Renesas RA Family

Getting Started with PWM Output with Sub Nanosecond Delay

Introduction

This application note describes the creation of a design that generates Pulse Width Modulation (PWM) outputs using the General PWM Timer (GPT) Enhanced High-Resolution modules on Renesas RA6M3 MCUs and shows how to adjust these PWM outputs using a PWM Output Delay circuit with an offset to delay the waveform by nanoseconds.

Some typical applications for PWM with nanosecond delays are power supply control, motor control, inverter control, battery charging, digital lighting control, and power factor correction (PFC).

This application note walks through all the necessary steps, including the following:

- Board setup.
- Application overview.
- FSP configuration.
- Application design highlights.
- Using the General-Purpose Timer to generate PWM output.
- Using the PWM Output Delay circuit to adjust PWM output.

Required Resources

Development tools and software

- e^2 studio 2025-04.1 (25.4.1)
- Renesas Flexible Software Package (FSP) v6.0.0

Hardware

- Renesas EK-RA6M3 kit (RA6M3 MCU Group)
- Oscilloscope (at least two channels)

Reference Manuals

- RA Flexible Software Package Documentation Release 6.0.0
- Renesas RA6M3 Group User's Manual Rev.1.20
- EK-RA6M3-v1.0 Schematics

Contents

1. Application Overview

The application project accompanying this document serves as a reference for setting up and adjusting the PWM waveform output from the GPT PWM Timer Enhanced High-Resolution channels using the PWM Delay Generation Circuit in the RA6M3 microcontroller (MCU).

For ease of understanding of the PWM Generation and Delay Generation Circuit, the application project covers the initialization procedure and steps to adjust the timing of rising edge and falling edge in PWM output waveforms, which are output from GPT channels 0 to 3. The project also includes GPT timer configuration and trigger source configuration for user push-button interrupts, which are implemented for user interaction. You can use this example configuration and change different settings to trigger/end operation as desired.

In this application note, GPT PWM channel 0 is used as the reference timer, and GPT PWM channel 1 is used as the PWM output delay channel. Users can press the push buttons S1 and S2 to adjust the timing of the rising and falling edges in PWM output waveforms of GPT PWM channel 1.

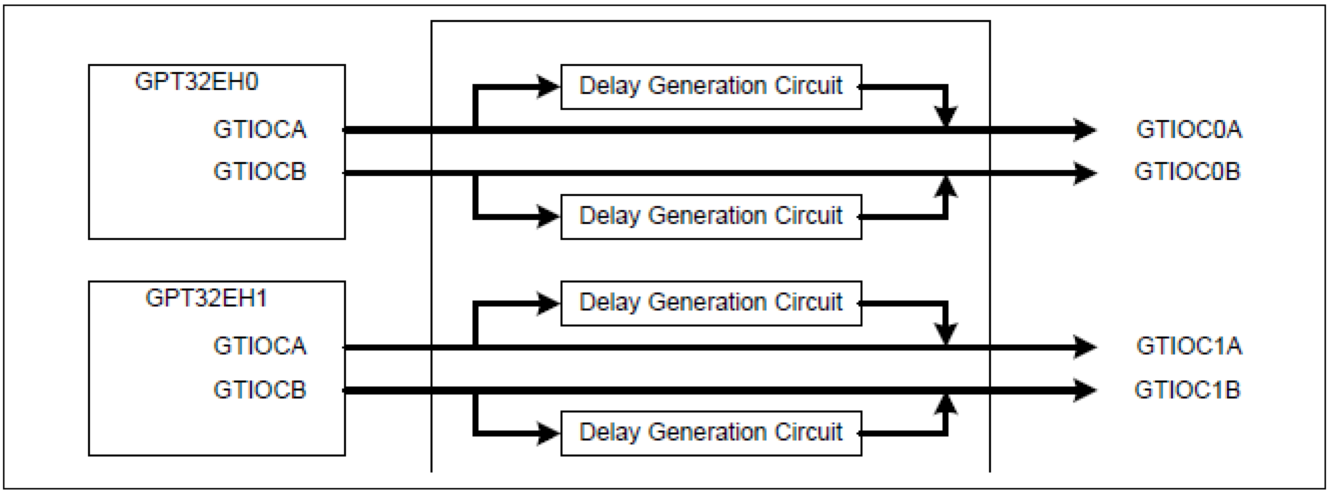

Figure 1. Block Diagram GPT PWM Channel 0 and 1 and PWM Delay Generation Circuit

When the PWM delay generation circuit is in use, the timing with which a PWM output signal rises and falls can be controlled to an accuracy of 1/32 of the period of the GPT operation clock, which is PCLKD. Since the PCLKD is set at 120 MHz in this application project, the accuracy will be 260 picoseconds.

2. GPT Timer Enhanced High-Resolution Channels

The General PWM Timer (GPT) module in the RA6M3 MCU is a 32-bit timer with six GPT32 channels, four GPT Timer Enhanced (GPT32E) channels, and four GPT Timer Enhanced High Resolution (GPT32EH) channels. PWM waveforms can be generated by controlling the up-counter, down-counter, or the up- and down-counter. The GPT can also be used as a general-purpose timer.

The rising edge and falling edge of PWM outputs from the GPT32EH channels, which are channels from 0 to 3, can be adjusted using the PWM Delay Generation circuit.

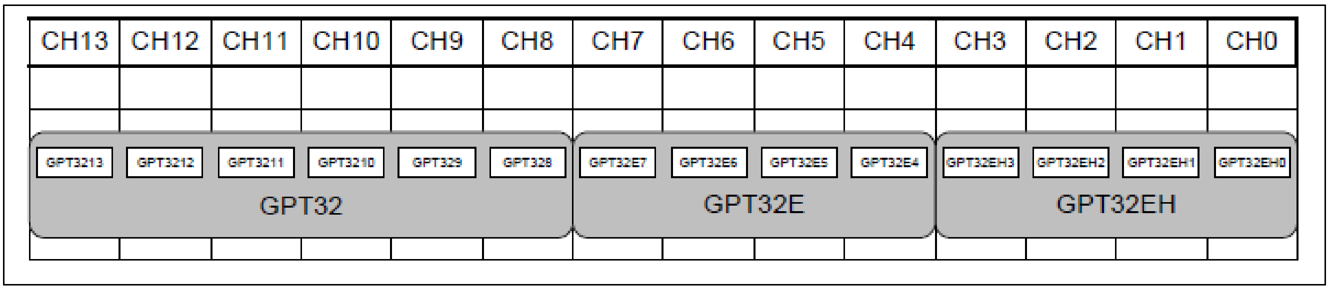

Figure 2. Correspondence between GPT Channels and Module Names

3. Board Setup

The EK-RA6M3 kit has a few switch settings that must be configured before running the application associated with this application note. These switches must be returned to the default settings per the EK-RA6M3 user manual. In addition to these switch settings, the boards contain a USB debug port and connectors to access the J-Link® programming interface.

Table 1. Switch settings for EK-RA6M3

Switch	Setting
J8	Jumper on pins 1-2
J9	Open

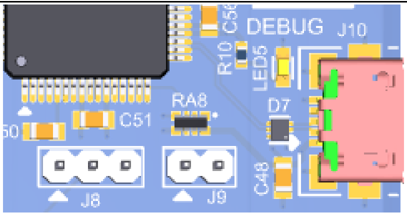

Figure 3. J8 and J9 on EK-RA6M3

The figure below shows the EK-RA6M3 kit.

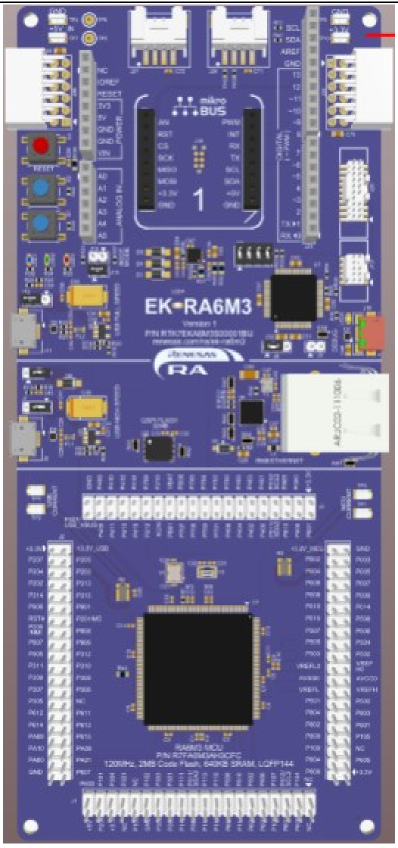

Figure 4. EK-RA6M3 Kit

4. FSP Configuration

When writing an FSP application, the FSP must first be configured. For more information, refer to the Renesas Flexible Software Package (FSP) user manual.

In this application, the FSP configuration is stored in a file named `configuration.xml`. Double-click on this file to open the **RA Configuration** tab for the project.

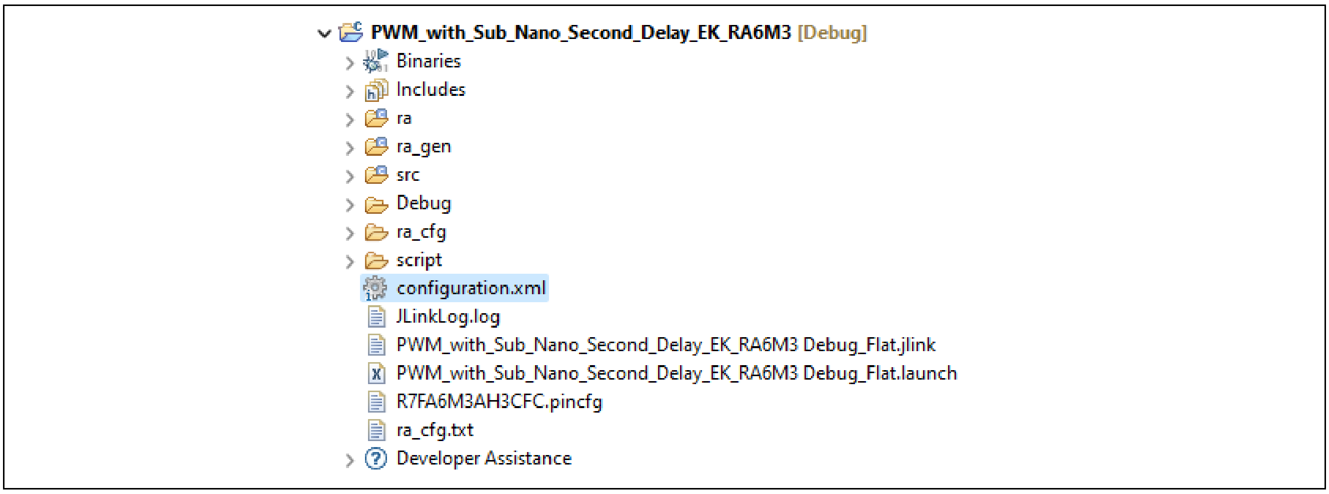

Figure 5. configuration.xml on the Project Plane

When a project is built from scratch, this configuration tab is where you will perform the initial configuration of the FSP. As shown in Figure 6, the RA Configuration pane contains a **Summary** tab highlighting the items that can be configured along with a scrolling window that lists all the software components currently selected for this project. Below this scrolling window are tabs that allow tailoring of the FSP to the needs of the specific application.

For the purposes of this application note, we will highlight a few of the details of the FSP configuration, such as the interrupt controller (r_icu) and the general PWM timer (r_gpt), as they pertain to this application. For additional details, refer to the FSP User's Manual on how to configure the FSP.

When the project has been configured appropriately, click the **Generate Project Content**, the green arrow button above the summary screen, to build all the auto-generated files necessary to implement the defined components.

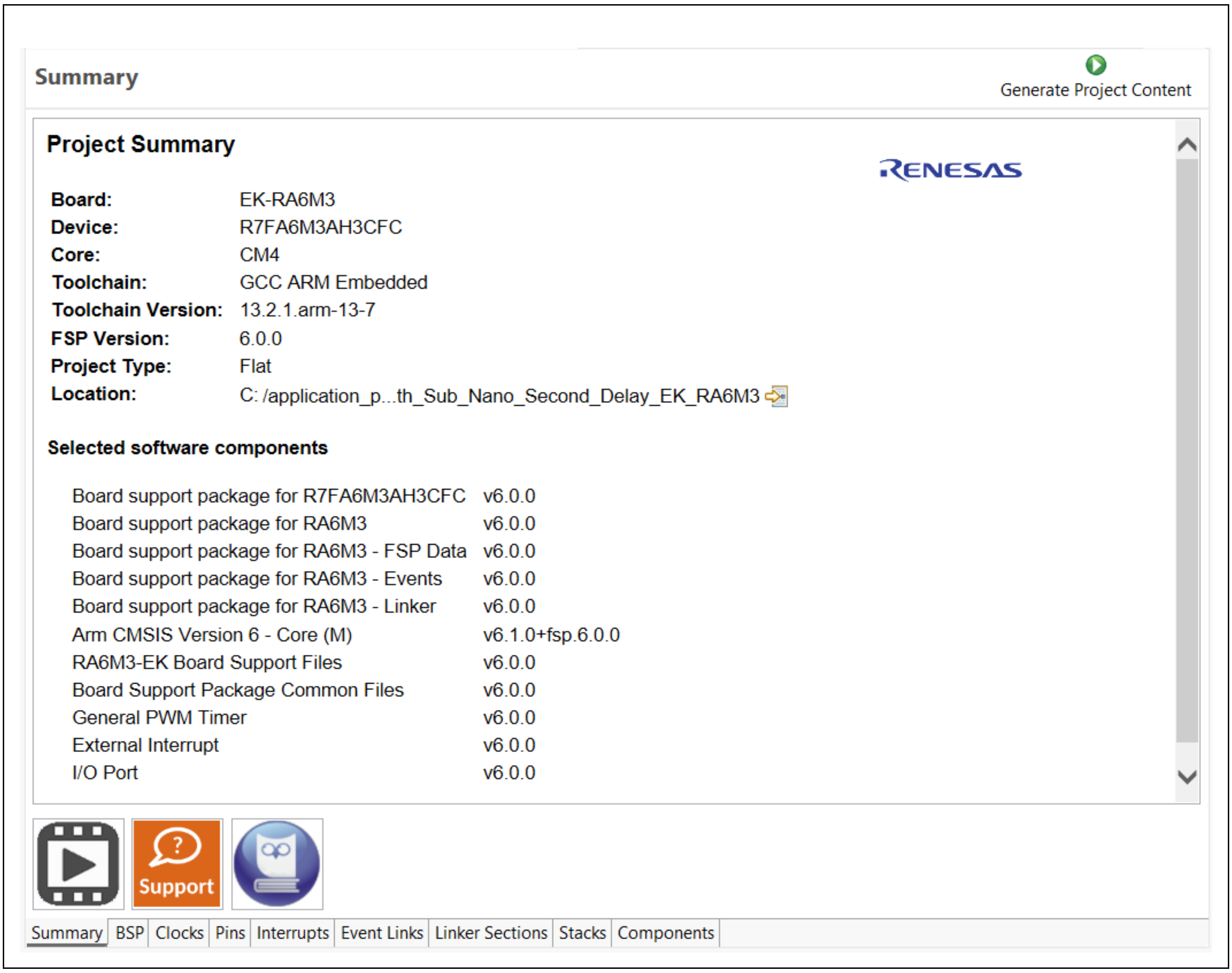

Figure 6. Summary of the Application Configuration

4.1 Components Tab

The **Components** tab shows the FSP modules that are used in this application. As shown in Figure 7, components are broken down into several categories.

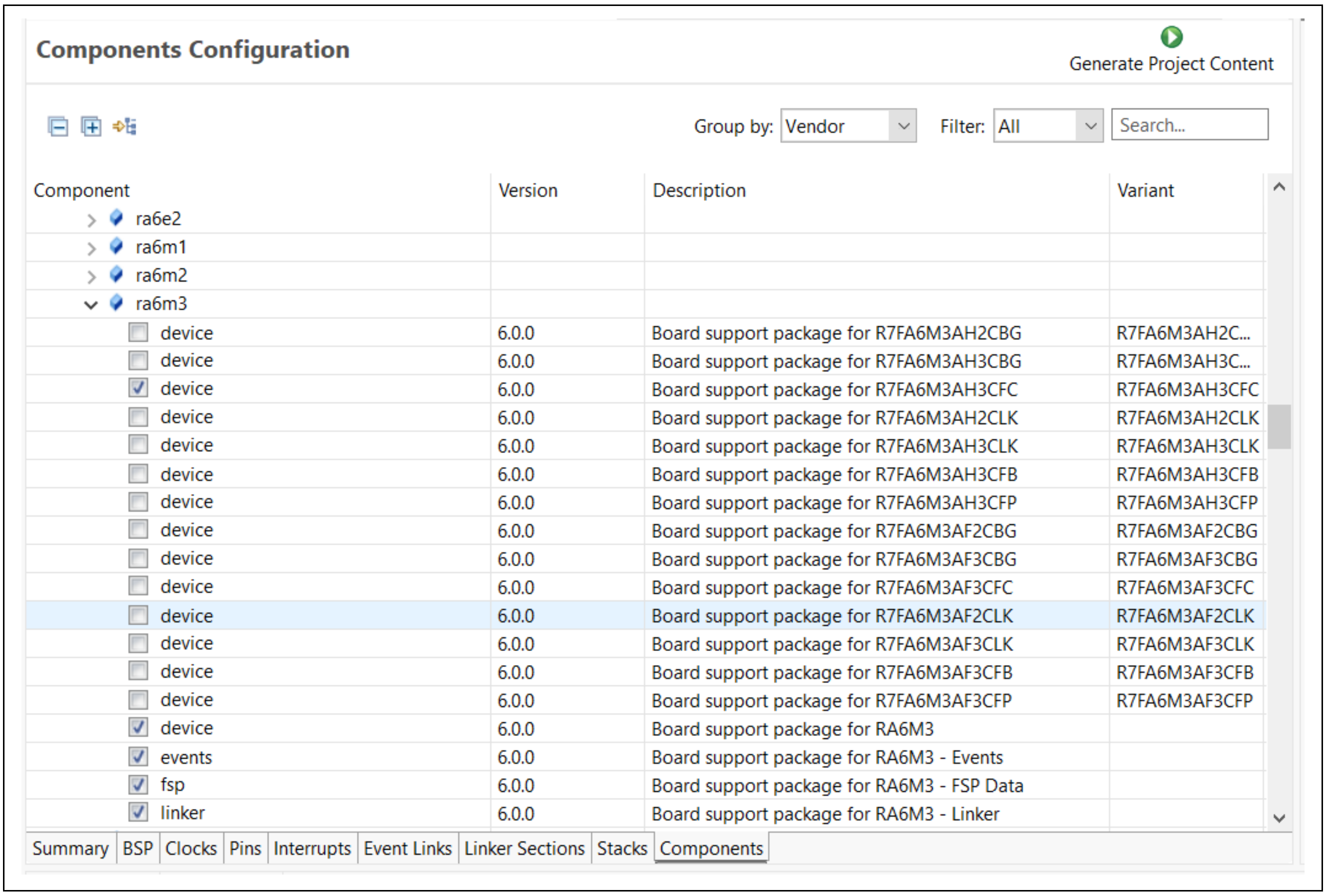

Figure 7. Components Tab Categories

You may expand any of the categories by clicking the arrow to the left of the category name.

The main FSP modules used in this application note are the general PWM timer (r_gpt), and the external input (r_icu).

4.2 Stacks Tab

The **Stacks** tab is where you can add and configure the threads that the FSP automatically creates for your application. You define a new thread by clicking the **New Stack** button  and then entering a unique name for the new thread. Once the new thread is added, the modules that the thread will use, along with any thread objects, must be defined.

As an example, after clicking HAL/**Common**, you should see something like the screen capture shown in Figure 8. This shows that the project requires multiple modules, for example, the r_icu driver which is used to control push buttons on the EK-RA6M3 kit.

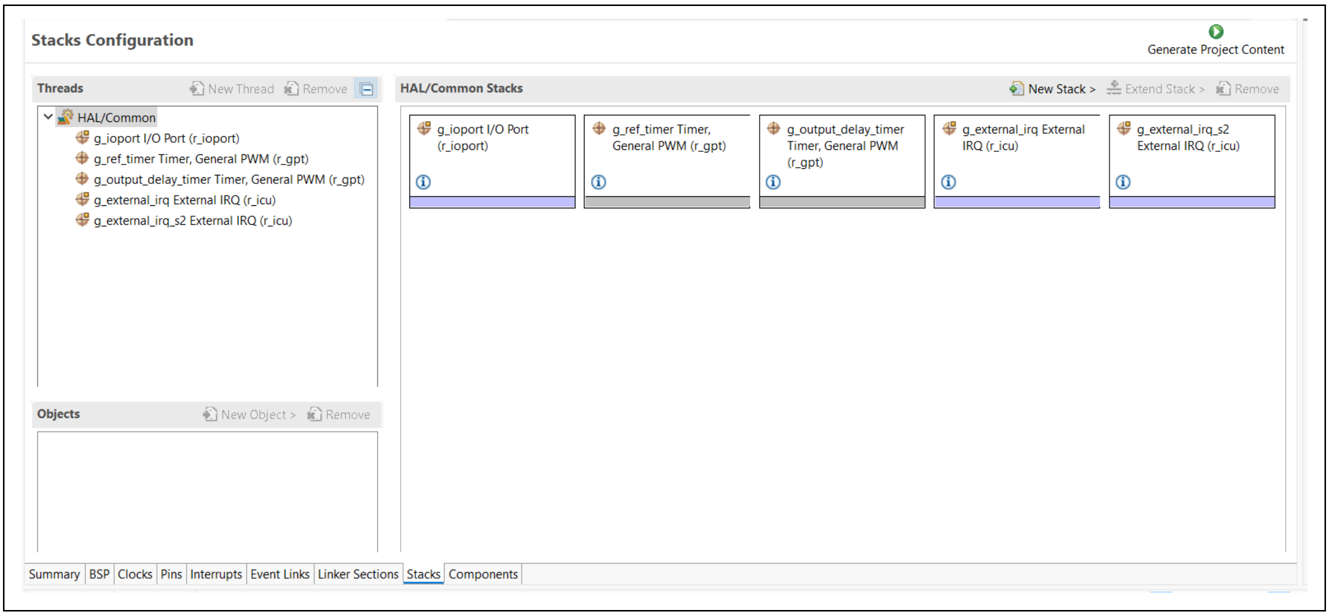

Figure 8. Properties and Modules Used for the Application

Additional modules can be added to any thread by clicking the New Stack button. If the appropriate components have been added prior to adding modules to the threads, there should not be any errors. As an example, Figure 9 shows how to add a GPT timer to the Timer Thread. The timer is added by choosing **(+) New Stack** > **Search** > **r_gpt**.

If the exact component is not chosen when the module is selected, the FSP automatically selects it. If the FSP detects errors with the module addition, it prefaces the module with an error. Errors can be examined by hovering over the module name.

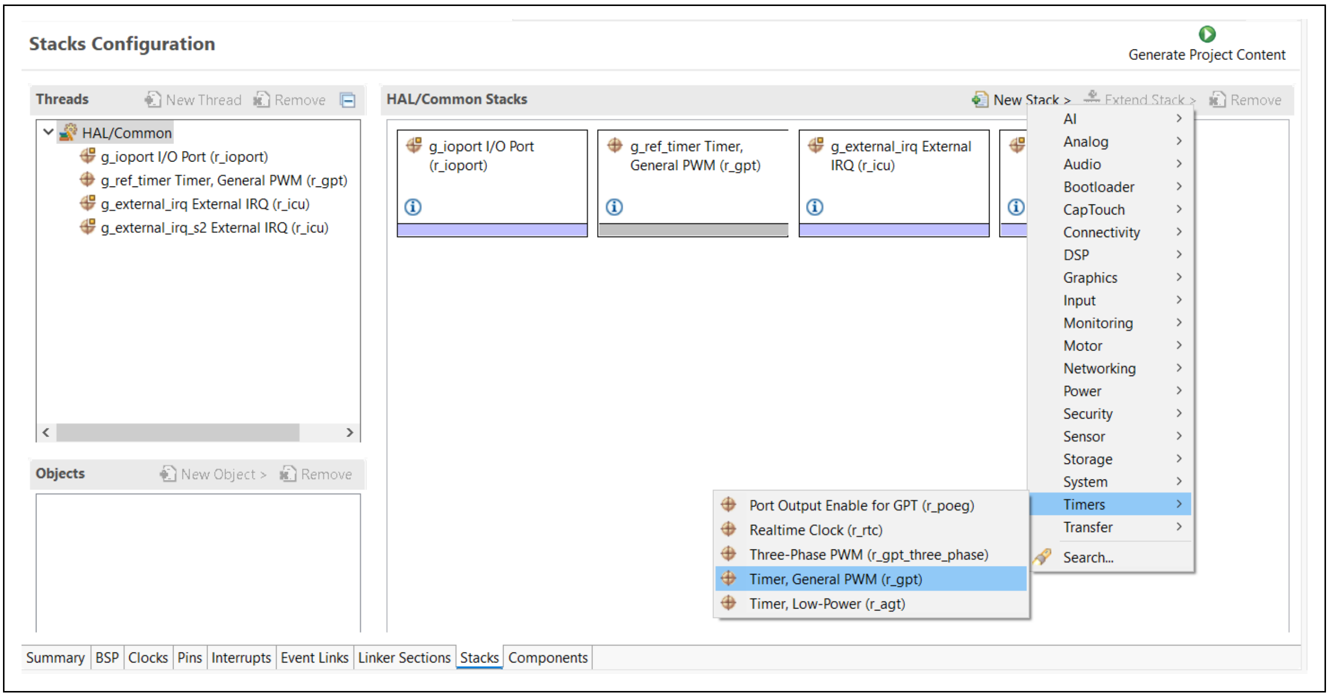

Figure 9. Adding r_gpt driver

4.3 Module Configuration

The properties must be configured once the new module has been added to the project. The properties depend on the module(s) that have been added. Use the **Properties** tab to configure them.

4.3.1 GPT Timers, PWM Output Pins

In this application note, the GPT timer channel 0 is used as the reference timer with the PWM output pin GTIOCA on P415. The GPT timer channel 1 is used as the output delay timer with PWM output pins, GTIOCA on P105, and GTIOCB on P104.

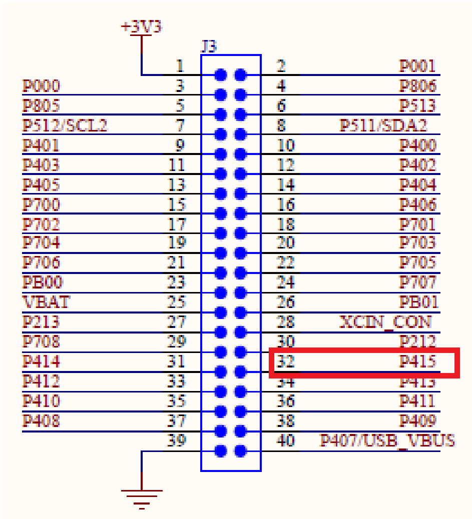

Figure 10. GPT Channel 0 GTIOCA Pin (P415)

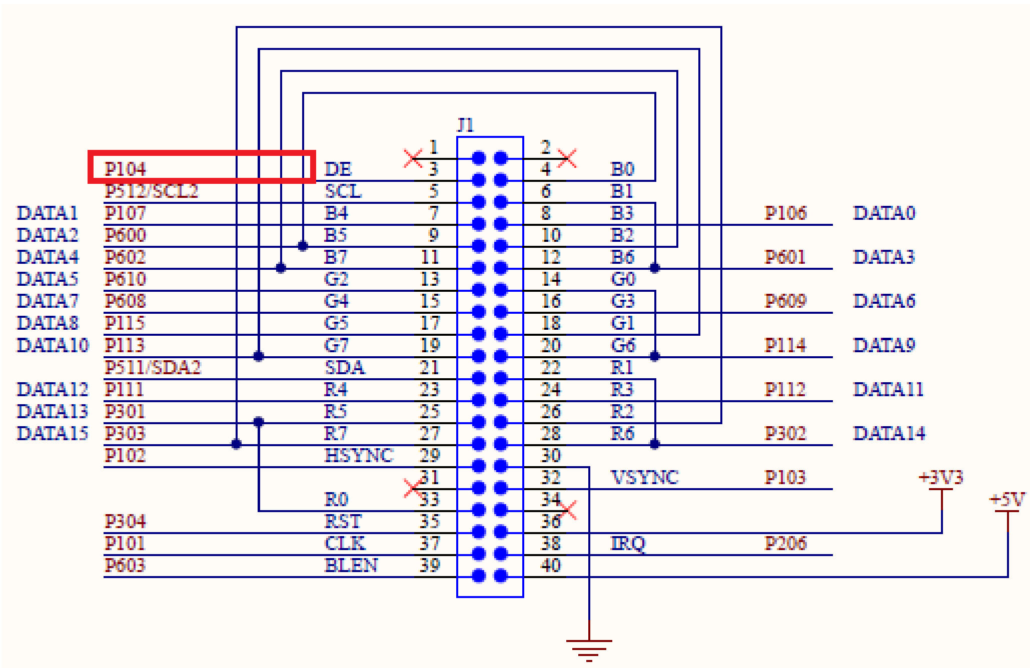

Figure 11. GPT Channel 1 GTIOCB Pin (P104)

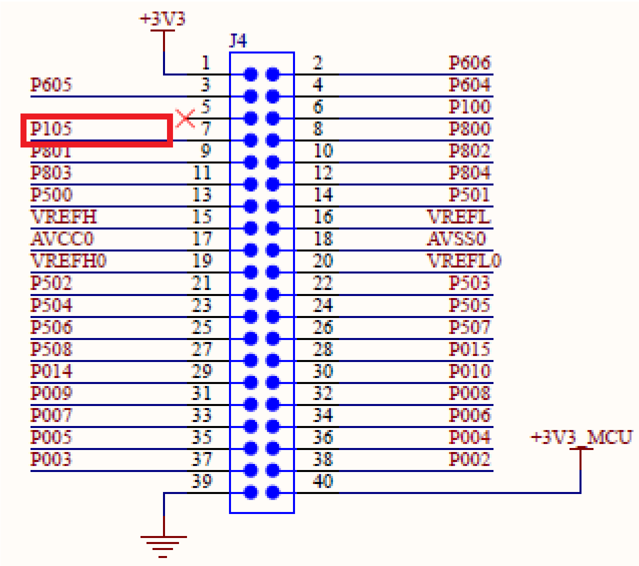

Figure 12. GPT Channel 1 GTIOCA Pin (P105)

4.3.2 GPT Timers, Pin Configuration

In **Pin Configuration**, set P415 as the GTIOCA output of GPT channel 0, set the Pin Group Selection to mixed, and set the **Operation Mode** to GTIOCA or GTIOCB.

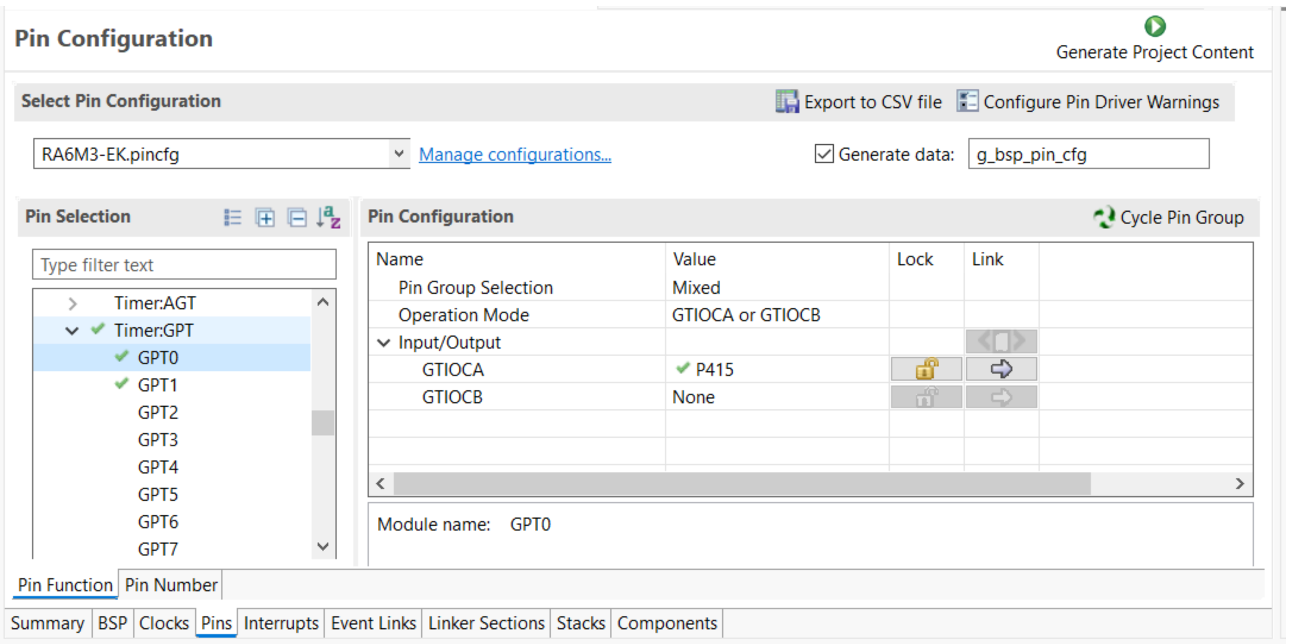

Figure 13. GPT Channel 0 Pin Configuration

In **Pin Configuration**, set P105 as the GTIOCA output of the GPT channel 1 and set P104 as GTIOCA output for the same channel. Set the **Pin Group Selection** to mixed and set the **Operation Mode** to GTIOCA and GTIOCB.

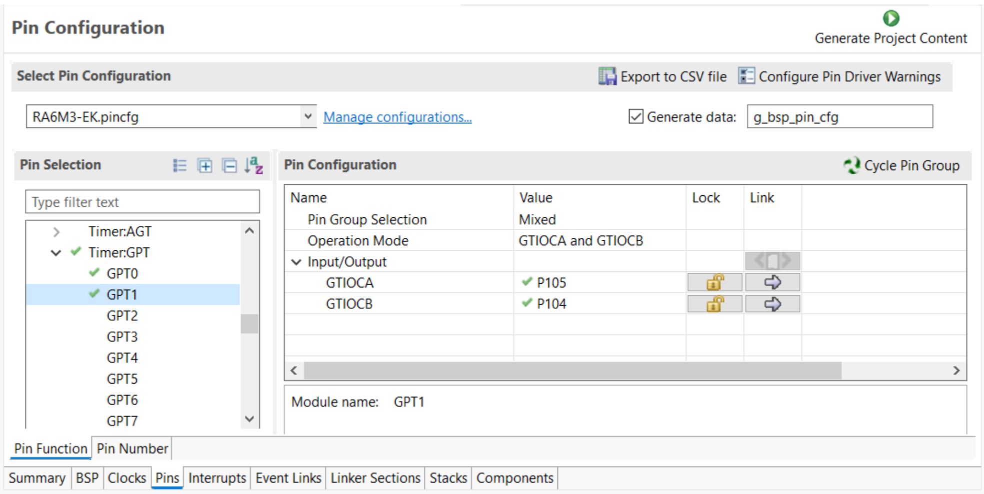

Figure 14. GPT Channel 1 Pin Configuration

4.3.3 GPT Timers, PWM Configuration

The reference timer (GPT channel 0) is set in PWM mode with timing and duty cycle as follows.

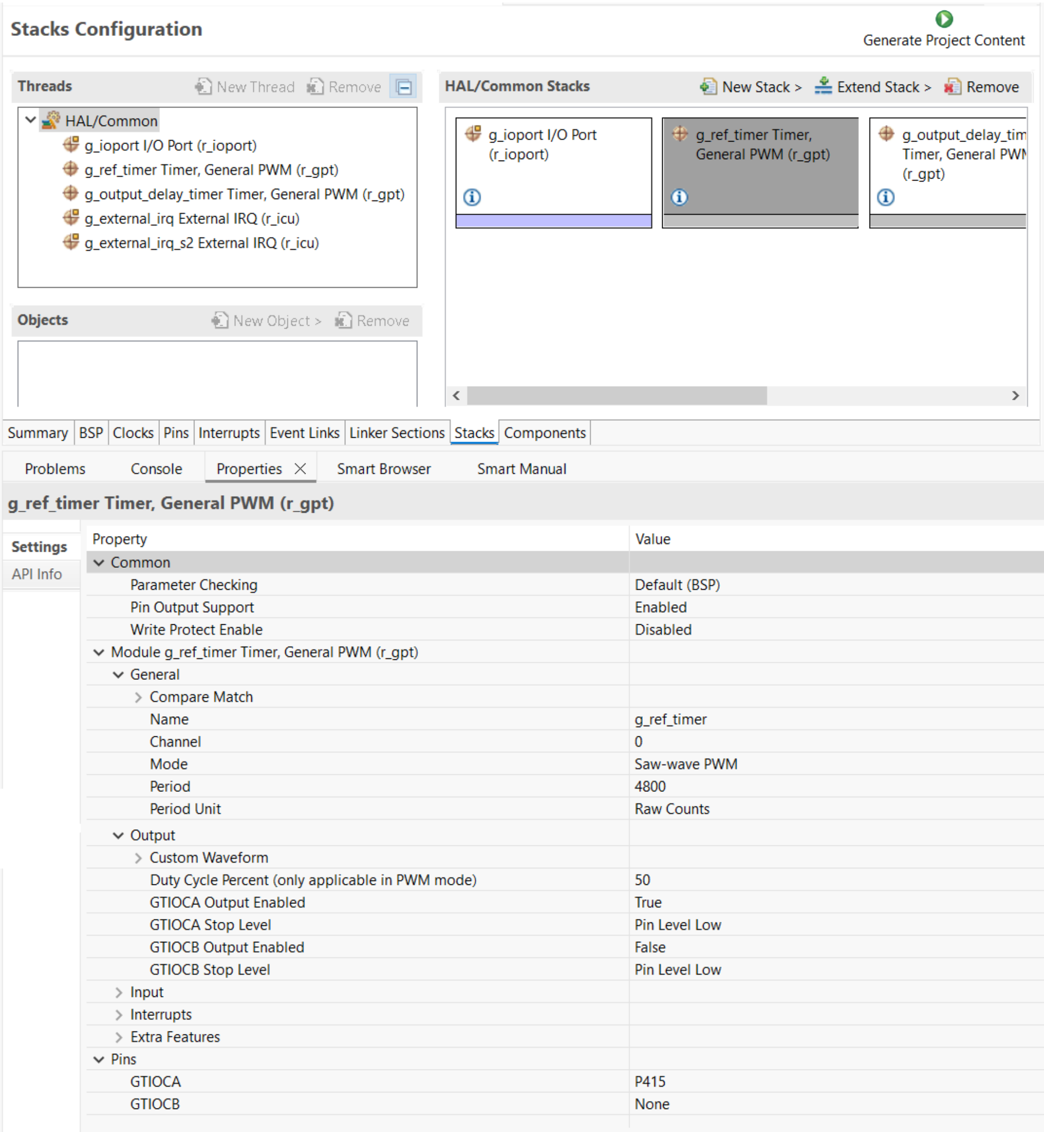

Figure 15. GPT Channel 0 Configuration

The output delay timer (GPT channel 1) is set in PWM mode with timing and duty cycle as follows.

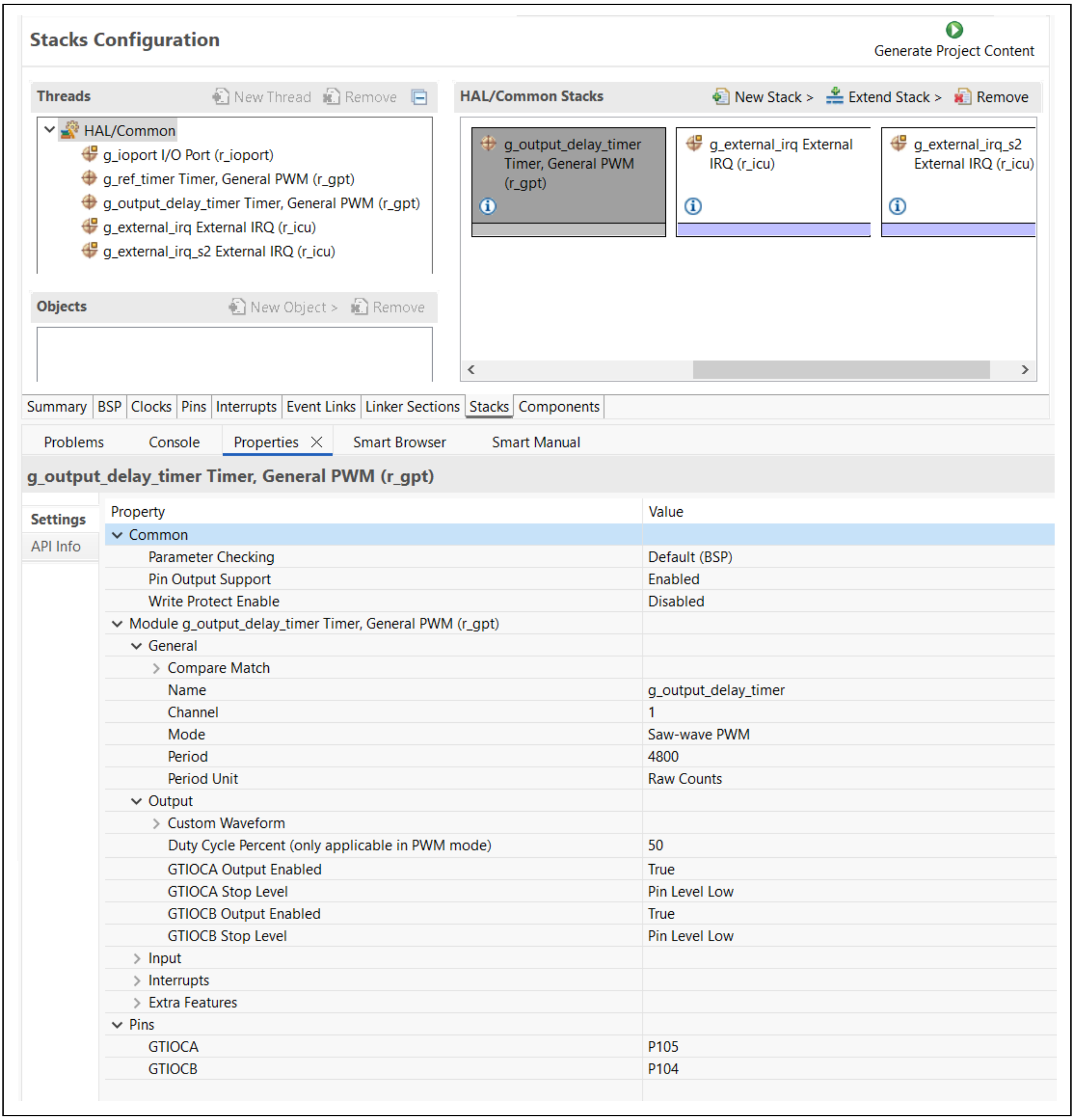

Figure 16. GPT Channel 1 Configuration

4.3.4 PWM Delay Generation Circuit

The RA6M3 MCU has four-channel delay circuits that can connect to the General PWM Timer (GPT). Figure 17 shows its block diagram.

The circuit can control the rise and fall timing of the two PWM output pins with an accuracy of up to 1/32 times the period of the GPT clock (PCLKD), for channel 0/1/2/3.

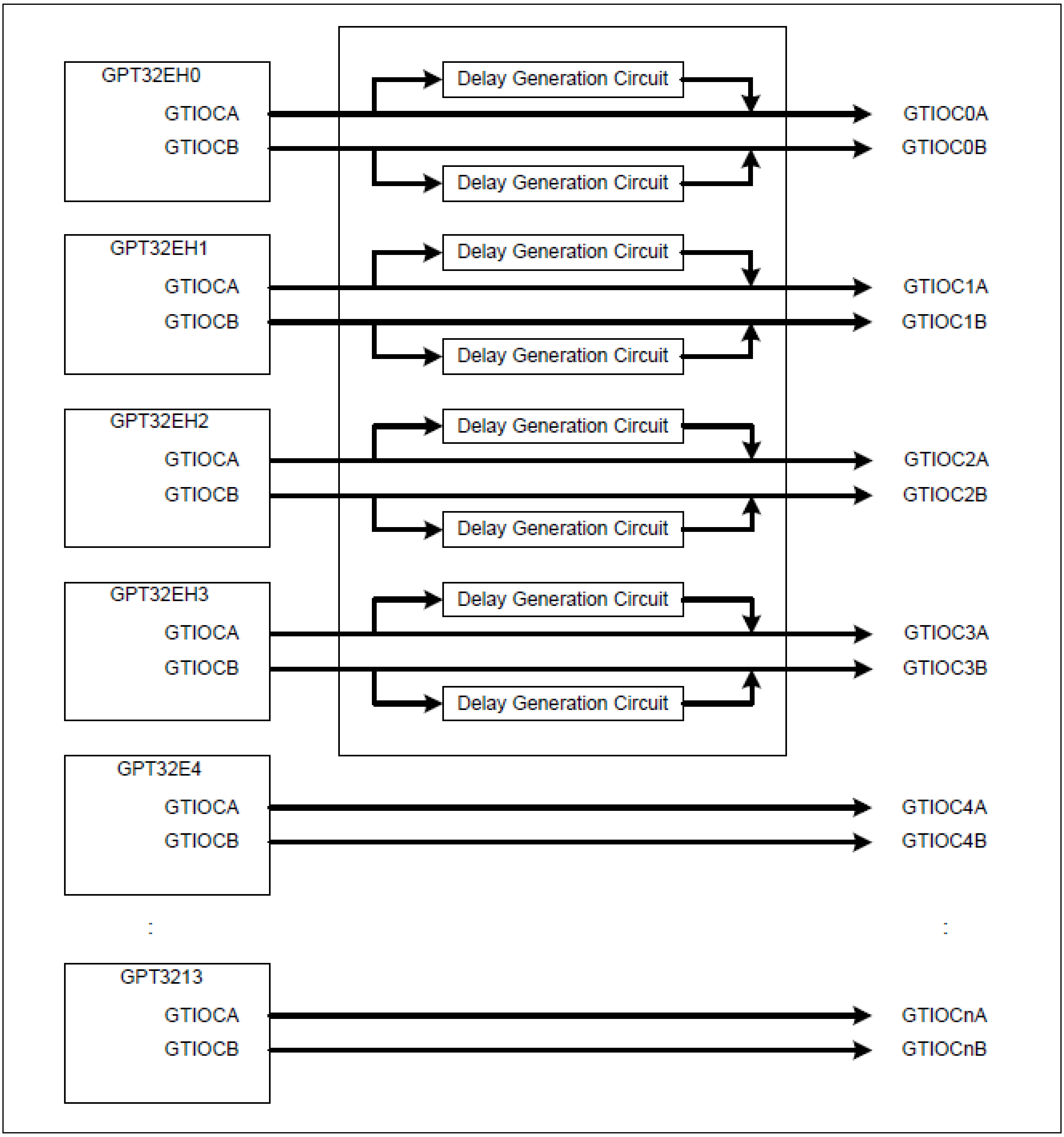

Figure 17. PWM Delay Generation Circuit Block Diagram

Figure 18 lists PWM delay generation circuit I/O pins available in RA6M3 MCU.

I/O pin	I/O	Function
GTIOC0A	Output	Delayed output of GTIOCA pin of GPT channel 0
GTIOC0B	Output	Delayed output of GTIOCB pin of GPT channel 0
GTIOC1A	Output	Delayed output of GTIOCA pin of GPT channel 1
GTIOC1B	Output	Delayed output of GTIOCB pin of GPT channel 1
GTIOC2A	Output	Delayed output of GTIOCA pin of GPT channel 2
GTIOC2B	Output	Delayed output of GTIOCB pin of GPT channel 2
GTIOC3A	Output	Delayed output of GTIOCA pin of GPT channel 3
GTIOC3B	Output	Delayed output of GTIOCB pin of GPT channel 3

Figure 18. PWM Delay Generation Circuit I/O Pins

4.3.5 Set Up PWM Delay Generation Circuit

The FSP provides register definitions for the PWM Generation Circuit in the MCU header file, such as R7FA6M3AH.h, shown in Figure 19.

```
/**
 * @brief PWM Delay Generation Circuit (R_GPT_ODC)
 */

typedef struct                         /*!< (@ 0x4007B000) R_GPT_ODC Structure                                  */
{
    union
    {
        __IOM uint16_t GTDLYCR1;       /*!< (@ 0x00000000) PWM Output Delay Control Register1                   */

        struct
        {
            __IOM uint16_t DLLEN  : 1; /*!< [0..0] DLL Operation Enable                                        */
            __IOM uint16_t DLYRST : 1; /*!< [1..1] PWM Delay Generation Circuit Reset                          */
            uint16_t              : 6;
            __IOM uint16_t FRANGE : 1; /*!< [8..8] GPT core clock Frequency Range                              */
            uint16_t              : 7;
        } GTDLYCR1_b;
    };

    union
    {
        __IOM uint16_t GTDLYCR2;         /*!< (@ 0x00000002) PWM Output Delay Control Register2                   */

        struct
        {
            __IOM uint16_t DLYBS0   : 1; /*!< [0..0] PWM Delay Generation Circuit bypass                          */
            __IOM uint16_t DLYBS1   : 1; /*!< [1..1] PWM Delay Generation Circuit bypass                          */
            __IOM uint16_t DLYBS2   : 1; /*!< [2..2] PWM Delay Generation Circuit bypass                          */
            __IOM uint16_t DLYBS3   : 1; /*!< [3..3] PWM Delay Generation Circuit bypass                          */
            uint16_t                : 4;
            __IOM uint16_t DLYEN0   : 1; /*!< [8..8] PWM Delay Generation Circuit enable                          */
            __IOM uint16_t DLYEN1   : 1; /*!< [9..9] PWM Delay Generation Circuit enable                          */
            __IOM uint16_t DLYEN2   : 1; /*!< [10..10] PWM Delay Generation Circuit enable                        */
            __IOM uint16_t DLYEN3   : 1; /*!< [11..11] PWM Delay Generation Circuit enable                        */
            __IOM uint16_t DLYDENB0 : 1; /*!< [12..12] PWM Delay Generation Circuit Disenable for GTIOCB          */
            uint16_t                : 3;
        } GTDLYCR2_b;
    };
    __IM uint16_t              RESERVED[10];
    __IOM R_GPT_ODC_GTDLYR_Type GTDLYR[4]; /*!< (@ 0x00000018) PWM DELAY RISING                                 */
    __IOM R_GPT_ODC_GTDLYR_Type GTDLYF[4]; /*!< (@ 0x00000028) PWM DELAY FALLING                                */
} R_GPT_ODC_Type;                          /*!< Size = 56 (0x38)                                                */
```

Figure 19. Register Definitions for PWM Delay Generation Circuit in R7FA6M3AH.h

You can directly access the above registers to set up and adjust the PWM Delay Generation circuit.

Figure 20 shows the initialization sequence for the PWM Delay Generation Circuit.

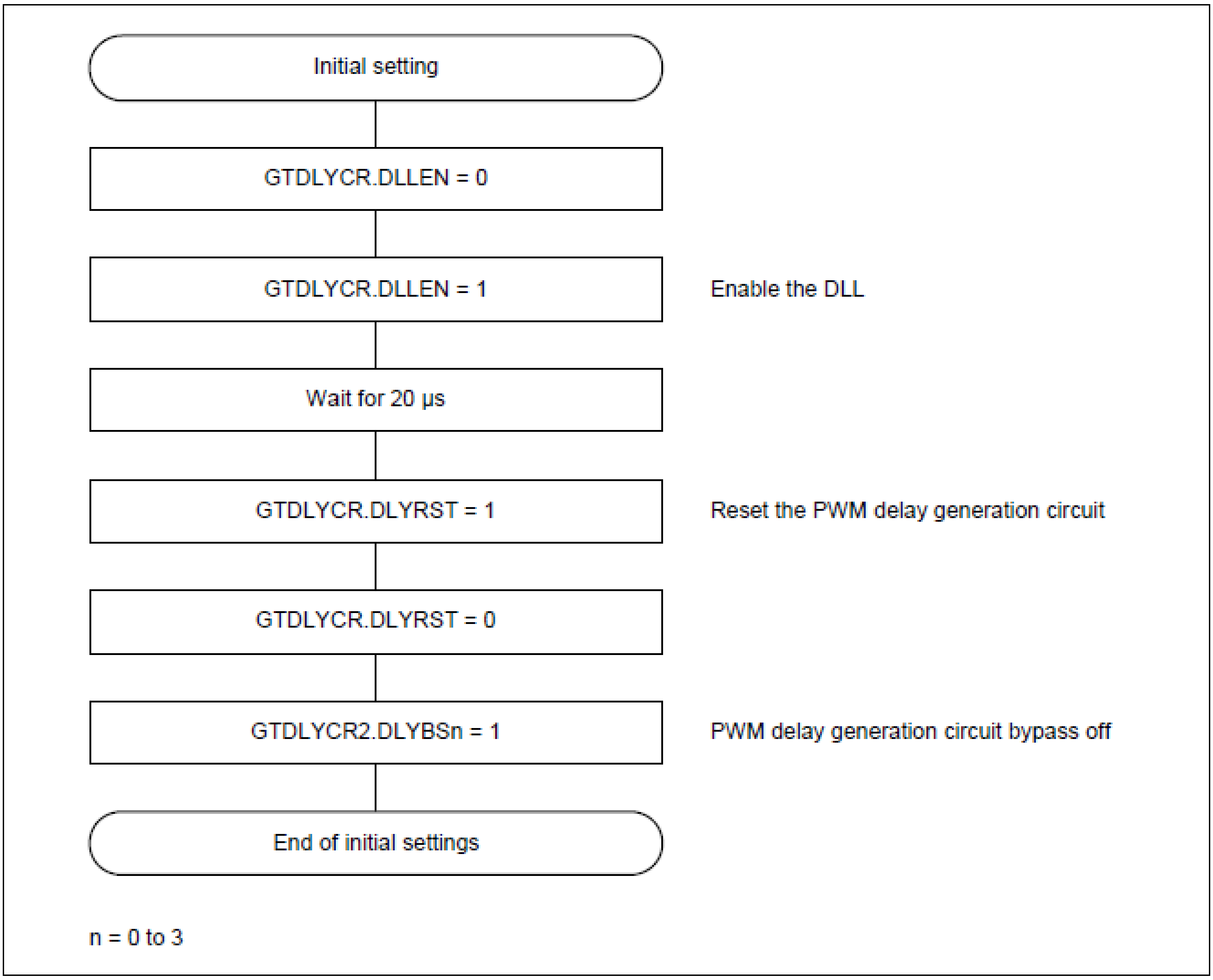

Figure 20. Initialization Flow for the PWM Delay Generation Circuit

In the PWM delay generation circuit, delay can be applied to rising and falling edges of the PWM output to an accuracy of 1/32 of the period of the GPT operation clock (PCLKD). This is described in section 23.3.3, PWM Output Operating Mode in R01UH0886 User's Manual. Delays associated with the settings are reflected in the PWM output with the timing described in section 24.3.2, Timing for Transfer of GTDLYRnA, GTLDYRnB, GTDLYFnA, and GTDLYFnB Register Settings in the R01UH0886 User's Manual on the Renesas website. The figure below shows the association between the GTDLYRnA, GTLDYRnB, GTDLYFnA, and GTDLYFnB registers and the PWM outputs.

PWM output pin	Rising-edge delay setting register	Falling-edge delay setting register
GTIOC0A	GTDLYR0A	GTDLYF0A
GTIOC0B	GTDLYR0B	GTDLYF0B
GTIOC1A	GTDLYR1A	GTDLYF1A
GTIOC1B	GTDLYR1B	GTDLYF1B
GTIOC2A	GTDLYR2A	GTDLYF2A
GTIOC2B	GTDLYR2B	GTDLYF2B
GTIOC3A	GTDLYR3A	GTDLYF3A
GTIOC3B	GTDLYR3B	GTDLYF3B

Figure 21. Association Between PWM Output Pins and Delay Setting

When the PWM delay generation circuit is in use, the timing with which a PWM output signal rises and falls can be controlled to an accuracy of 1/32 of the period of the GPT operation clock (PCLKD). When this option is not in use, the period of the PWM output waveform is controlled to accuracy of one period of PCLKD.

Additionally, the delay settings also control the periods at high and low level for the PWM waveform to the given accuracy. PWM delay generation circuit channels can be individually enabled or disabled.

5. Application Code Highlights

This section details some highlights of the application.

```
 * @brief This function is setting up GPT/PWM timer
fsp_err_t gpt_ref_timer_PWM_init(void)
{
    fsp_err_t err = FSP_SUCCESS;
    /* Open GPT */
    err = R_GPT_Open(&g_ref_timer_ctrl, &g_ref_timer_cfg);
    if(FSP_SUCCESS != err)
    {
        return err;
    }
    /* Enable GPT Timer */
    err = R_GPT_Enable(&g_ref_timer_ctrl);
    /* Handle error */
    if (FSP_SUCCESS != err)
    {
        return err;
    }
    /* Reset GPT timer */
    err = R_GPT_Reset(&g_ref_timer_ctrl);;

    return err;
}

 * @brief This function is setting up GPT/PWM timer
fsp_err_t gpt_output_delay_timer_PWM_init(void)
{
    fsp_err_t err = FSP_SUCCESS;
    /* Open GPT */
    err = R_GPT_Open(&g_output_delay_timer_ctrl, &g_output_delay_timer_cfg);
    if(FSP_SUCCESS != err)
    {
        return err;
    }
    /* Enable GPT Timer */
    err = R_GPT_Enable(&g_output_delay_timer_ctrl);
    /* Handle error */
    if (FSP_SUCCESS != err)
    {
        return err;
    }
    /* Reset GPT timer */
    err = R_GPT_Reset(&g_output_delay_timer_ctrl);;

    return err;
}
```

Figure 22. Initialize GPT Timers

```
void odc_init(void)
{
    /* Initialize the PWM Delay Generation Circuit */
    R_GPT_PwmOutputDelayInitialize();

    /* GPT0 (Ref PWM) & GPT1 (PWM Output Delay) simultaneously */
    R_GPT0->GTSTR = 0x00000003;
}
```

Figure 23. Initialize PWM Delay Generation Circuit

```
/* Wait for the pushbutton S1 to to be pressed and the GPT1 (PWM Output Delay) overflow to occur at the same time */
if((true == g_sw1_press) && ( 1 == g_output_delay_overfow_flag ))
{
    /* Clear user pushbutton flag */
    g_sw1_press = false;

    /* Set output delays */
    /* for rising edge, pin A */
    err = R_GPT_PwmOutputDelaySet(&g_output_delay_timer_ctrl,
                                  GPT_PWM_OUTPUT_DELAY_EDGE_RISING,
                                  output_delay_s1,
                                  GPT_IO_PIN_GTIOCA);
    if(err != FSP_SUCCESS)
    {
        APP_ERR_TRAP(err);
    }

    /* for falling edge, pin B */
    err = R_GPT_PwmOutputDelaySet(&g_output_delay_timer_ctrl,
                                  GPT_PWM_OUTPUT_DELAY_EDGE_FALLING,
                                  output_delay_s1,
                                  GPT_IO_PIN_GTIOCB);
    if(err != FSP_SUCCESS)
    {
        APP_ERR_TRAP(err);
    }

    output_delay_s1++;
    if(output_delay_s1 >= 32)
    {
        output_delay_s1 = 0;
    }
}
```

Figure 24. Add Delays on Rising Edge and Falling Edge of PWM Waveform

6. Importing and Building the Project

To bring the application into e² studio, follow these steps:

1. Launch e² studio.
2. In the workspace launcher, browse to the chosen workspace.
3. Close the **Welcome** window.
4. In e² studio go to **File** > **Import.**
5. In the **Import** dialog box, pick **Existing Projects into Workspace.**
6. Select the archive file.
7. Select the project and click **Finish**.
8. Open `configuration.xml`.
9. Click on **Generate Project Content** on the FSP configurator window.
10. Now build the project.

7. Downloading the Executable to the EK-RA6M3 Kit

To connect and run the code, follow these steps:

1. Connect the PC to the USB port next to the Ethernet jack silkscreened DEBUG using the USB cable.
2. Go to **Run > Debug configurations.**
3. Click **Debug**. The program will break at the reset handler.
4. Click **Switch** to the **Debug perspective** when prompted by the e^2 studio.
5. Click **Run** > **Resume**.
6. Press S1 to adjust the output delay on GPT channel 1 GTIOCA rising edge and GTIOCB falling edge. Press S2 to adjust the output delay on GPT channel 1 GTIOCB rising edge and GTIOCA falling edge.

8. Output Waveforms

The project generates 25-kHz PWM output on P415 (GPT channel 0 GTIOCA), P105 (GPT channel 1 GTIOCA) and P104 (GPT channel 1 GTIOCB).

The following figures show the picture of initial output waveforms on these pins with P415 output in yellow, P105 in green and P104 in blue.

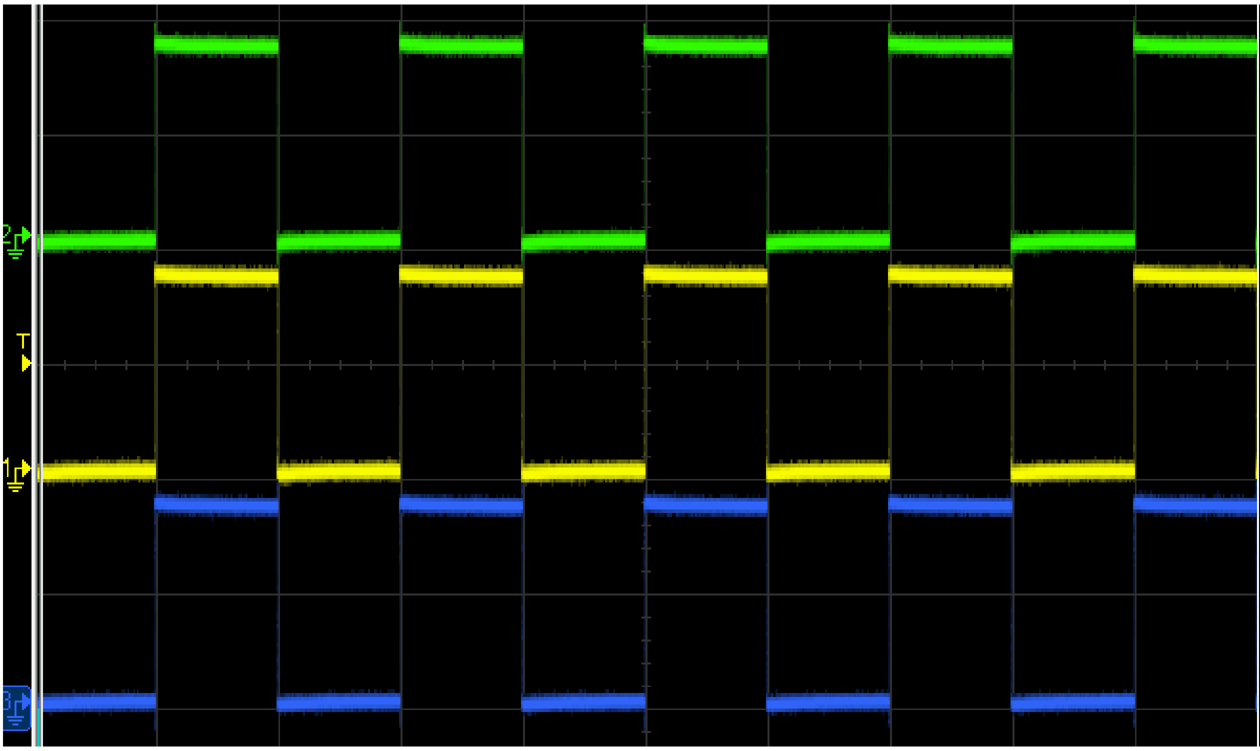

Figure 25. Initial Output Waveforms

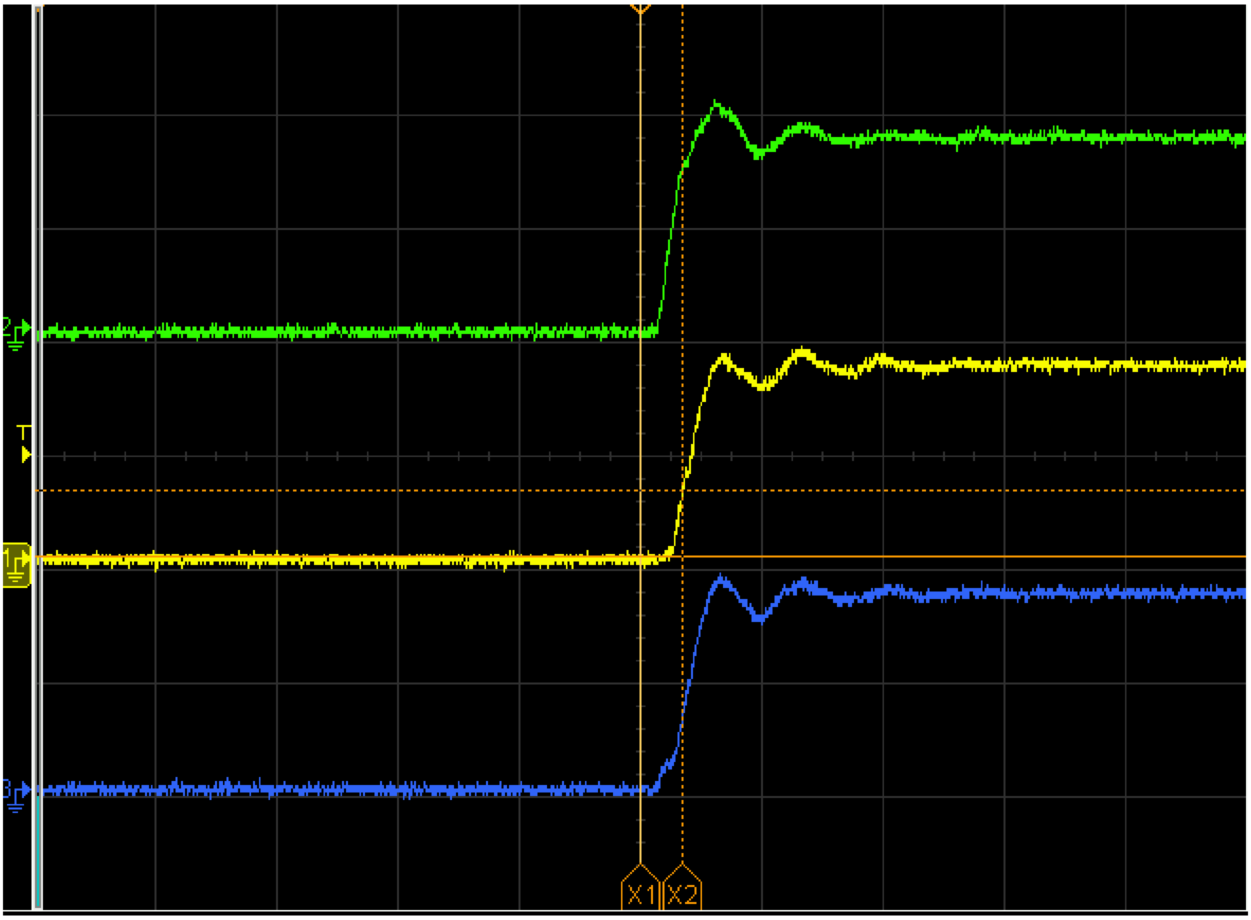

Figure 26. Output Waveforms with No Delay

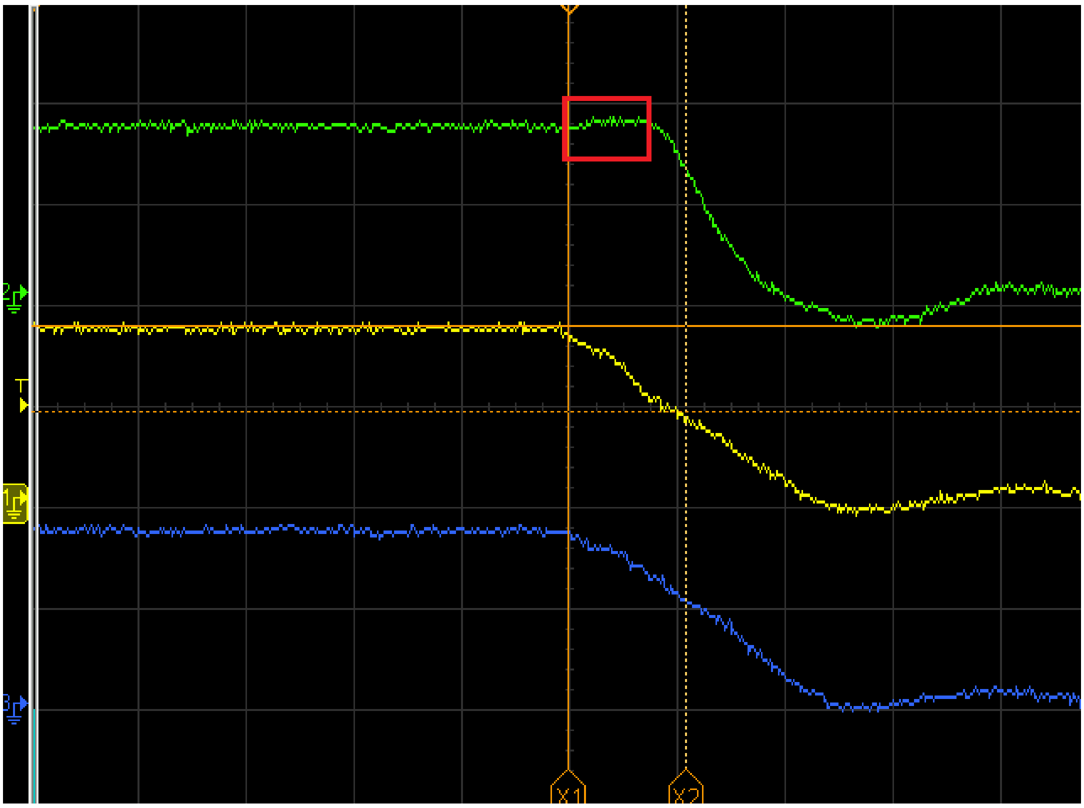

Figure 27. Output Waveforms with Delay on Falling Edge GTIOCA1: GTDLYF[1].A = 16.

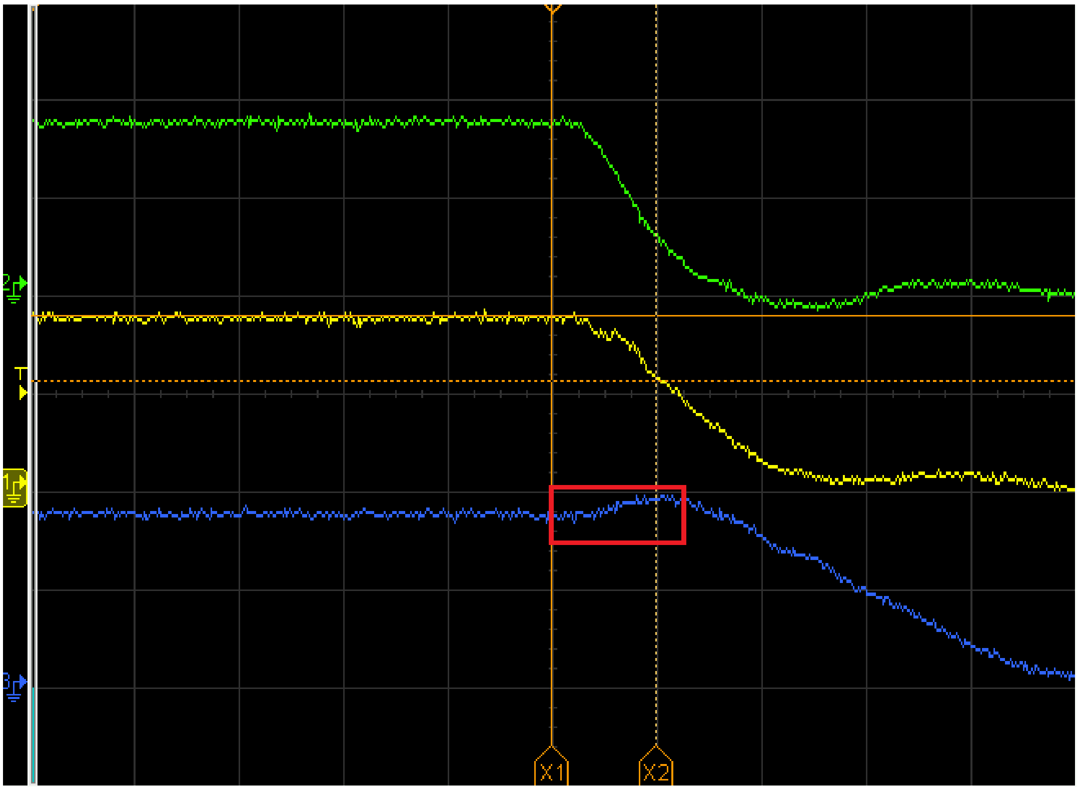

Figure 28. Output Waveforms with Delay on Falling Edge GTIOCB1: GTDLYF[1].B = 31.

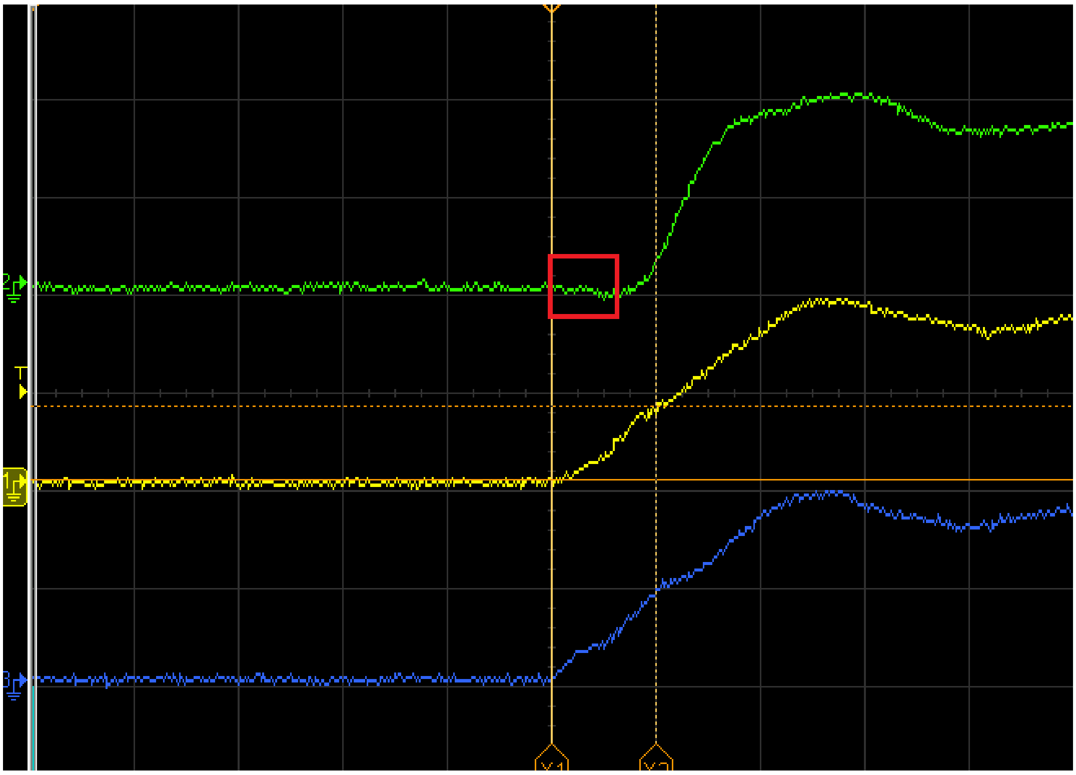

Figure 29. Output Waveforms with Delay on Rising Edge GTIOCA1: GTDLYR[1].A = 16.

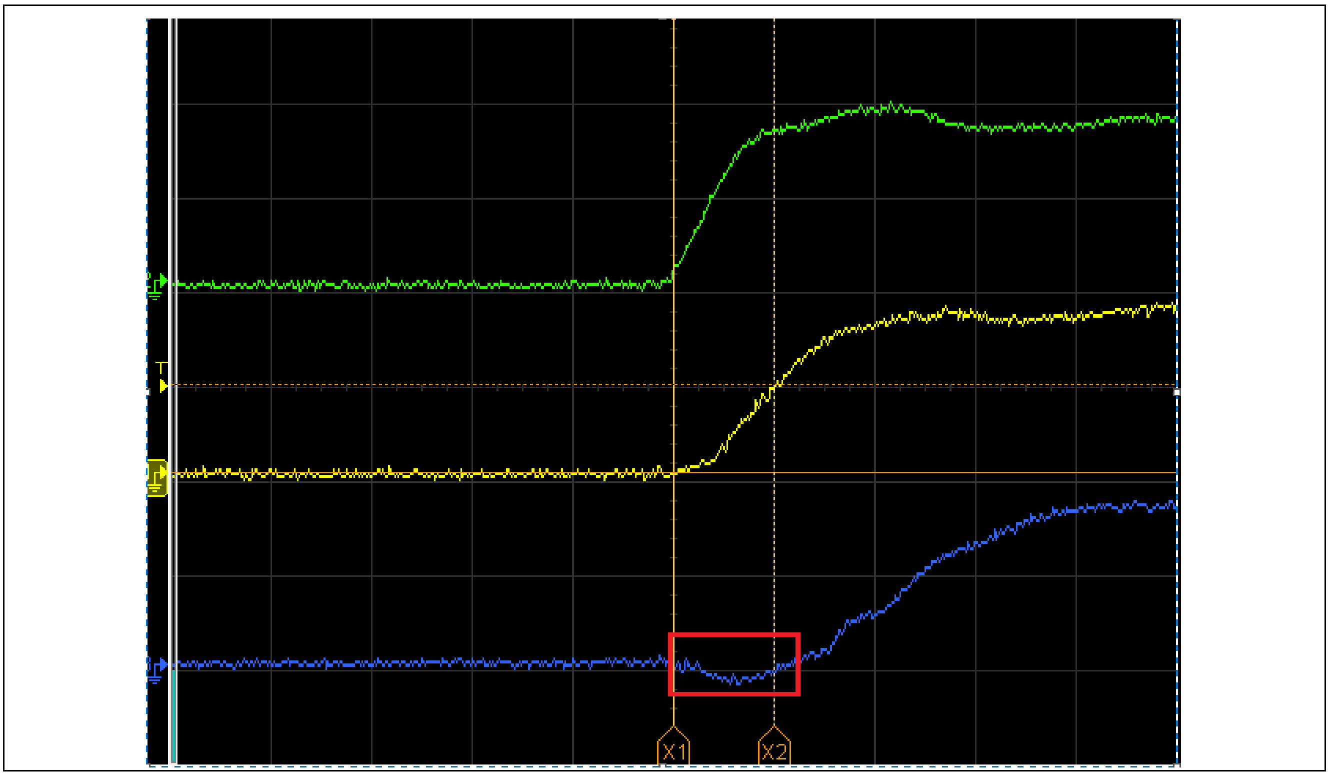

Figure 30. Output Waveforms with Delay on Rising Edge GTIOCB1: GTDLYR[1].B = 31.

Website and Support

Visit the following vanity 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	Feb.16.22	-	Initial version
1.01	Aug.03.23	-	Updated for FSP v4.4.0
1.02	Jul.16.24	-	Updated for FSP v5.2.0
1.03	Jul.08.25	-	Updated for FSP v6.0.0

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

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

1. Precaution against Electrostatic Discharge (ESD)

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

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

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

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

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

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

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

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

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

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/.