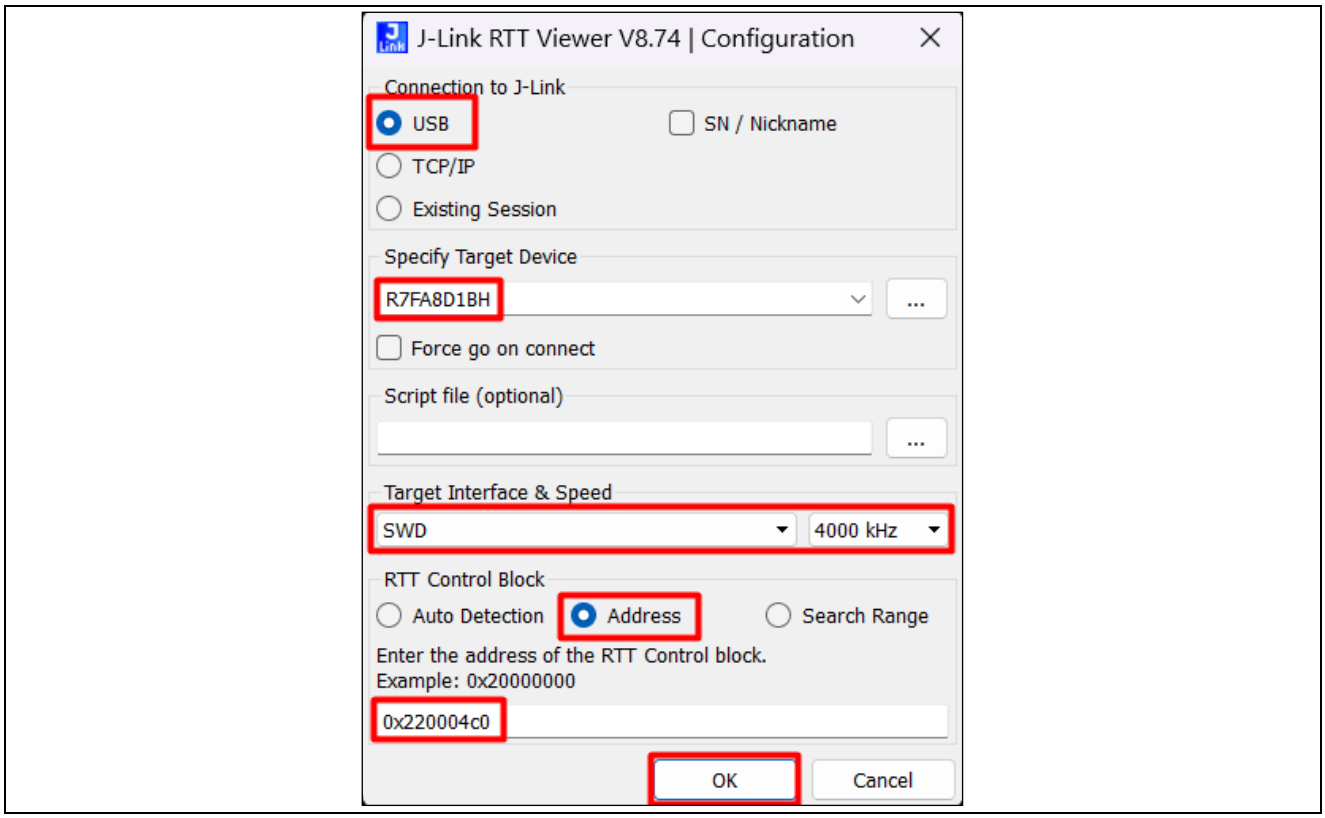

Figure 6. RTT-Viewer Configuration

2.3.2 Setup to running with Tera Term

Remember to connect the RA board to the host PC before opening the Tera Term software.

At the **"Serial"** option, select **"COMx: JLink CDC UART Port (COMx)"** → click **"OK"**.

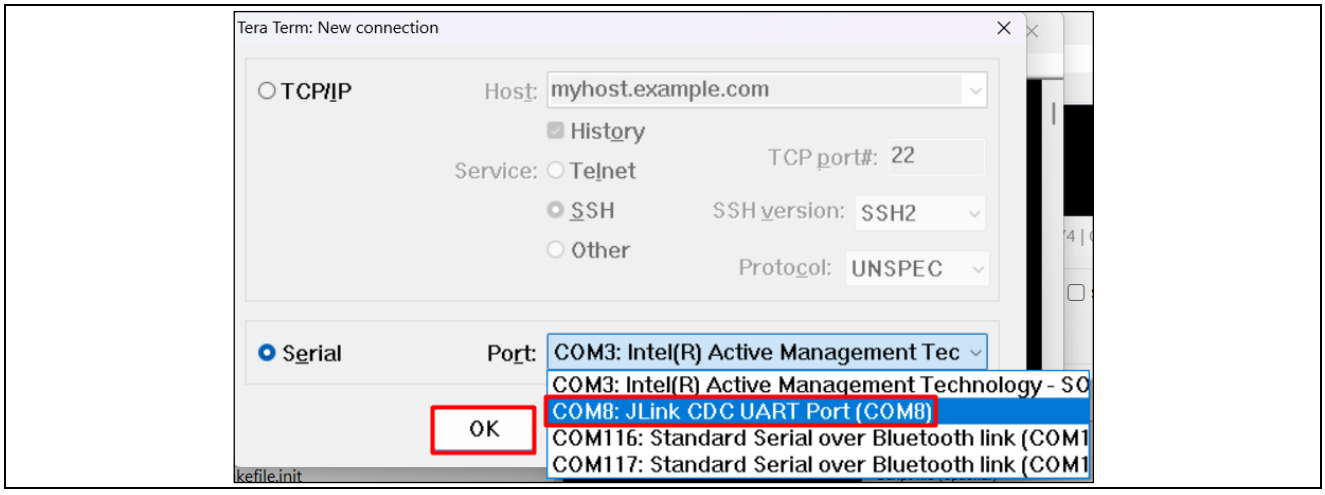

Figure 7. Tera Term: New connection window

After the **"COMx – Tera Term VT"** terminal opens, the user needs to configure the **"Serial port"** and **"Terminal"** settings.

Select **"Setup"** → **"Terminal…"** → enable the **"Local echo"** option to echo back what was typed in Tera Term → click **"OK"**.

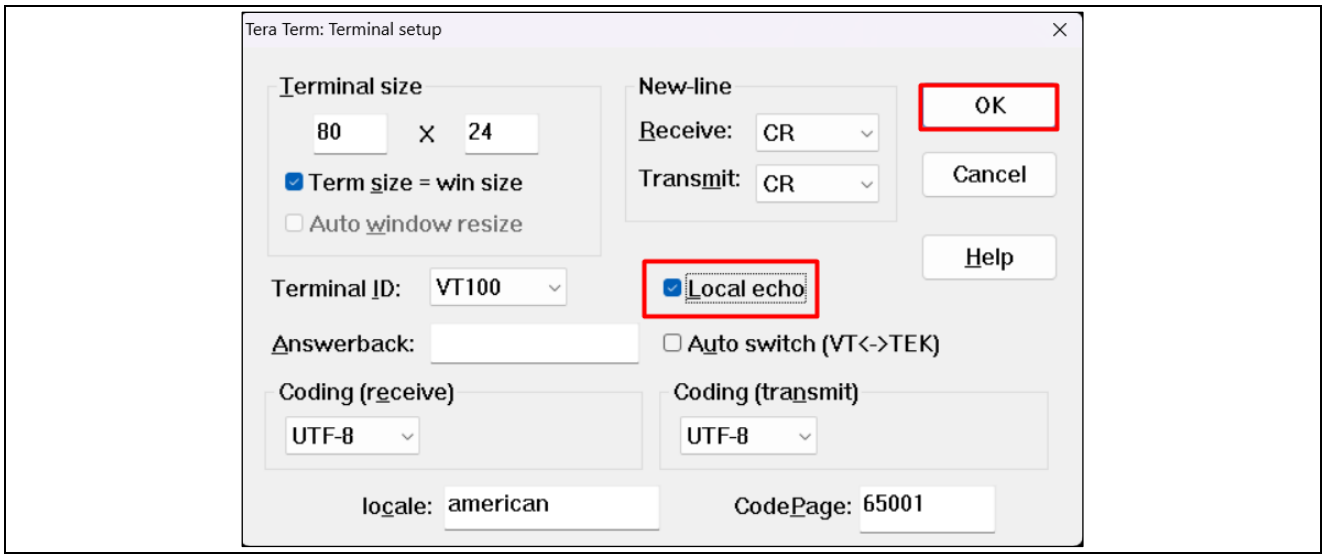

Figure 8. Tera Term: Terminal setup window

Then, select **"Setup"** → **"Serial port…"** → configure the serial port settings as shown in the Figure 9 → click **"OK"**.

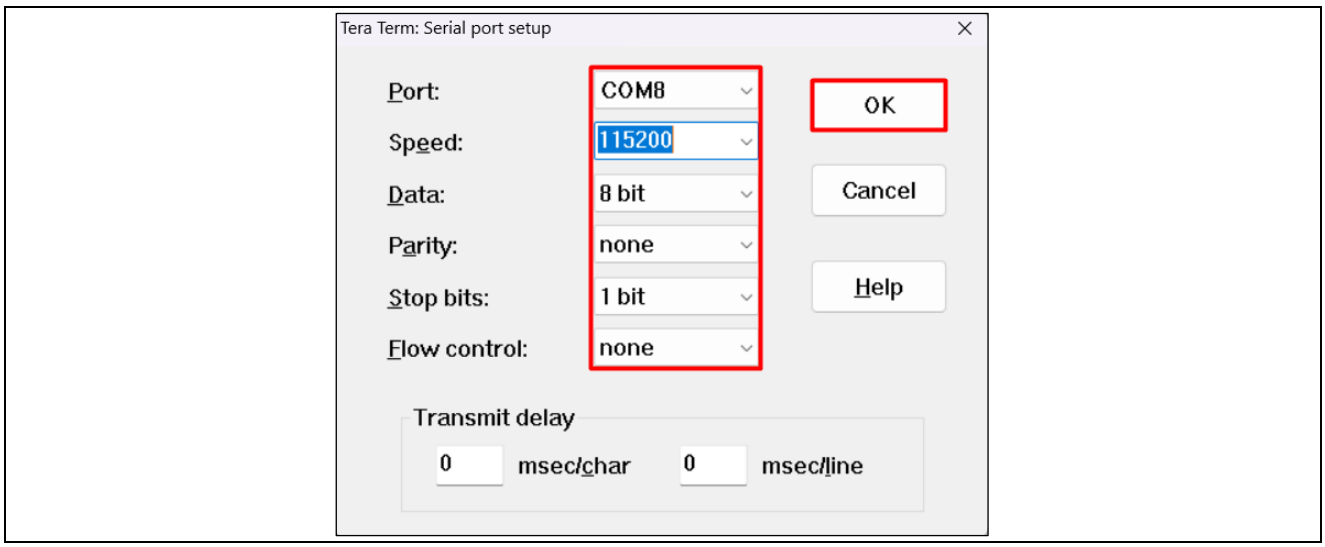

Figure 9. Tera Term: Serial port setup window

2.3.3 Download and Operate Application

Step 1: After completing the build instructions in section 2.1 and configuring all necessary hardware in section 2.2, the user can now download and run the project.

Select the project folder name, then click the **"Debug"** icon "  " at the top left corner of the e² studio. The debugger will start and download the application image to the board.

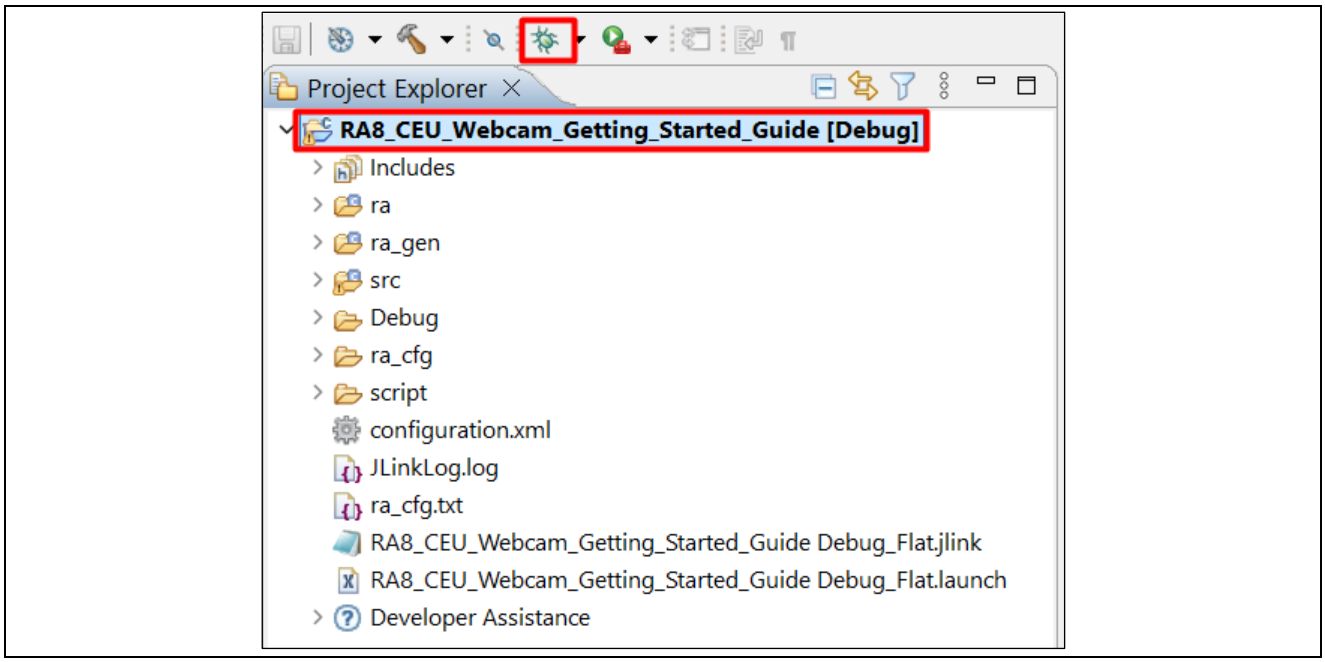

Figure 10. Start debug to run CEU Webcam application

Step 2: Next, connect to the terminal software corresponding to the selected terminal mode.

See 2.3.1 when running with **SEGGER RTT Viewer**, or section 2.3.2 when running with **Tera Term**.

Click the **"Resume (F8)"** symbol " " twice to start running the application. Wait for the application to complete network establishment, the Webpage URL and Encoding Method Selection Menu will then be shown on terminal as Figure 11.

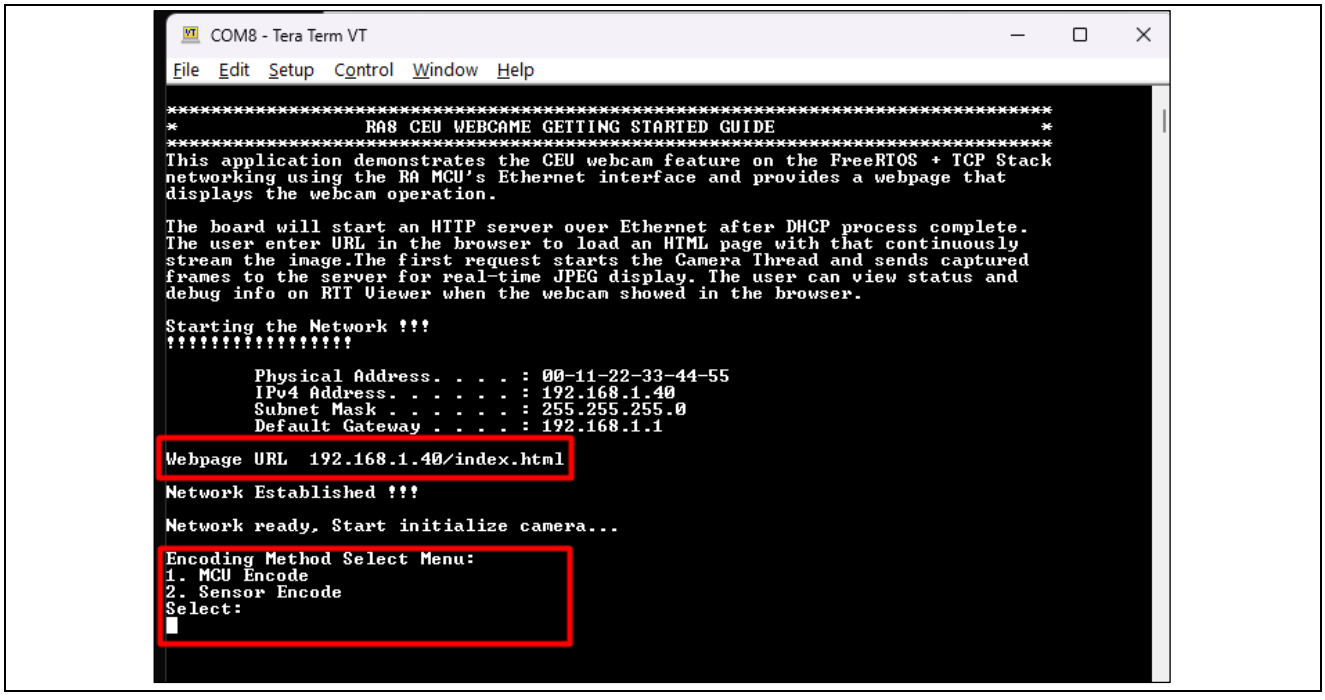

Figure 11. CEU Webcam application interface

Step 3: Start operating the application functionalities.

RA8 CEU Webcam application implements two image encoding methods: MCU encoding and Sensor encoding. In MCU Encoding method, images are encoded by third-party software, whereas in the Sensor encoding method, images are encoded by the OV3640 camera's internal encoder. Refer to section 5 for detailed information.

Enter **“1”** to select **“MCU Encode”** method.

Follow the instructions shown in the terminal, then enter the Webpage URL in Microsoft Edge web browser to start the webcam demonstration.

Note: The application operates correctly across different web browsers; however, the UI may not be displayed correctly on browsers other than Microsoft Edge due to JavaScript compatibility.

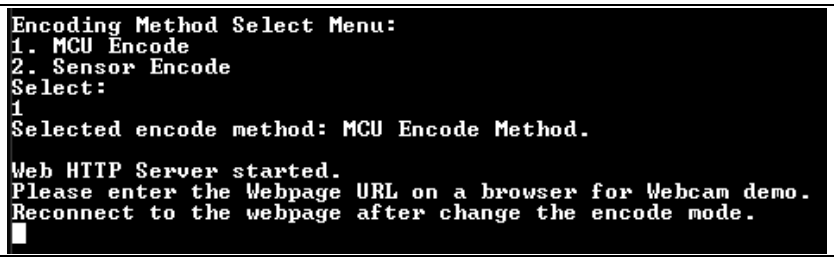

Figure 12. MCU Encode Method selected

The webcam application will begin streaming JPEG images on the webpage UI.

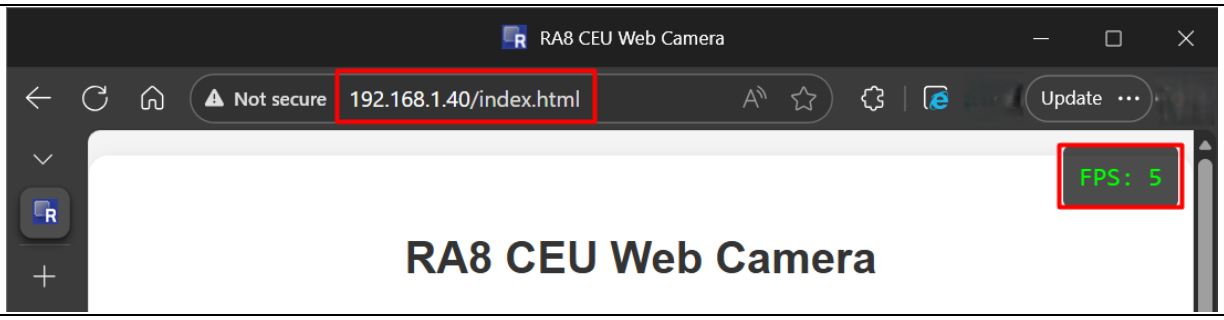

Figure 13. RA8 CEU Webcam application UI

Next, enter **“1”** again to **stop the current stream to change encode mode**.

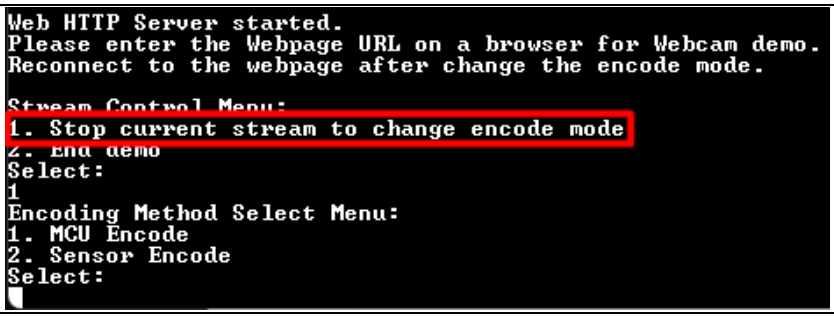

Figure 14. Stop current stream to change encode mode

When the encoding method selection menu appears, choose option **“2”** to switch to **“Sensor Encode”** method.

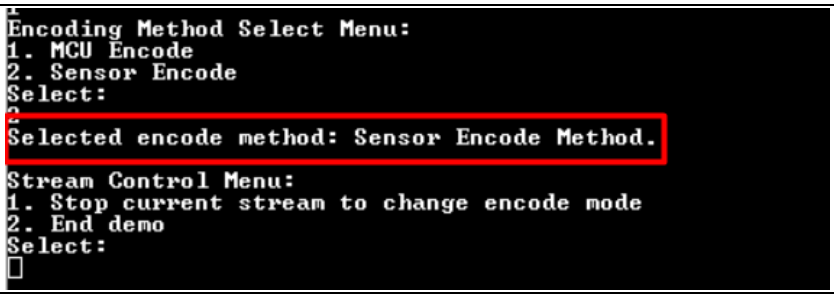

Figure 15. Change encode mode in run time

After changing the encode mode, the user must enter the Webpage URL again or simply refresh the browser.

The user can continue switching encode modes or select option **“2”** to **end the demonstration** while the webcam is streaming. The application demonstration will automatically end after several seconds.

```
Stream Control Menu:
1. Stop current stream to change encode mode
2. End demo
Select:
2
Demo Ending...

================================================================
=                          Server closed                       =
=                          Demo end here                       =
================================================================
```

Figure 16. Demonstration end

3. CEU Webcam Design Workflow

3.1 Application Description

The RA8 CEU Webcam application demonstrates a webcam reference design on the Renesas RA8D1 microcontroller using the Camera Engine Unit (CEU). The application captures image data from an external image sensor and processes the captured frames within the RA8 device to support real-time image streaming.

The objective of this application is to demonstrate the integration of an external image sensor with the RA8 CEU, along with the implementation of an embedded web server that streams the captured image data to a server-based web interface. The application showcases camera initialization, essential register configuration, image capture control, and the delivery of image data for continuous display on a web page.

3.2 High Level Design

This application implements a TCP/IP-based embedded web server that streams MJPEG video captured from the OV3640 image sensor. The project demonstrates a CEU webcam application running on the Renesas RA8D1 MCU, interfaced with the OV3640 image sensor through the Camera Engine Unit (CEU). The RA8D1 board functions as an HTTP server over Ethernet and obtains its IPv4 address dynamically using DHCP (Dynamic Host Configuration Protocol) from the connected router. Once the network is initialized, the application provides a webpage interface that continuously displays JPEG images captured by the camera in real time.

The workflow begins with system initialization and network setup. After the HTTP server is ready, the user enters the board’s IP address in a web browser to access the server. The server then delivers an HTML page with JavaScript that requests the MJPEG stream. Users can view status and debug information through RTT Viewer or Tera Term while observing the webcam demonstration in the web browser.

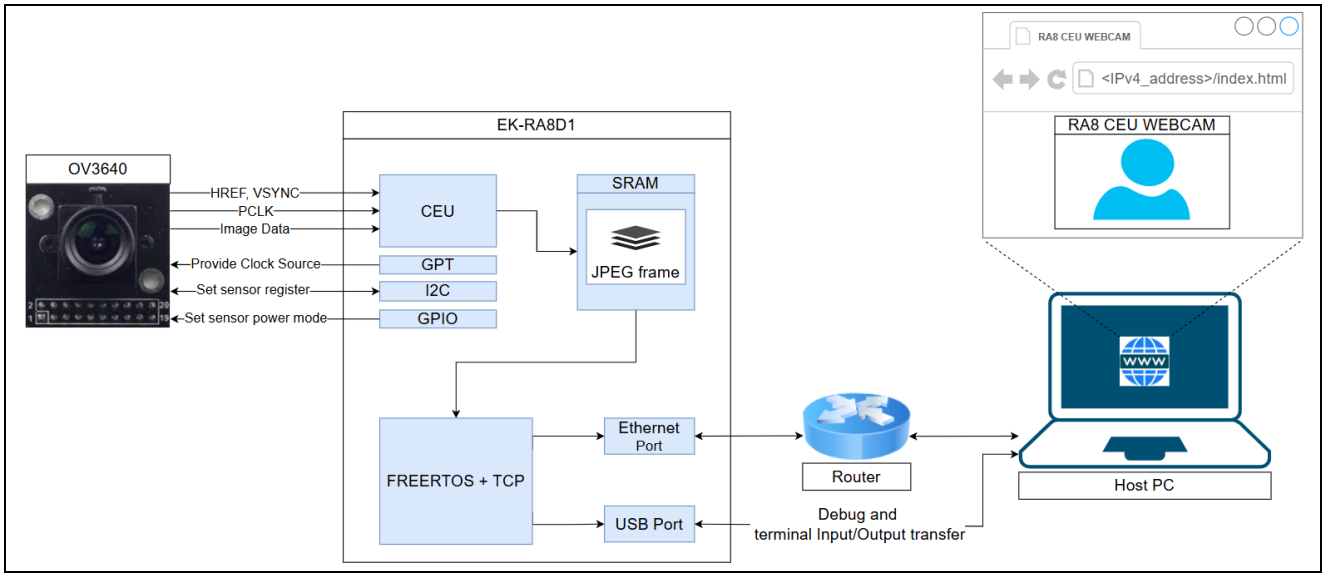

Figure 17. Application High Level Design

3.3 Technical Operating Principles

3.3.1 Network Establishment Process

The TCP/IP model is a layered networking architecture that defines how data is generated, transmitted, routed, and received across interconnected networks. In this application, the TCP/IP stack enables communication between the RA8D1 board and the host PC through a router. Figure 18 illustrates the TCP/IP model and example protocols of each layer, compared with the CEU Webcam application, the functions of each layer can be understood as follows:

- Application Layer: This layer handles user-level protocols such as HTTP. The RA8D1 runs an embedded HTTP server that provides the webpage interface and delivers MJPEG data to the client browser.
- Transport Layer: Responsible for reliable end-to-end communication. The application uses TCP to stream JPEG images to the browser, ensuring that packets arrive in order and intact.
- Internet Layer: This layer provides IP addressing and routing. The RA8D1 and host PC use IPv4 addresses assigned by the router to communicate over the network.
- Network Access Layer: Defines how data is transmitted over Ethernet. It manages MAC addressing, frame formatting, and physical signaling between the RA8D1 board, router, and PC.

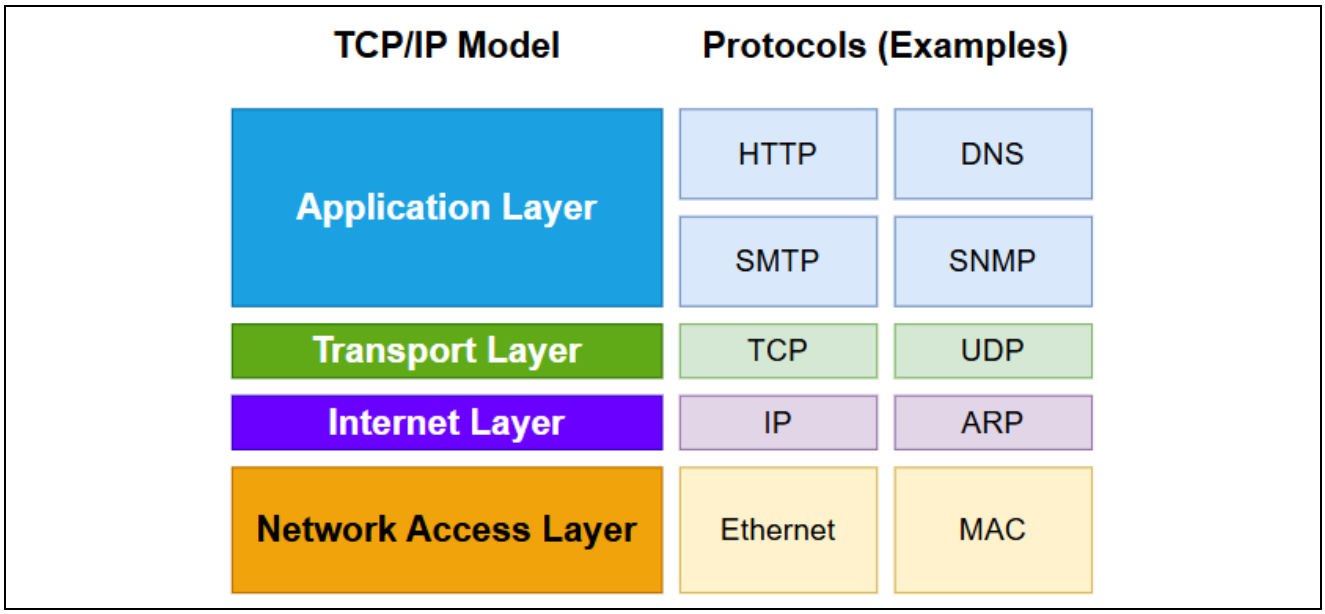

Figure 18. TCP/IP Model

The application uses DHCP to automatically assign an IPv4 address to both the RA8D1 board and the host PC. This eliminates the need for manual IP configuration.

The DHCP process proceeds as follows:

1. RA8D1 Evaluation Kit and the host PC connect to the same router.
2. RA8D1 Evaluation Kit sends a DHCP Discover request.
3. The router responds with a DHCP Offer.
4. RA8D1 Evaluation Kit sends a DHCP Request to accept the offered address.
5. The router sends a DHCP Acknowledge, providing the IP address and network parameters.

RA8D1 uses the FSP DHCP Client API to perform this negotiation. Once both devices have valid IP addresses, they can communicate over the TCP/IP network.

In summary, DHCP configuration plays a crucial role in enabling automatic IP assignment for this application. Once the network stack initializes, the DHCP Client is started, allowing the RA8D1 to obtain a valid IP address. Successful DHCP negotiation ensures stable communication for the embedded HTTP server and enables continuous MJPEG image streaming to the web browser.

3.3.2 Client and Server Communication

Client–server communication refers to the exchange of data between two entities in which one system (the client) requests a service, and another system (the server) provides that service. In this application, the web browser acts as the client, and the RA8D1 board functions as the server, delivering webcam streaming data.

There are several general methods for client–server communication, such as:

- Sockets Mechanism
- Remote Procedure Call
- Message Passing
- Inter-process Communication
- Distributed File Systems

This application specifically uses the **Socket mechanism**, where sockets act as communication endpoints between the client and the server. Sockets enable full bidirectional data exchange either locally or across a network, making them ideal for real-time MJPEG image transfer.

The RA8D1 board hosts an embedded HTTP server, which follows the standard request-response model:

1. The client (web browser) sends an HTTP request to access resources.
2. The server processes the request and responds with either the requested resource or an error message.

In this application:

- After the network is initialized, the user enters the board's IP address into a browser.
- The RA8D1 HTTP server responds with an HTML page containing JavaScript.
- This JavaScript sends requests for MJPEG frames.
- The server provides the latest JPEG image captured from the camera, enabling real-time webcam display.

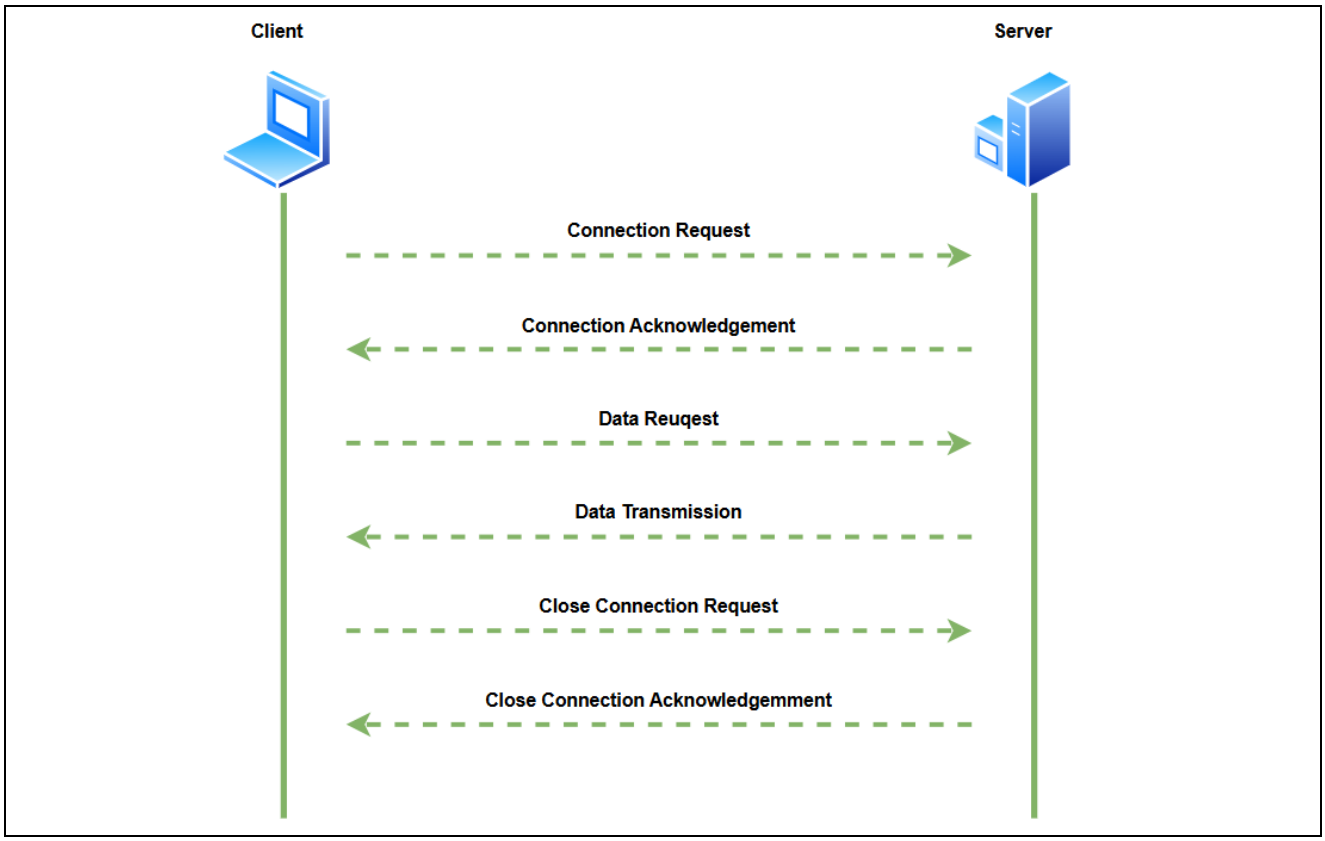

Figure 19. Client Server Communication using Socket Mechanism

The HTTP server on the RA8D1 board is configured to:

- Handle multiple simultaneous client connections efficiently.
- Serve static HTML content for the application UI.
- Deliver dynamic JPEG image data generated by the camera thread.
- Manage timing and resource usage for smooth streaming performance.
- Provide basic error handling for invalid or failed requests.

In summary, this application uses a socket-based HTTP client–server model where the browser requests MJPEG data and the RA8D1 board responds with real-time JPEG images streamed through an embedded web server.

4. CEU Webcam Detail Configuration

4.1 FreeRTOS Task Architecture and Resource Configuration

4.1.1 Thread Architecture

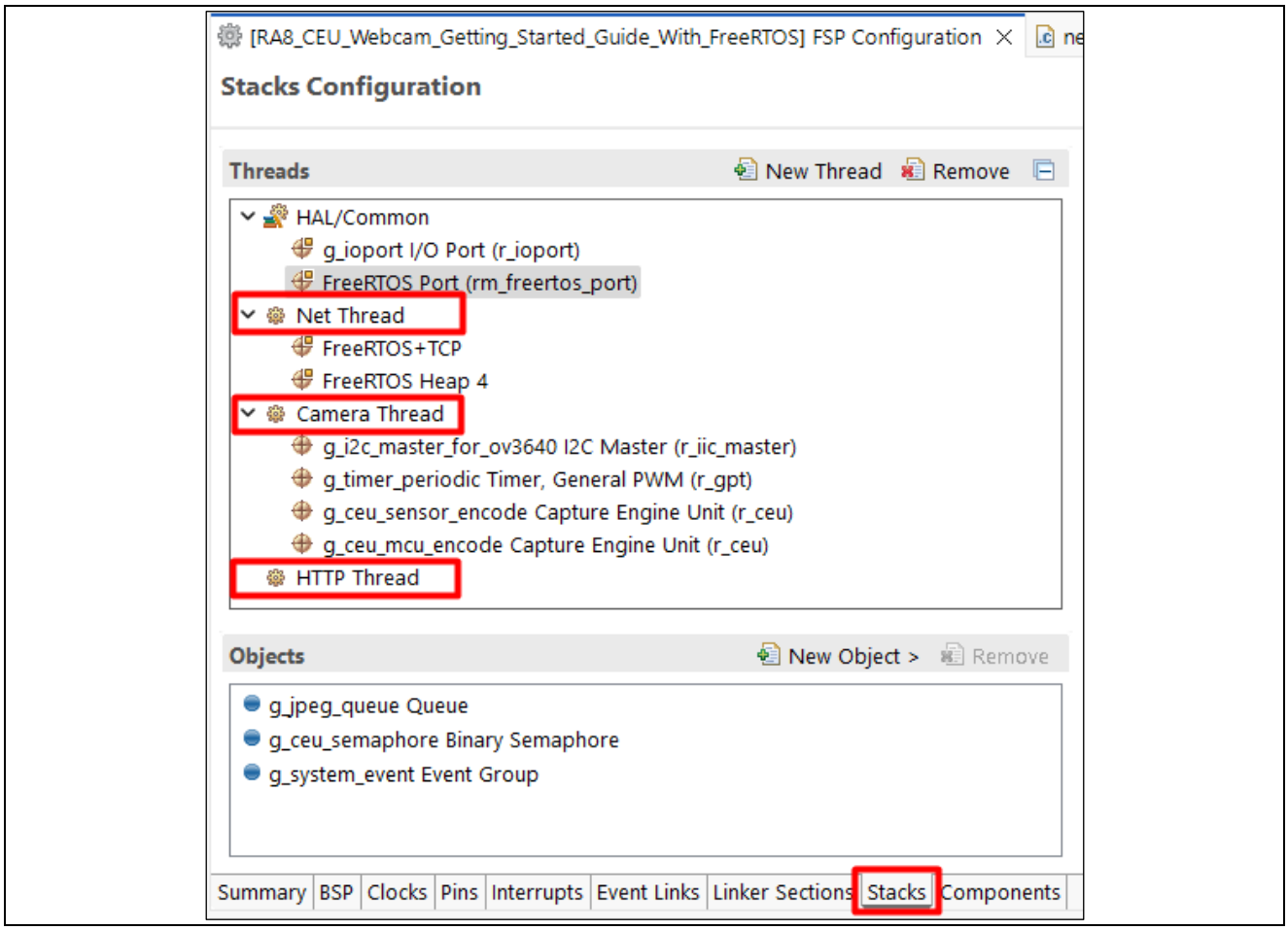

Figure 20. Application Thread Architecture

This application uses three dedicated threads, each responsible for a specific functional domain:

- Net Thread: Manages the TCP/IP stack (FreeRTOS+TCP), DHCP client operations, and overall network handling.
- Camera Thread: Configures the OV3640 image sensor, handles image capture functionality, and processes CEU interrupt events.
- HTTP Thread: Handles HTTP requests from the web browser and delivers JPEG image data for real-time streaming.

Table 1 CEU Webcam application thread properties

Task	Stack size	Priority	Rationale
Net Thread	1024	1	The TCP/IP stack thread is executed only once to obtain the IP address and is then deleted to reduce CPU workload.
HTTP Thread	4096	3	Handles large dynamic buffers for responses.
Camera Thread	4096	6	CEU ISR + sensor control. Highest priority for best performance

4.1.2 Inter-Task Synchronization and Data Flow

To ensure smooth coordination between threads, the CEU Webcam application uses event groups to signal important state transitions during system startup and runtime. Figure 21 illustrates the key events used throughout the system.

Network Initialization

When the system starts, the **Net Thread** is the first component to run. It initializes all resources required to bring up the network stack, including DHCP, TCP/IP services, and socket layers. After completing these tasks, the thread sets the **EVT_NET_READY** event, notifying the **HTTP Thread** that the networking layer is prepared.

HTTP Server Initialization

Upon receiving **EVT_NET_READY**, the **HTTP Thread** proceeds with initializing the web server environment. This includes creating the HTTP server instance, setting up sockets and inter-task. Once these operations are complete, it raises the **EVT_HTTP_READY** event to indicate that the web server is fully operational.
At this point, the system has completed most of the essential initialization steps required for webcam functionality, with only the final camera configuration remaining which occurs during runtime.

Camera Initialization and Runtime Configuration

Next, the **Camera Thread** begins its initialization routine. It configures the CEU module, prepares internal resources, and initializes the OV3640 image sensor.
Because the CEU Webcam application supports runtime switching between configurations, the Camera Thread interacts with the user to determine the desired encode method (Sensor Encode or MCU Encode). Based on the user's selection, it dynamically applies the appropriate configuration.

Runtime Events

```
/* Network Events */
#define EVT_NET_READY                (1U << 0)   /* Network initialized and IP obtained */

/* HTTP Server Events */
#define EVT_HTTP_READY               (1U << 1)   /* HTTP server successfully initialized */

/* Camera Events */
#define EVT_CAM_READY                (1U << 2)   /* Camera configured and ready to capture */
#define EVT_CAM_START                (1U << 3)   /* Start camera capture */
#define EVT_CAM_STOP                 (1U << 4)   /* Stop camera capture */
#define EVT_DEMO_END                 (1U << 5)   /* Request to terminate the demo */
```

Figure 21. Application Events Group

Once the camera is configured, the application is fully operational and users can access the webpage to view the live stream.
For runtime control and demonstration flow, the event group also defines several camera related events:

- EVT_CAM_READY
- EVT_CAM_START
- EVT_CAM_STOP
- EVT_DEMO_END

These events support transitions between states such as stopping the stream or ending the demonstration.

Additionally, users can create synchronization objects such as **Event Groups**, **Queues**, and **Semaphores** directly in the configuration.xml file, as shown in the figure below. By clicking New Object, users can easily add these RTOS components and view their detailed configuration options.

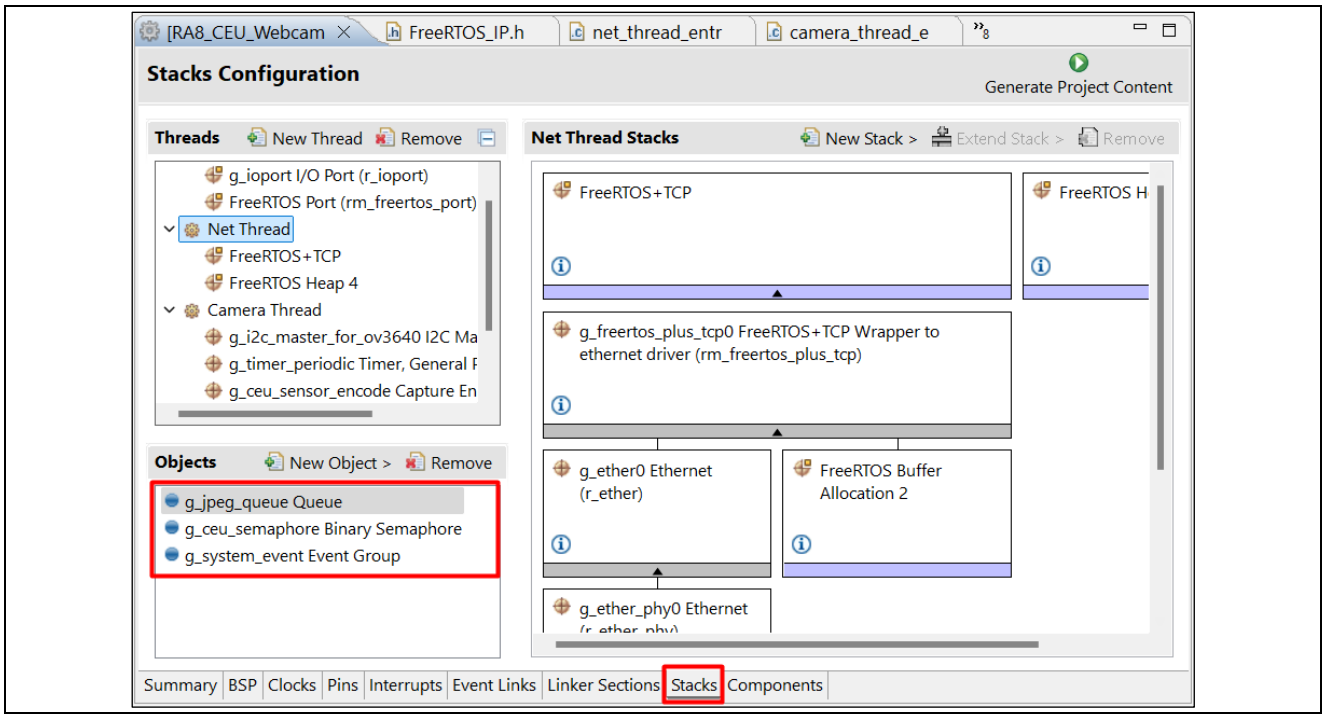

Figure 22. Application Object Creation

The application utilizes the **g_jpeg_queue** to manage the queue of captured image frames. This queue acts as the transfer mechanism between the Camera Thread and the HTTP Thread, ensuring that JPEG frames are delivered in the correct order for web streaming once a capture is completed.

The **g_ceu_semaphore** is used for a different purpose: it synchronizes the CEU hardware capture process. When the CEU finishes capturing an image, the CEU callback releases this semaphore. This mechanism allows the Camera Thread to know exactly when a frame is fully captured and when it is safe to begin the next capture operation. By doing so, the application prevents the newly captured data from being overwritten while the previous frame is still being processed.

For detailed handling logic, users can refer to the source code, where the interaction between event groups, the JPEG queue, and the CEU semaphore is fully demonstrated.

4.2 Thread Detail Configuration

4.2.1 Net Thread

4.2.1.1 FreeRTOS Plus TCP

FreeRTOS+TCP is a lightweight **TCP/IP stack** designed for use with **FreeRTOS**. In this application, it is integrated through the **rm_freertos_plus_tcp** wrapper, which provides the network interface required by the Ethernet drivers (**r_ether** or **r_rmac**).

For detailed configuration guidelines, refer to this link:

RA Flexible Software Package Documentation: FreeRTOS+TCP Wrapper to r_ether (rm_freertos_plus_tcp).

To use FreeRTOS+TCP, the system must provide heap memory for functions such as rand() and srand(). A minimum heap size of **4 KB (0x1000)** is recommended. This value can be configured in:

At the **"BSP"** tab → navigate to **"RA Common"** section → then configure **"Heap Size"**

Additionally, FreeRTOS+TCP requires **dynamic memory allocation**, so the property **"ConfigSUPPORT_DYNAMIC_ALLOCATION"** must be enabled in the FreeRTOS Memory Allocation settings for the Net Thread.

TCP Transmit Buffer Adjustment

The default TCP transmit buffer size is 3,000 bytes, which is significantly smaller than the size of each JPEG image frame (64 KB). A buffer that is too small forces the TCP stack to break the image into many fragments, increasing overhead and reducing throughput.

To improve transmission performance, the Tx buffer size should be increased to 65,536 bytes (64 KB).

With this configuration:

- Each image frame fits completely inside the TCP transmit buffer
- Fragmentation is minimized
- Protocol overhead is reduced
- Data flow becomes smoother and more stable

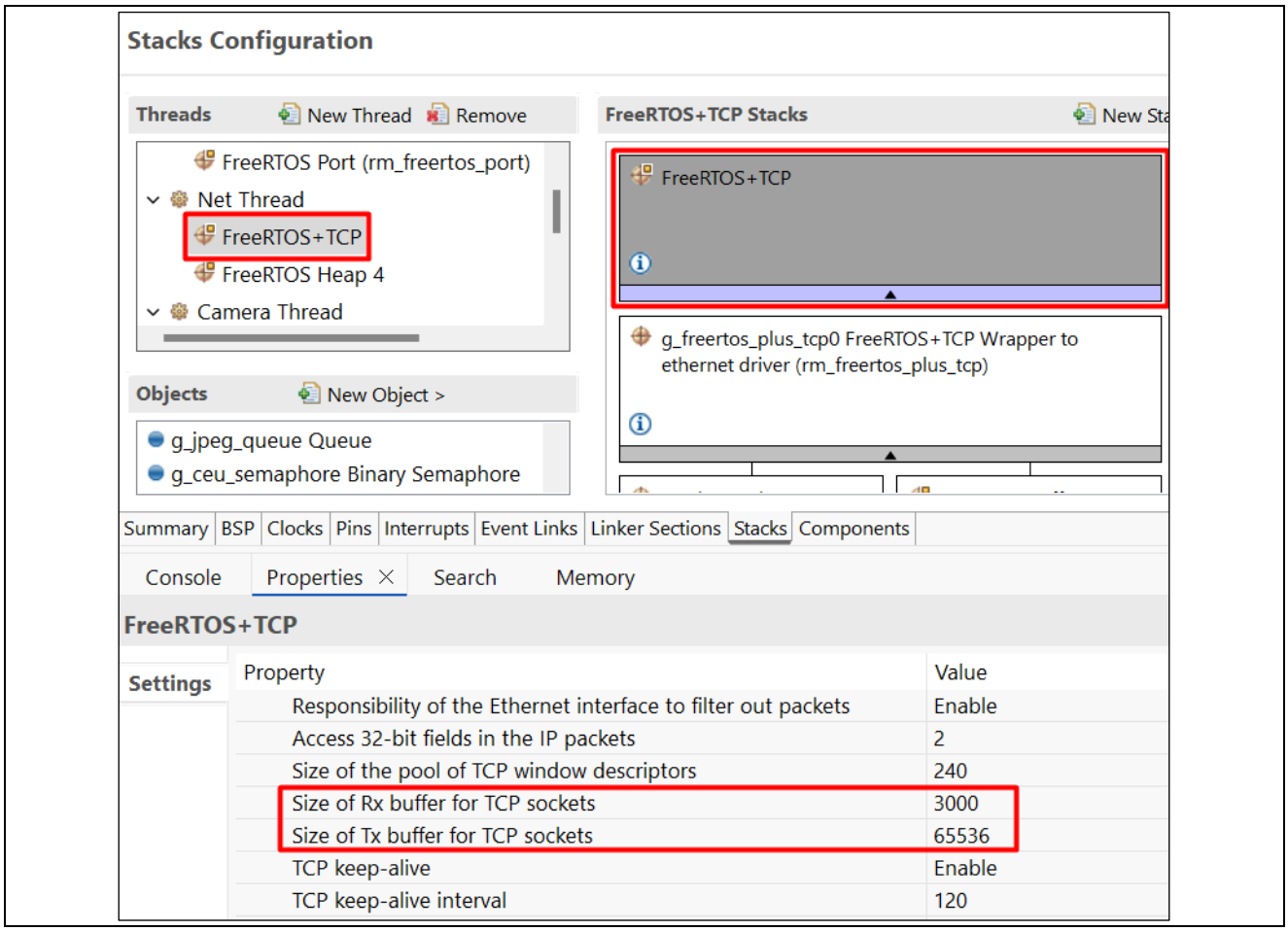

Figure 23. FreeRTOS+TCP Configuration

In summary, matching the TCP Tx buffer size to the JPEG frame size significantly improves overall data transmission efficiency.

4.2.1.2 FreeRTOS Heap 4

By default the FreeRTOS heap is declared by FreeRTOS and placed in memory by the linker. FreeRTOS provides five memory allocation schemes (heap_1 to heap_5), each offering different capabilities:

- heap_1: Simplest. memory cannot be freed
- heap_2: Allows freeing but does not merge adjacent free blocks
- heap_3: Wraps malloc()/free() for thread safety
- heap_4: Merges adjacent free blocks to reduce fragmentation; supports absolute address placement
- heap_5: Extends heap_4 with support for multiple non-contiguous memory regions

In this application, **Heap_4** is selected because it provides a balance between flexibility and fragmentation control.

FreeRTOS Plus TCP relies heavily on dynamic memory for resources such as:

- Network buffers
- TCP/UDP sockets
- Stream buffers
- Windowing and retransmission structures

Using Heap_4 ensures that these allocations remain efficient and minimizes memory fragmentation during long-running operation.

4.2.2 Camera Thread

The Camera Thread is responsible for controlling the OV3640 image sensor, initiating image captures, and storing the captured data into memory buffers. To perform these tasks, the thread relies on three key modules, each serving a specific purpose:

- I2C Master **(IC2)**: Provides APIs required to configure the OV3640 sensor registers. This includes setting image resolution, exposure settings, output format, and initializing the sensor during startup.
- General PWM Timer **(GPT)**: Generates the 24 MHz input clock required by the image sensor. Without this clock source, the sensor cannot operate or begin streaming pixel data.
- Capture Engine Unit **(CEU)**: Receives the pixel data output from the sensor and handles the hardware capture process. The CEU writes the captured image directly into memory buffers for further processing.

The CEU Webcam application supports two different encode methods, and each method uses its own set of sensors register configurations and a dedicated CEU instance:

- **Sensor Encode Method**: The sensor performs JPEG compression internally before sending data to the CEU.
- **MCU Encode Method**: The sensor outputs YUV image data, and JPEG compression is handled by the MCU.

Because these methods differ in data format, buffer requirements, and CEU operational settings, each encode method uses a dedicated configuration tailored to its capture workflow.

4.2.2.1 I2C Master

This module can be added from the Stacks tab by selecting:

Select **“New Stack”** → navigate to **“Connectivity”** section → then select **“I2C Master (r_iic_master)”**.

In this application, Channel 1 is used, and the corresponding data pins are assigned automatically.

According to the official OV3640 application note, the correct 7-bit I²C slave address of the sensor is 0x3C. However, some other OV3640 documents list the device address as 0x78 or 0x79, which can be misleading and may cause communication failures if used directly as the slave address.

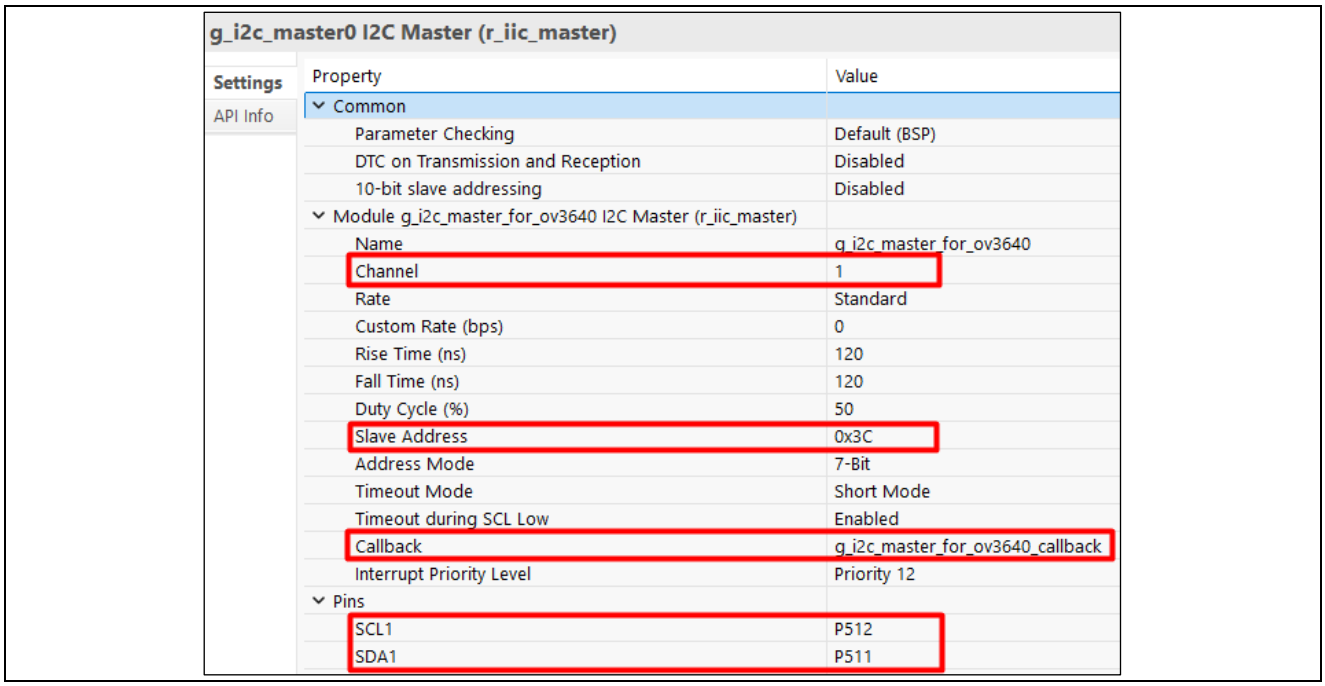

Figure 24. g_i2c_master_for_ov3640 configuration

This discrepancy comes from the way I²C addresses are represented. The transmitted I²C address byte consists of: **[7-bit slave address] [1-bit R/W]**

Based on this format:

- **0x78** represents the write address → **[0x3C << 1] | 0**
- **0x79** represents the read address → **[0x3C << 1] | 1**

Therefore, **0x78 and 0x79 are 8-bit address formats**, where the R/W bit is already included. When configuring an I²C peripheral on the MCU, the required value is the 7-bit address (0x3C). The I²C controller automatically appends the read/write bit during communication.

The I2C peripheral communicates with external devices using two pins. When configuring **Channel 1**, the RA Smart Configurator automatically assigns:

- **P511 → SDA1**
- **P512 → SCL1**

These pins must be enabled and configured correctly to ensure reliable communication with the OV3640 image sensor.

4.2.2.2 Timer General PWM

The GPT (General PWM Timer) module is used in this application to generate the external master clock (MCLK) required by the OV3640 image sensor. GPT is a multifunction timer capable of counting events, measuring external signals, generating periodic interruptions, or producing PWM waveforms at a GTIOC output pin.

Camera Clock Requirement

The OV3640 image sensor requires an external input clock within the range of 6 MHz to 27 MHz. This clock drives the sensor's internal PLL and pixel processing pipeline.

For this application, a 24 MHz clock is provided to the sensor because it is:

- A stable and commonly supported frequency
- High enough to ensure smooth pixel throughput
- Suitable for VGA capture with consistent performance
- Recommended by many reference designs to maintain stable frame timing

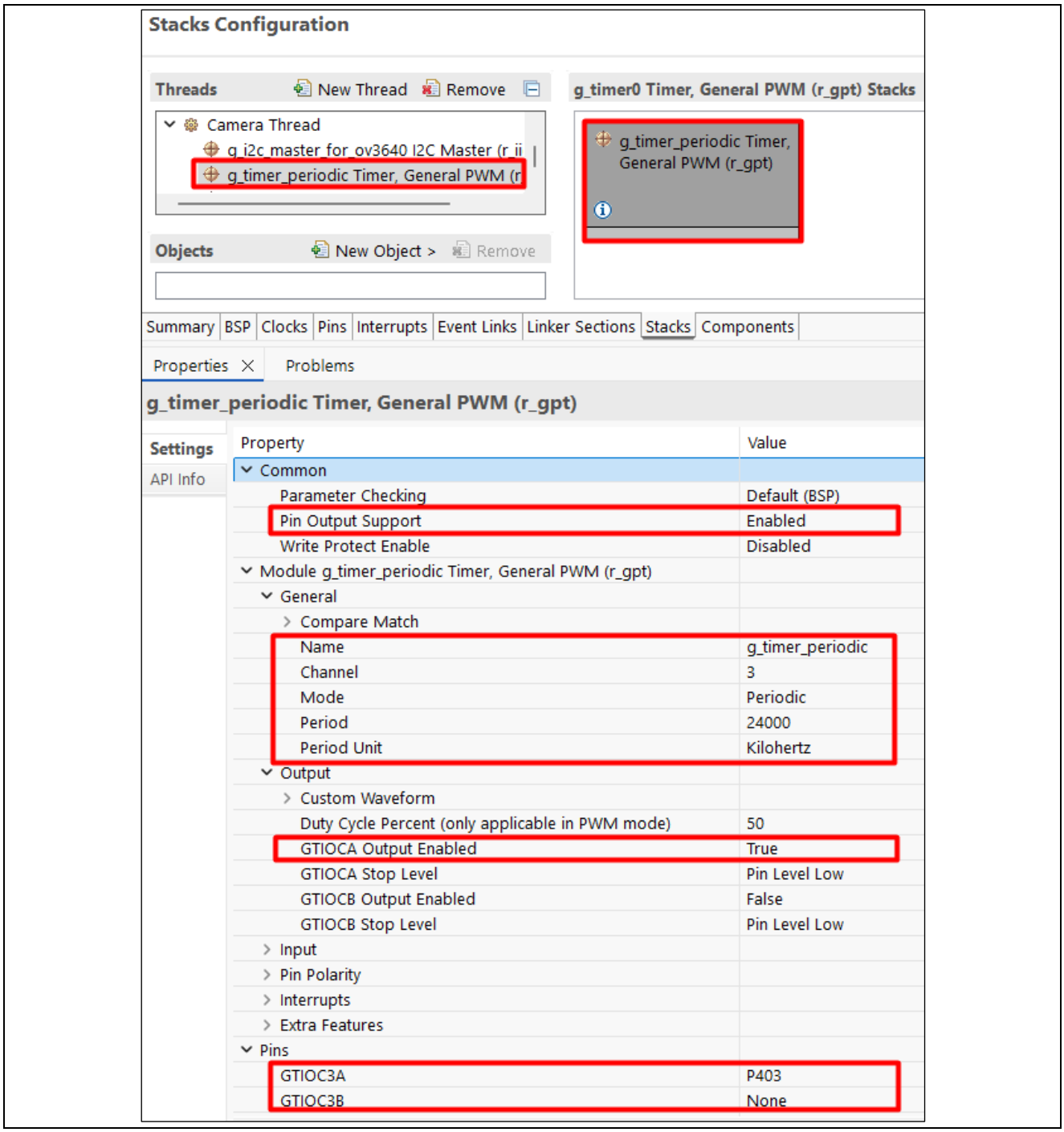

Figure 25. g_timer_periodic configuration

GPT as the Camera Clock Generator

The GPT module outputs a PWM waveform on a GTIOC pin. By configuring GPT in Periodic Mode with a suitable period value, the application can generate a precise 24 MHz clock used as the OV3640's input source.

In summary, the GPT module provides the essential 24 MHz clock signal required for the OV3640 image sensor. Proper configuration of the PWM output ensures stable camera operation, predictable timing, and reliable frame capture by the CEU.

4.2.2.3 Capture Engine Unit

The CEU peripheral is responsible for interfacing with the external camera module by receiving pixel data along with the associated timing signals. During operation, the CEU generates callback events to signal frame completion, vertical sync detection, or error conditions such as capture overruns or invalid line counts.

This application supports two encoded methods, each requiring a different CEU instance and configuration:

- MCU Encode Method
- Sensor Encode Method

Because these two methods use different data formats and capture flows, each requires a separate CEU configuration.

MCU Encode Method (YUV Capture)

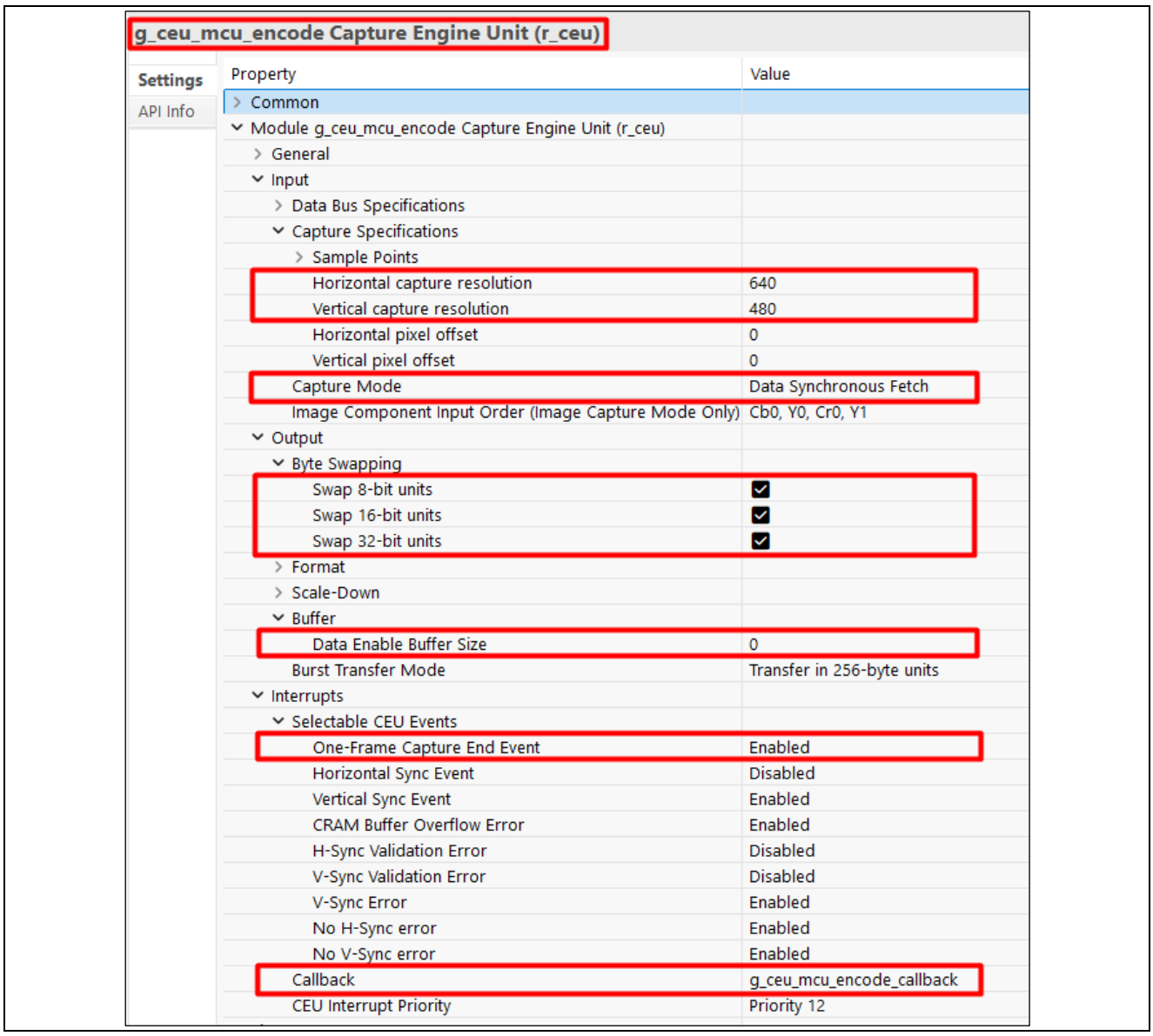

Figure 26. MCU encode method CEU instance configuration

In MCU Encode Method, OV3640 outputs image data in YUV format with a fixed resolution.

For this application:

- Resolution: VGA (640 × 480)
- Horizontal Capture: 640 pixels
- Vertical Capture: 480 pixels
- Output Format: YUV

The CEU operates in Data Synchronous Fetch Mode, where the capture size is calculated directly from:

Horizontal Resolution * Vertical Resolution * Bytes Per Pixel

The CEU captures the YUV frame into memory, and afterward the MCU performs JPEG encoding on the captured buffer. This mode therefore requires a large YUV buffer in SRAM (600 KB), along with a JPEG output buffer.

Sensor Encode Method (JPEG Capture)

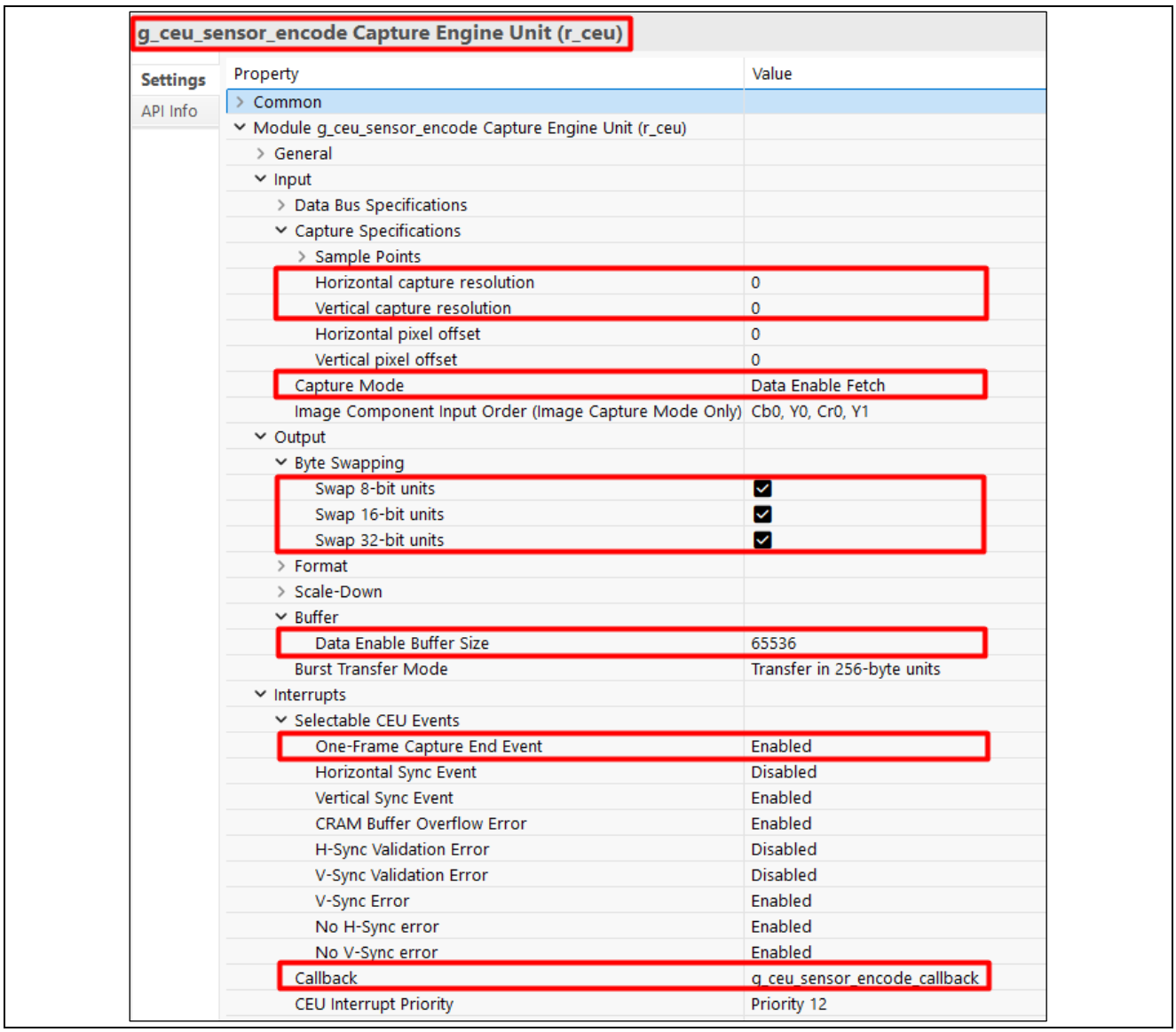

Figure 27. Sensor encode method CEU instance configuration

In Sensor Encode Method, the OV3640 performs JPEG compression internally and outputs a pre-compressed JPEG bitstream.

The CEU is configured differently because the incoming data is not pixel-aligned, but rather a binary JPEG stream.

Key characteristics of this method:

- Output Format: JPEG compressed
- Expected Size: less than 64 KB
- CEU operates in Data Enable Fetch Mode

In this method, the CEU does not use horizontal or vertical resolution values. Instead:

- Capture boundaries are based on VD rising → VD falling (frame duration)
- Maximum capture range is defined using Data Enable Buffer Size, which sets the upper memory limit (firewall address)
- The external camera module must send JPEG data in 4-byte aligned units

This provides a continuous, variable-length data stream suitable for JPEG capture.

CEU Interrupts and Frame Completion

The CEU generates several types of interruption events, including:

- V-Sync event
- Frame End event
- Error events (overrun, invalid line count, FIFO full, etc.)

In this application, the Frame End event is used to determine when a full frame has been captured. The callback releases a semaphore, allowing the Camera Thread to safely begin a new capture without risking memory overwrites.

Support for 8-bit Camera Bus and Resolution Range

The project uses an 8-bit parallel camera interface and supports image capture sizes up to 3 MP (within the CEU's maximum capability of approximately 5 MP).

JPEG mode is especially efficient because frame size is automatically delimited by the JPEG bitstream, eliminating the need to define resolution parameters in the CEU configuration.

For JPEG capture, the OV3640 inserts a JPEG end-of-image marker (0xFFD9), enabling the system to recognize the end of each frame without relying on horizontal/vertical capture settings.

4.2.3 HTTP Thread

The HTTP Thread does not include additional FSP stacks. Instead, it orchestrates the web server lifecycle and manages two internal FreeRTOS tasks to serve the web UI and stream MJPEG frames to a single client at a time.

Internal tasks it manages:

- http_client_handler_task(): parses HTTP requests from the browser and serves HTML or MJPEG data.
- stream_control_task(): reacts to system events to start/stop streaming and to terminate the demo cleanly.

HTTP Client handle Task

Inside http_client_handler_task() the first request line drives routing:

- GET /index.html → send_html_response()

  Returns the embedded HTML UI (index_html) with Connection: keep-alive.
- GET /mjpeg stream → MJPEG pipeline

  Sends JPEG image via stream pipeline
- GET /favicon.ico → returns 404, closes connection.
- Others → returns 404 text, closes connection.

Note: The JPEG queue is the bridge between Camera Thread (producer) and HTTP Thread (consumer). Camera Thread ensures frame integrity and enqueues (address, size) pairs only after capture completion (signaled by CEU callbacks).

Stream control Task

stream_control_task() centralizes runtime control:

- **EVT_DEMO_END**
  Sets **stream_allowed = false**, **server_running = false** → shuts down active streaming and the listening loop.
- **EVT_CAM_STOP**
  Temporarily pauses streaming: sets **stream_allowed = false**, then waits for **EVT_CAM_READY** to be raised again before the next client session is allowed to proceed.

This design cleanly separates control-plane events from the data-plane activities in the client handler.

In summary, the HTTP Thread exposes a single client HTTP server, starts/stops streaming via events, and continuously pulls JPEG frames from **g_jpeg_queue** to deliver an MJPEG stream over a persistent multipart connection.

4.3 Pin Configuration

To enable proper operation of the CEU peripheral with the OV3640 image sensor, the necessary CEU-related pins must be configured in the Pins tab. However, several pins required by Ethernet B conflict with the CEU pin set. Because of this hardware limitation, the application must switch from Ethernet B to Ethernet A to avoid pin contention.

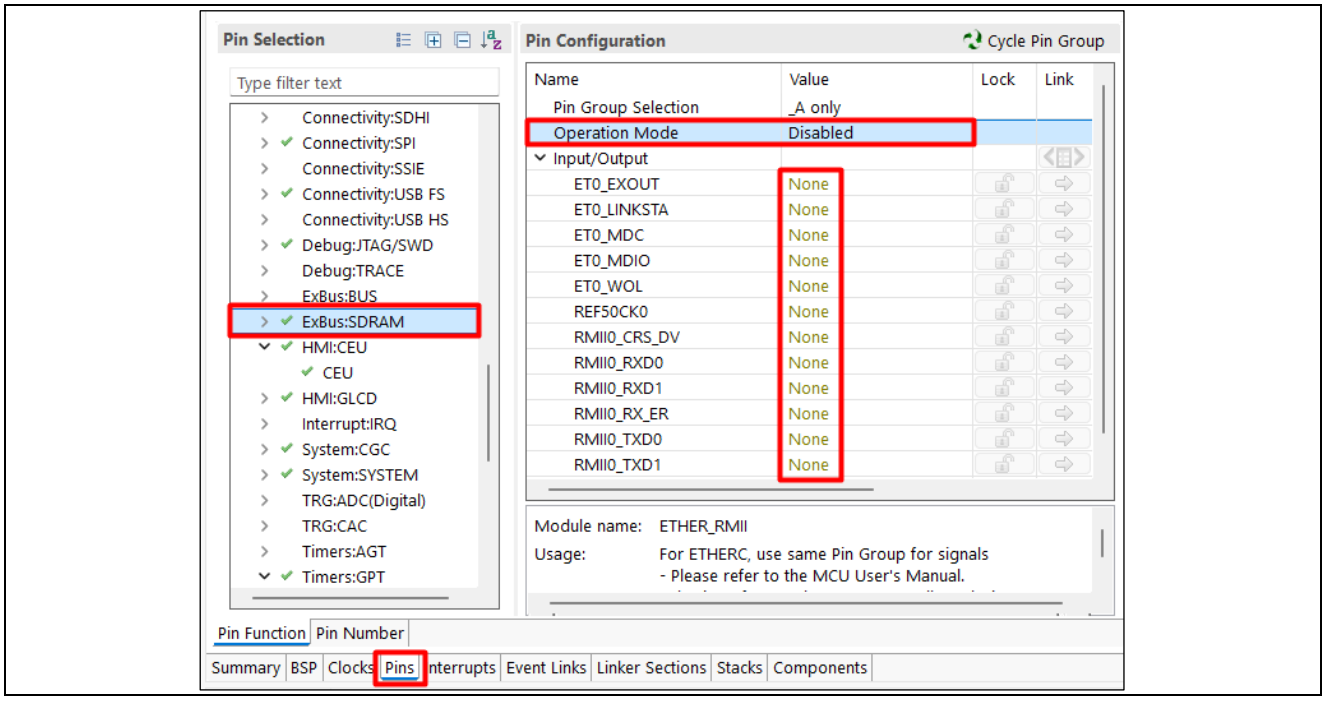

Figure 28. Disable SDRAM Pin Configuration

Because the pins of Ethernet B overlap with the signals required by the CEU, Ethernet B cannot be used. The application therefore switches to Ethernet A; however, its pins conflict with those of the SDRAM, requiring SDRAM to be disabled.

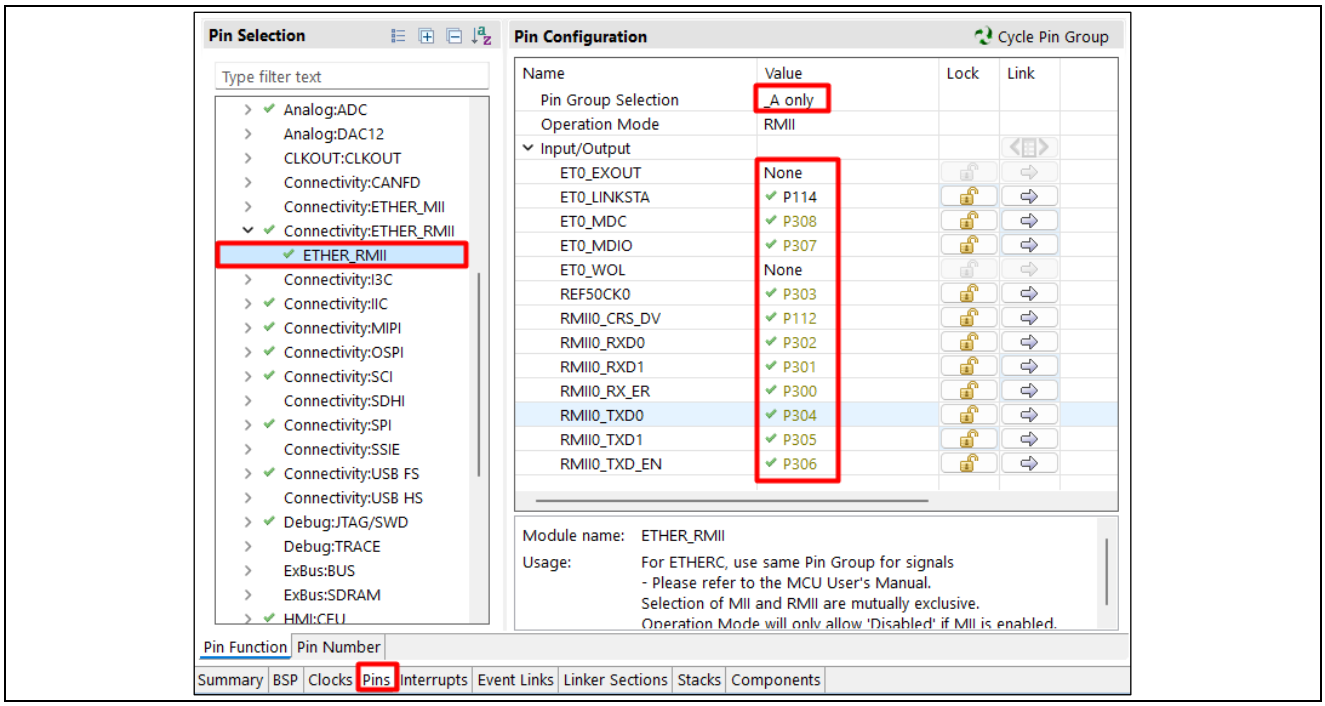

Figure 29. Ethernet Pin Configuration

Figure 29 show how to configure Ethernet pin. Next, configure for CEU pin as shown in Figure 30.

With SDRAM turned off, the full CEU pin group can be configured without conflict. The CEU interface uses an 8-bit data bus along with clock, Hsync, and Vsync signals, all assigned in the Pins tab. This configuration ensures the correct hardware interface between the RA8D1 CEU peripheral and the OV3640 camera.

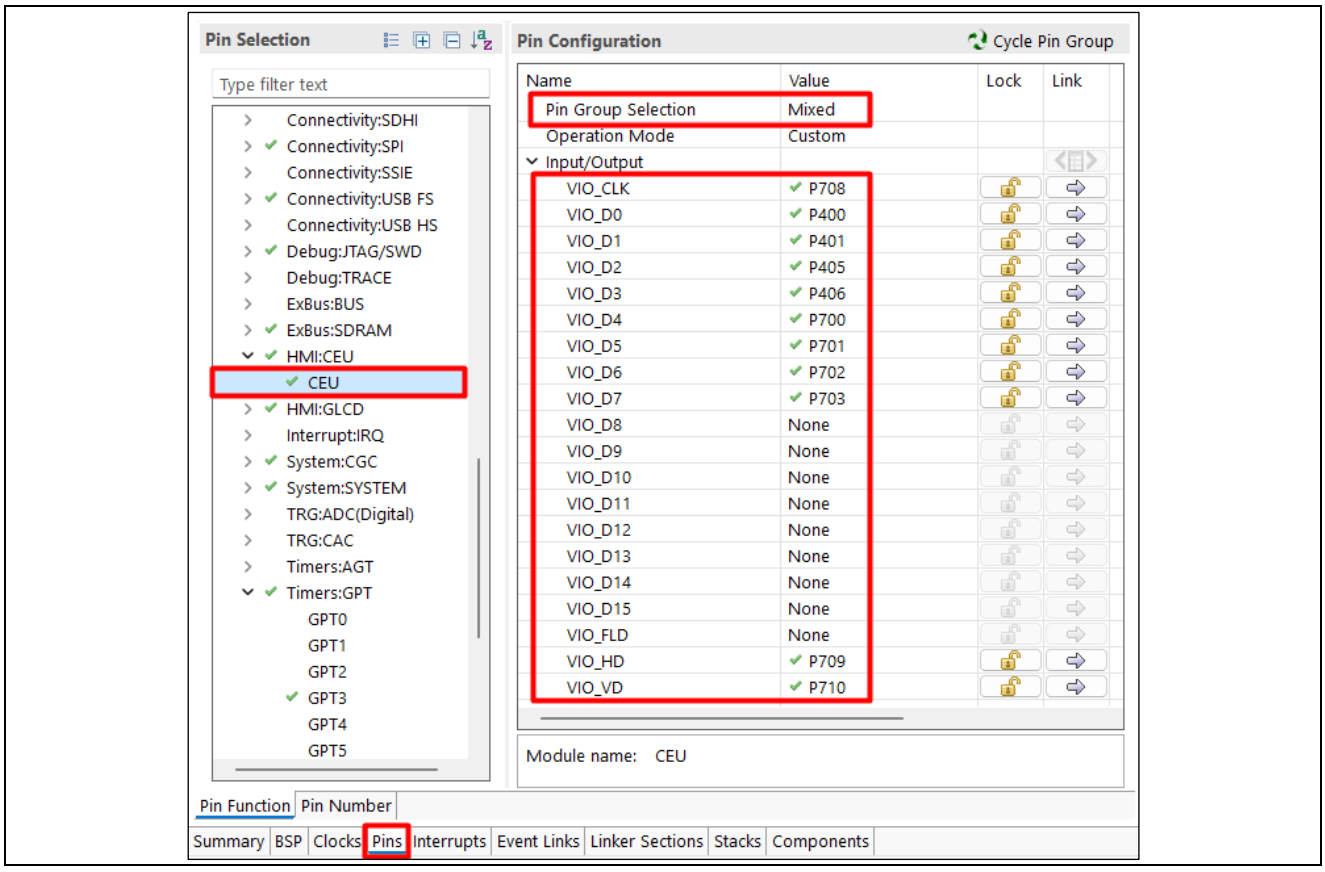

Figure 30. CEU Pin Configuration

4.4 Memory Management

4.4.1 FreeRTOS Memory Allocation

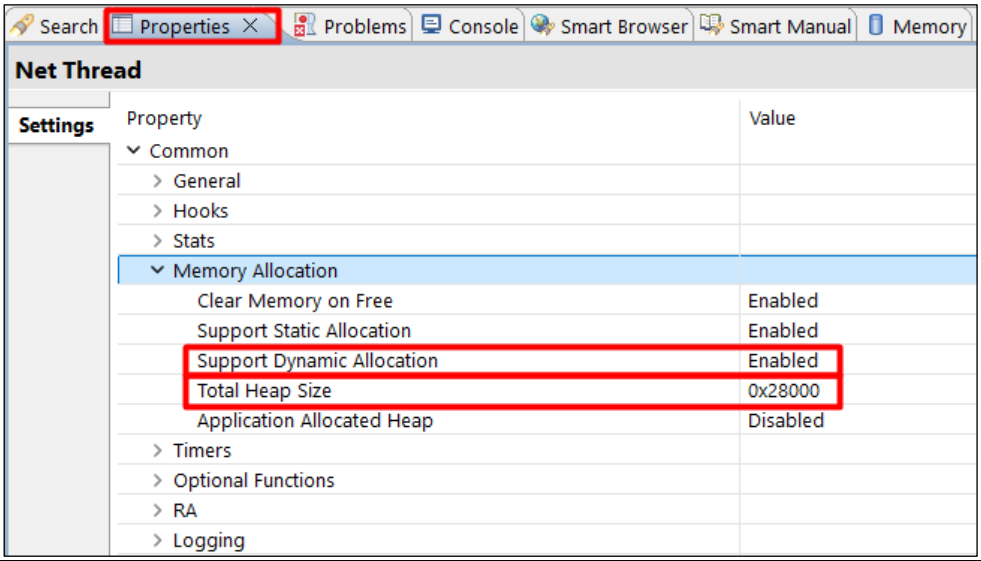

Figure 31. FreeRTOS Memory Allocation

"Total Heap Size" define the total amount of RAM available in the FreeRTOS heap. This value will only be used if Support Dynamic Allocation is Enabled and the application makes use of one of the sample memory allocation schemes provided in the FreeRTOS source code.

4.4.2 SRAM Memory Allocation

As mentioned earlier, when using both the camera and Ethernet on this board, one limitation is that SDRAM cannot be used. As a result, the application must rely entirely on SRAM, and the available SRAM on the RA8D1 is only 896 KB. Therefore, memory must be allocated and managed carefully to ensure efficient operation.

The application supports two encoding methods.

- The Sensor Encode Method requires only a single memory region for the JPEG buffer.
- The MCU Encode Method requires an additional buffer for the YUV data.

Based on calculations, a YUV buffer at VGA resolution consumes approximately 600 KB, while the JPEG buffer is optimally allocated at 64 KB. This results in a combined memory usage of about 664 KB for both buffers.

The remaining SRAM is reserved for tasks, stack, and source code execution, and is dynamically allocated through FreeRTOS memory management.

5. Comparison of the JPEG Encoding Methods

This section compares two JPEG generation approaches supported by the CEU Webcam application: **Sensor JPEG Encoding method** and **MCU JPEG Encoding method**. Each method has unique advantages and trade-offs, and the optimal choice depends on the user's performance requirements, memory constraints, and expected image quality.

The CEU Webcam application uses VGA resolution by default, it's a relatively large frame size that requires significant memory bandwidth and processing time. In certain applications, users may choose lower resolutions to reduce RAM usage and increase frame rate while still maintaining flexibility in image quality.

When deciding between the two encoding methods, users must consider whether they prioritize:

- Frame rate performance
- Image quality and control
- Memory footprint
- Configuration complexity

The following table breaks down the strengths and limitations of each approach.

Table 2 Comparison of the JPEG Encoding Methods

Category	MCU Encode Method	Sensor Encode Method
Performance	Low frame rate (~5 fps); requires ~75 ms for YUV transfer + ~135 ms for encoding	High frame rate (up to 19 fps); JPEG created by sensor; ~55 ms transfer time
Image Quality Control	Adjustable compression level; overall better and tunable image quality	Compression fixed by sensor; no quality control options
Memory footprint	600 KB for YUV + 64 KB for JPEG	Only 64 KB needed for JPEG
Configuration Complexity	Simple sensor configuration for YUV mode	More complex sensor register configuration required for JPEG mode

6. Verifying MVE with LLVM optimization

This section guides users on how to determine whether an application has applied MVE optimizations. On the RA8D1 board, the LLVM toolchain provides an optimization feature, and users can refer to High Performance with RA8 MCU using Arm® Cortex®-M85 core with Helium™ for more detailed information.

To begin, start debugging the application as usual and set a breakpoint at the function the user wants to verify.

For example, let verify **“encode_yuv_to_jpeg()”** function.

Connect a terminal, run the application, select option **“1”**, and access the webpage to reach the breakpoint.

At this point, open the **“Disassembly”** window. In the top-left corner of the e² studio, select **“Window”** → then choose **“Show View”** → and finally select **“Disassembly”**. The Disassembly view will be shown as Figure 32.

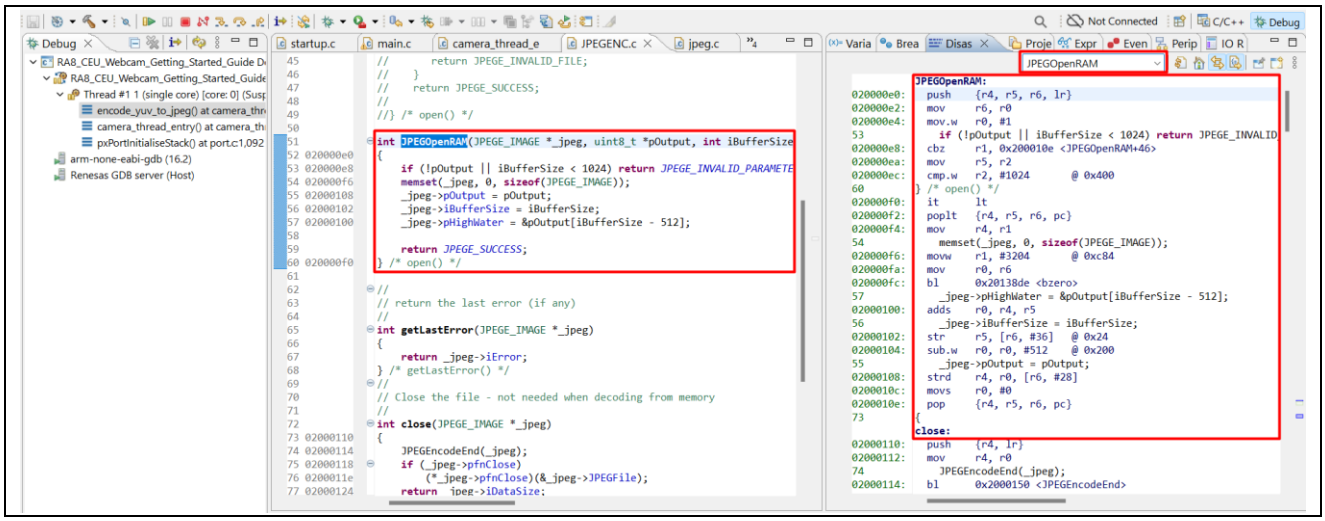

Figure 32. Verify MVE applied in JPEGOpenRAM function.

In the **“JPEGOpenRAM()”** function, no vectorized instructions are present, indicating that MVE optimizations were not applied.

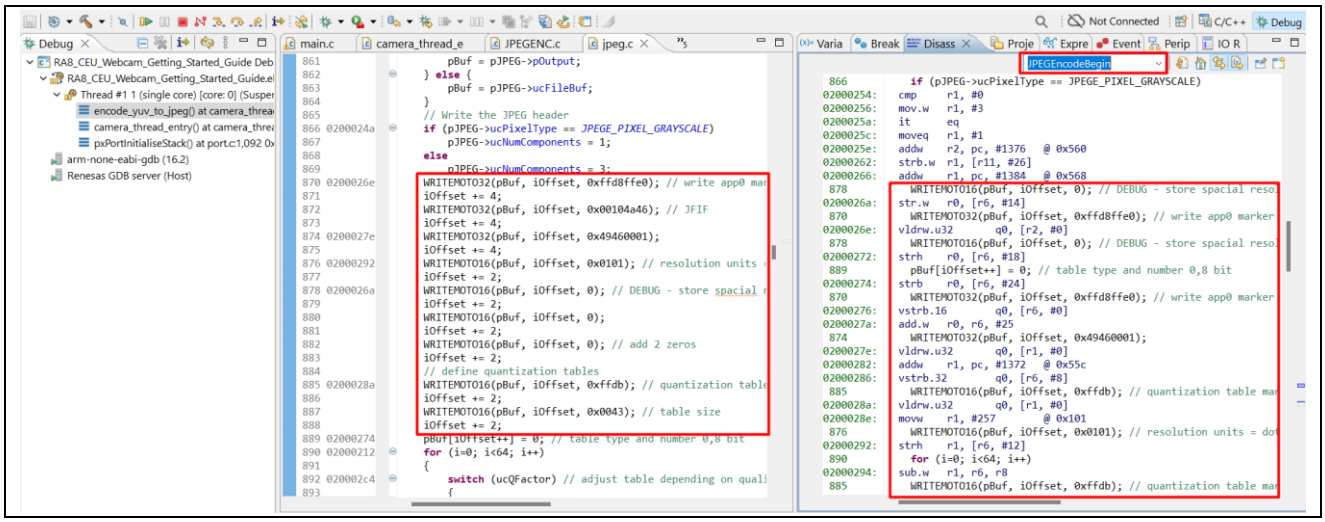

Figure 33. Verify MVE applied in JPEGEncodeBegin function.

In Figure 33, shows that the **"JPEGEncodeBegin()"** function does utilize MVE, demonstrating successful optimization.

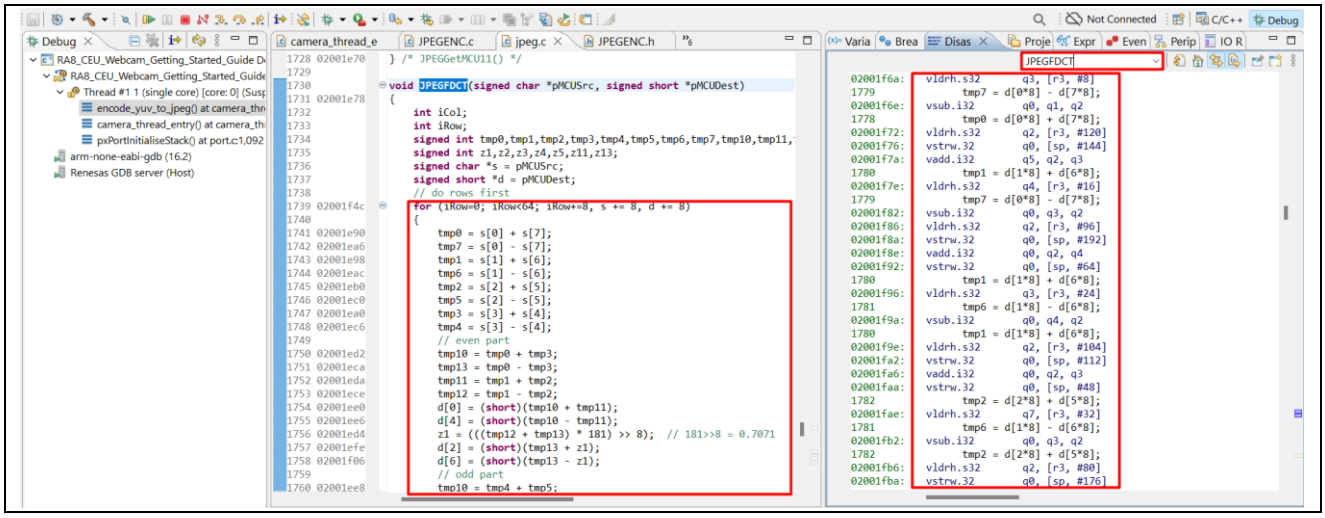

Figure 34. Verify MVE applied in JPEGFDTC function.

Next, examining functions deeper within **"addFrame()"** reveals that the subsequent routines do contain vectorized instructions in the disassembly output, confirming that they were compiled with MVE optimizations enabled.

In these subsequent functions, vectorized instructions are present in the disassembly code, confirming that they were compiled with MVE optimization enabled.

Note: Vectorized instructions are instructions that have "v" character at the beginning of each word.

7. OV3640 Sensor Register Configuration

7.1 OV3640 Camera

The OV3640 Camera Chip™ is a compact, high-performance 1/4-inch 3.1-megapixel CMOS image sensor designed using OmniPixel3™ technology. It delivers full QXGA resolution (2048×1536) and supports multiple output formats, including full-frame, sub-sampled, windowed, or scaled 8-bit/10-bit images. The sensor operates at up to 15 frames per second in QXGA mode and offers complete user control over image quality, formatting, and data transfer through interfaces such as SCCB, MIPI, or its embedded microcontroller.

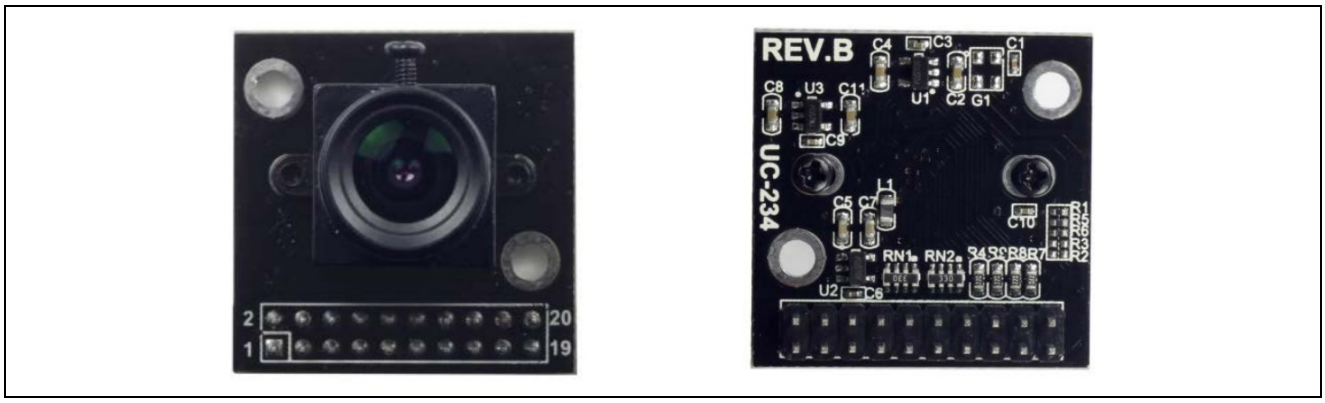

Figure 35. OV3640 camera

The OV3640 camera supports a variety of output formats, including RAW RGB, RGB656/555/444, YUV422/420, YCnCr422, and JPEG compression. It also offers built-in data compression and provides a digital video port (DVP) for parallel output.

Supports QXGA (2048x1536) and any smaller sizes, with a maximum transfer rate of 15fps for QXGA and any size scaling down from QXGA. For XGA (1024x768) and any size scaling down from XGA, the maximum rate is 30fps.

The RA8D1 board manages the OV3640 camera by controlling its power mode and reset through GPIO, configuring sensor registers and essential settings via the I2C interface, generating the required clock source using the GPT module, and capturing image data along with synchronization signals (Pixel Clock, VSYNC, HREF) through the CEU, then the captured image is stored as a JPEG frame in the SRAM buffer.

Once camera activated, it continuously acquires frames and processes them into JPEG format suitable for web transmission. These JPEG frames are then relayed to the HTTP server, which manages timely delivery to connected clients for seamless real-time viewing. This approach ensures minimal latency between image capture and display, enhancing the user experience during live demonstrations.

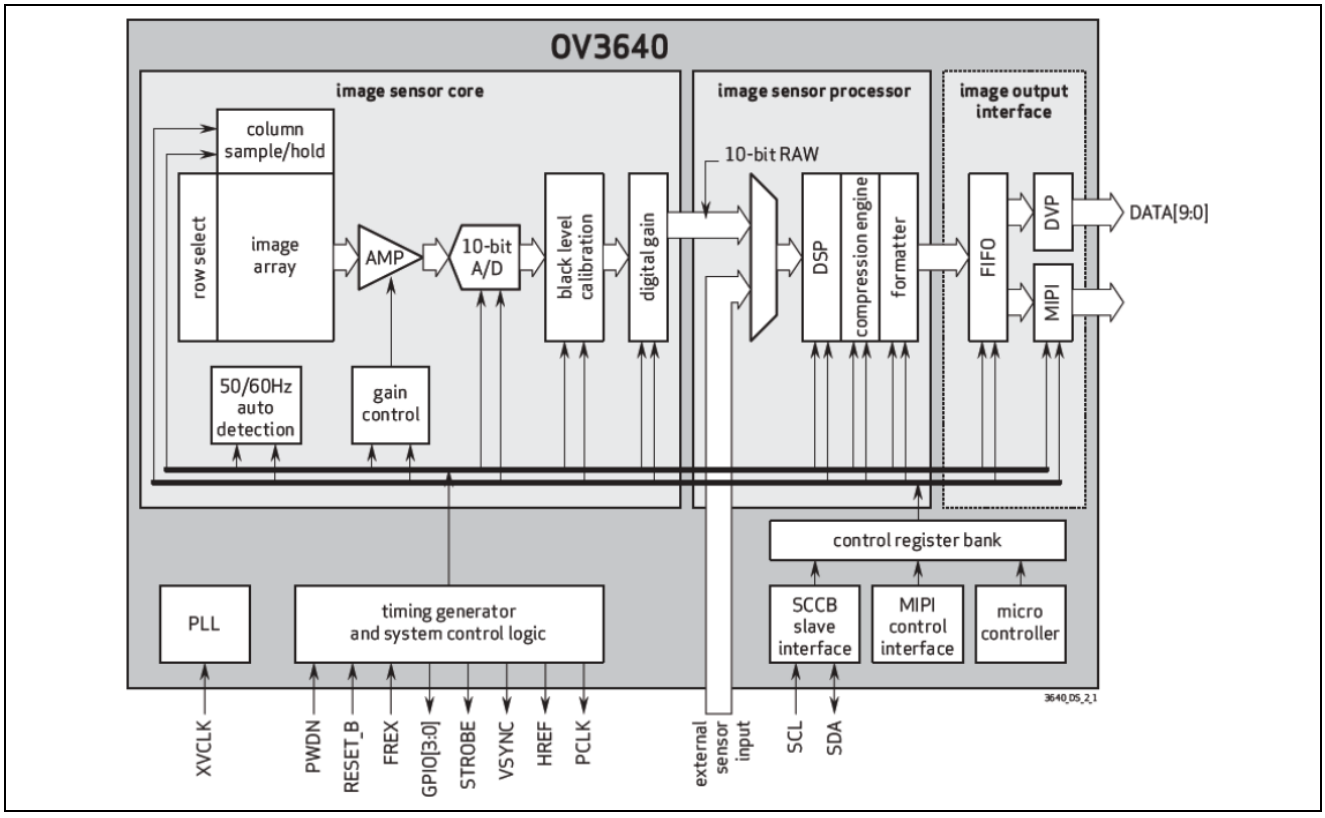

Figure 36. OV3640 block diagram

Referencing the OV3640 block diagram shown in Figure 36, the image processing workflow begins at the image sensor core, where the image array captures incoming light and sends the resulting signals through amplification and a 10-bit analog-to-digital converter. These signals are then calibrated for black level and adjusted with digital gain before being processed by the image sensor processor. Within this processor, tasks such as image enhancement are performed using a DSP and, if needed, the optional compression engine can be used. The final processed data is buffered and output via interfaces like DVP or MIPI. Throughout this sequence, all operations are coordinated by the register bank and microcontroller, which utilize SCCB or MIPI commands for system control.

7.2 Camera Register configuration

In this application, two image output formats must be configured for the camera to interface correctly with the CEU. Detailed configuration for each output format can be found in the /src/ov3640.c.

For the YUV output format, users can directly refer to the **"Reference Settings"** section in the OV3640 application notes.

For the JPEG output format, the camera must be configured to enable its compression mode. Several additional parameters related to this mode also need to be adjusted to prevent operational errors. Users may refer to the OV3640 datasheet for a complete description of the required configuration steps.

Note: The following sections intentionally address only a subset of register configurations that are considered critical to the functionality of the application. While additional registers may be configured differently from their default values in accordance with the device documentation, these configurations have not been exhaustively evaluated and are not guaranteed to be strictly required. Users who intend to implement a customized or fully optimized solution should refer to the official datasheet and conduct further investigation into the remaining register settings.

7.2.1 IO Control

The OV3640 is a flexible camera sensor that can work in different system roles:

- Output image data to an MCU / FPGA (normal camera mode)
- Share a bus with another camera
- Accept external sync, strobe, or control signals
- Use GPIOs for autofocus, flash, or triggers

Because of this, many pins are bidirectional, so user must explicitly define whether these pins are driving a signal or receiving a signal.

Table 3 IO Control Register

Register	Default Value	Application Value	Description
0x30b0	0xff	0xff	IO Control 0 CY[7:0]
0x30b1	0xef	0xff	IO Control 1 C_GP[1:0], C_VSYNC, C_STROBE, C_PCLK, C_HREF, CY[9:8]
0x30b2	0x00	0x10	IO Control 2 GPO_monitor, C_FREX, R_PAD[3:0]

Practical meaning of Application Value shown in Table 3:

- When the OV3640 outputs image pixels over the **DVP interface**, DATA pins **must be outputs**
- If configured as input by mistake, **no image data will reach the MCU**

7.2.2 Compression Related Register

To output image data in JPEG (compressed) format, the OV3640 must be configured to enable its internal compression engine and define the compressed image width. This is done through a small set of compression-related registers described below.

Table 4 Compression Related Register

Register	Default Value	Application Value	Description
0x3100	0x00	0x32	System Control 0 Bit[4]: Compression enable 0: Compression mode disabled 1: Compression mode enabled
0x3610	0x40	0x00	DVP Control 10 Bit[7:0]: Width_man[7:0]/4 Manual output image width
0x3611	0x0C	0x20	DVP Control 11 Bit[7:5]: Width_man[10:8] Width_man_real=width_man*4

Register 0x3100 (System Control 0) is used to enable or disable JPEG compression. When this bit is set, the image data produced by the ISP is passed through the internal JPEG compression engine before being output.

Registers 0x3610 (DVP Control 10) and 0x3611 (DVP Control 11) are used to define the output image width when compression mode is enabled. Width manual defines the internal line width used by the JPEG encoder and DVP output, not the logical image resolution. A larger value (such as 1024 pixels) is commonly used to ensure stable JPEG output, even when the actual image width is smaller.

7.2.3 DSP Control Register

The OV3640 integrates an internal image processing pipeline (often referred to as the DSP/ISP) that performs operations such as windowing (cropping), scaling, and format processing on the sensor data before it is output to the host.

To obtain the desired output resolution, the OV3640 uses a two-stage scaling path followed by a final ISP output size setting:

1. Scaling input size: Defines the width and height used as the input to the ISP scaling block. This size must be equal to or smaller than the sensor window selected by the windowing registers
2. Zoom-out (scaling output) size: Defines the size produced by the scaling block. This size must be equal to or smaller than the scaling input size.
3. Final ISP output size: Defines the final image size delivered to the host (preview/capture output). This size must be equal to or smaller than the zoom-out size.

Note: To ensure stable image output, the internal camera processing resolution is configured slightly larger than the final output size. This allows the image processing pipeline to operate with proper internal alignment.

Table 5 Compression Related Register

Register	Default Value	Application Value	Description
0x335f	0x68	0x68	**SIZE_IN_MISC** Bit[7]: Reserved Bit[6:4]: Vsize_in[10:8] Bit[3:0]: Hsize_in[11:8]
0x3360	0x18	0x18	**HSIZE_IN_L** Hsize_In[7:0]
0x3361	0x0c	0x0c	**VSIZE_IN_L** Vsize_In[7:0]
0x3362	0x68	0x12	**SIZE_OUT_MISC** Bit[7]: Reserved Bit[6:4]: Vsize_Out[10:8] for zoom_out Bit[3:0]: Hsize_Out[11:8] for zoom_out

0x3363	0x18	0x88	**HSIZE_OUT_L** Hsize_Out[7:0] for zoom_out
0x3364	0x0C	0xe4	**VSIZE_OUT_L** Vsize_Out[7:0] for zoom_out
0x3088	0x80	0x02	**ISP_XOUT[15:8]** ISP X-direction Output Size [15:8] Bit[7:4]: Not used Bit[3:0]: X_size_in[11:8]
0x3089	0x00	0x80	**ISP_XOUT[7:0]** ISP X-direction Output Size [7:0]
0x308a	0x06	0x01	**ISP_YOUT[15:8]** ISP Y-direction Output Size [15:8] Bit[7:3]: Not used Bit[2:0]: Y_size_in[10:8]
0x308b	0x00	0xe0	**ISP_YOUT[7:0]** ISP Y-direction Output Size [7:0]

Example: VGA (640 × 480) output configuration

1. Scaling input size (0x335F–0x3361)

   From the register description, the default scaling input values decode as:

   - Hsize_in = 0x818 = 2072 pixels
   - Vsize_in = 0x60C = 1548 pixels

2. Zoom-out (scaling output) size (0x3362–0x3364)

   To support a VGA output while maintaining the constraint ISP output ≤ zoom-out size, the zoom-out size is set slightly larger than 640 × 480:

   - Hsize_out = 648 = 0x288
     - Hsize_out[11:8] = 0x2 (in SIZE_OUT_MISC)
     - Hsize_out[7:0] = 0x88 (HSIZE_OUT_L)
   - Vsize_out = 484 = 0x1E4
     - Vsize_out[10:8] = 0x1 (in SIZE_OUT_MISC)
     - Vsize_out[7:0] = 0xE4 (VSIZE_OUT_L)

   This ensures the final output size can be configured cleanly while staying within the device's scaling rules.

3. Final ISP output size (0x3088–0x308B)

   Finally, the ISP output is set to the standard VGA size:

   - ISP_XOUT = 640 = 0x280
     - ISP_XOUT[15:8] = 0x02 (0x3088)
     - ISP_XOUT[7:0] = 0x80 (0x3089)
   - ISP_YOUT = 480 = 0x1E0
     - ISP_YOUT[15:8] = 0x01 (0x308A)
     - ISP_YOUT[7:0] = 0xE0 (0x308B)

With the above settings, the OV3640 outputs VGA frames at 640 × 480 to the host.

8. Application Limitations

8.1 Single-Client connection only

RA8 CEU Webcam is intentionally designed with a single-client connection model to simplify the architecture and highlight the core camera and networking features of the RA8 MCU.

For users intending to develop a scalable or production-ready solution, it is recommended to redesign the application architecture by decoupling camera capture from client request handling, introducing shared frame buffering, and implementing proper synchronization mechanisms.

Alternatively, the RA8 device can be positioned as an image acquisition endpoint, streaming captured frames to an external gateway or host system that handles multi-client distribution and higher-level networking requirements.

8.2 MCU SDRAM Is Disabled When using the CEU

As mentioned in section 4.3, the MCU SDRAM is disabled when the Camera Engine Unit (CEU) is enabled. As a result, all image buffers and application data must reside in internal SRAM.

This design choice simplifies memory management and avoids resource conflicts, making it suitable for demonstration and getting-started purposes. However, it limits the ability to buffer large images or multiple frames and restricts scalability for memory-intensive or multi-client applications.

Users intending to develop production-ready systems are advised to adopt streaming-oriented architecture or offload buffering and advanced processing to an external host or gateway.

In the current implementation, when operating in **MCU JPEG encode mode**, the image compression quality is configured to **Medium**, which results in a relatively stable and predictable output image size. Under this configuration, the encoded JPEG data typically fits within the allocated **64 KB JPEG output buffer**.

However, if the compression quality is increased to **High** or **Best**, the resulting JPEG image size may vary significantly depending on the captured scene characteristics, such as very bright or very dark environments. In such cases, the encoded JPEG data may exceed the currently allocated buffer size.

If the JPEG output size exceeds the allocated buffer, image corruption or frame buffer overflow may occur, which can directly impact the stability and correct operation of the application.

9. References

RA8D1 MCU Group Hardware User’s Manual

Renesas RA FSP User’s manual (v6.4.0)

OV3640 Software Application Notes

OV3640 Datasheet

High Performance with RA8 MCU using Arm® Cortex®-M85 core with Helium™

Appendix 1. Camera working principle

A CMOS image sensor captures light by converting incoming photons into electrical signals using an Active-Pixel Sensor (APS) design. Each pixel contains a photodiode that absorbs light and generates an electrical current proportional to the brightness and color intensity. This current is then amplified by transistors integrated directly into each pixel.

A CMOS image sensor is structured as a 2×2 repeating Bayer pattern consisting of four photodetectors:

- One Red (R)
- One Blue (B)
- Two Green (G)

Green is duplicated because the human eye is most sensitive to green light, which improves overall luminance accuracy. In the Bayer pattern, odd rows alternate between Blue–Green, while even rows alternate between Green–Red.

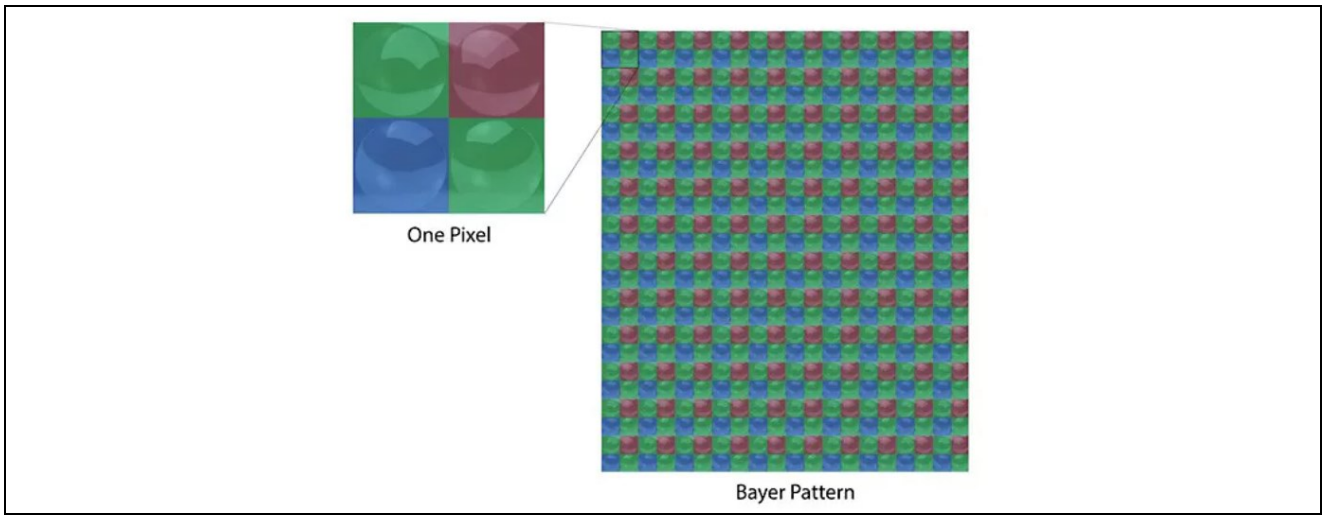

Figure 37. How a 2x2 pixel of photo detectors is arranged in a Bayer

Each photodetector is built on a silicon substrate and includes:

- A photodiode that collects photons
- A row-select transistor
- A source-follower amplifier
- A reset transistor

Above each pixel lies a microlens to focus light and a color filter to isolate the red, green, or blue wavelength. The photodiode converts light into an electrical charge, which is then amplified and transferred across pixel buses for downstream processing.

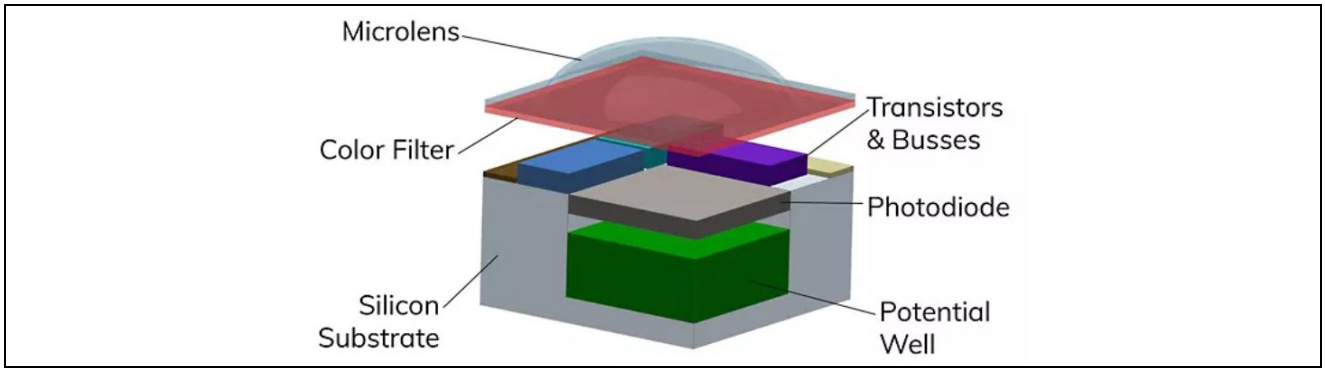

Figure 38. The geometry in a single photodetector

The entire array of pixels sits at the focal plane of the lens assembly. Surrounding analog and digital circuitry collects the amplified signals, digitizes them, and produces the final output image. This process enables the sensor to capture continuous image data for real-time applications such as the webcam streaming used in this project.

In summary, a CMOS image sensor captures light using an APS pixel structure, where each photodiode converts photons into electrical signals that are then amplified. The sensor arranges pixels in a 2×2 Bayer pattern (R, G, G, B) to optimize luminance and color sampling. Microlenses and color filters guide light into each pixel, enabling the system to generate continuous digital image data for real-time applications such as the webcam stream in this project.

Website and Support

Visit the following URLs to learn about key elements of the RA family, download components and related documentation, and get support.

RA Product Information	www.renesas.com/ra
RA Product Support Forum	www.renesas.com/ra/forum
RA Flexible Software Package	www.renesas.com/FSP
Renesas Support	www.renesas.com/support

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Mar.31.26	-	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/.