Renesas RA0 Family

Low-speed On-chip Oscillator (LOCO) and Middle-speed On-chip Oscillator (MOCO) Clock Frequency Correction

Introduction

This application note describes how to correct the oscillation clock frequency of the low-speed on-chip oscillator (LOCO) and the middle-speed on-chip oscillator (MOCO). It utilizes the middle-speed on-chip oscillator trimming register (LIOTRM) for LOCO and the middle-speed on-chip oscillator trimming register (MIOTRM) for MOCO, both of which are integrated in the RA0E1 MCU.

An error in the oscillation frequency of the LOCO or the MOCO is detected using a high-speed on-chip oscillator (HOCO). The LOCO trimming register (LIOTRM) or the MOCO trimming register (MIOTRM) is then adjusted to set the oscillation frequency of the LOCO to approximately 32.768 kHz or the oscillation frequency of the MOCO to approximately 4 MHz, respectively.

The RA0E1 MCU is the focus of the primary settings and examples, but the RA0E2 also follows the same principles. Although there are differences in LED pins between the two FPB boards used in this application, they are handled by the Renesas FSP.

Refer to the section Porting the Example Project to a Different RA0 MCU regarding how to port the application project to a different RA0 MCU, which includes necessary modifications to the FSP configuration and the source code.

Required Resources

The following resources are referenced throughout this application note.

Development Tools and Software

- e^{2}studio version: 2025-04 (25.4.0)
- LLVM Embedded Toolchain for Arm v18.1.3
- Renesas Flexible Software Package (FSP) v5.9.0 or later.

Target Devices

RA0E1, RA0E2 MCU

When adapting the sample program from this application note for a different microcontroller (MCU), ensure that you modify it to align with the specifications of the target microcontroller. After implementing these changes, conduct a thorough evaluation of the updated program.

Hardware Requirements

- The application note uses a PC running the Windows® 10 operating system as an example. For a complete list of supported operating systems, please refer to the corresponding user manual for the development tools and software.
- FPB-RA0E1
- FPB-RA0E2

Reference Manuals

- RA Flexible Software Package Documentation Release v5.9.0
- Renesas RA0E1 Group User’s Manual Rev.1.1
- Renesas RA0E2 Group User’s Manual Rev.1.0
- FPB-RA0E1-v1.0 Schematics
- FPB-RA0E2-v1.0 Schematics

Contents

1. Specifications

1.1 Overview of Specifications

This application note uses HOCO to detect an error in the clock oscillation frequency of the LOCO and the MOCO. The LIOTRM and MIOTRIM registers are adjusted to set the LOCO's oscillation frequency close to 32.768 kHz and the MOCO's oscillation frequency close to 4MHz, respectively.

When the start switch is short-pressed, the 32-bit interval timer TML32 operates in 16-bit capture mode to measure the period of either LOCO or MOCO. The HOCO serves as the count clock for the 32-bit interval timer. If the count value from the 32-bit interval timer is not within the desired range, the LIOTRM register or MIOTRM register is changed to adjust the LOCO's frequency to close to 32.768 kHz or the MOCO's frequency to close to 4 MHz, respectively. The target range for the LOCO's oscillation frequency is 32.768 kHz ± 1.6% (32.571 kHz to 32.965 kHz), and the target range for the MOCO's oscillation frequency is 4 MHz ± 1.3% (3.948 MHz to 4.052 MHz), under the condition that the HOCO frequency is within ±1.0%.

The 32-bit interval timer measures the period of both the LOCO and MOCO. Measure the pulse interval four times to enhance accuracy and detect any errors in the oscillation frequencies of the LOCO and MOCO.

Caution: The sample code provides specified times and calibration methods as examples. In this code, input from the start switch initiates the calibration process, simplifying processing flows and enhancing clarity. Adjust the timing for starting calibration and the intervals between start timings according to the specific system requirements. This application note outlines two calibration methods; select the one that is most appropriate for your system.

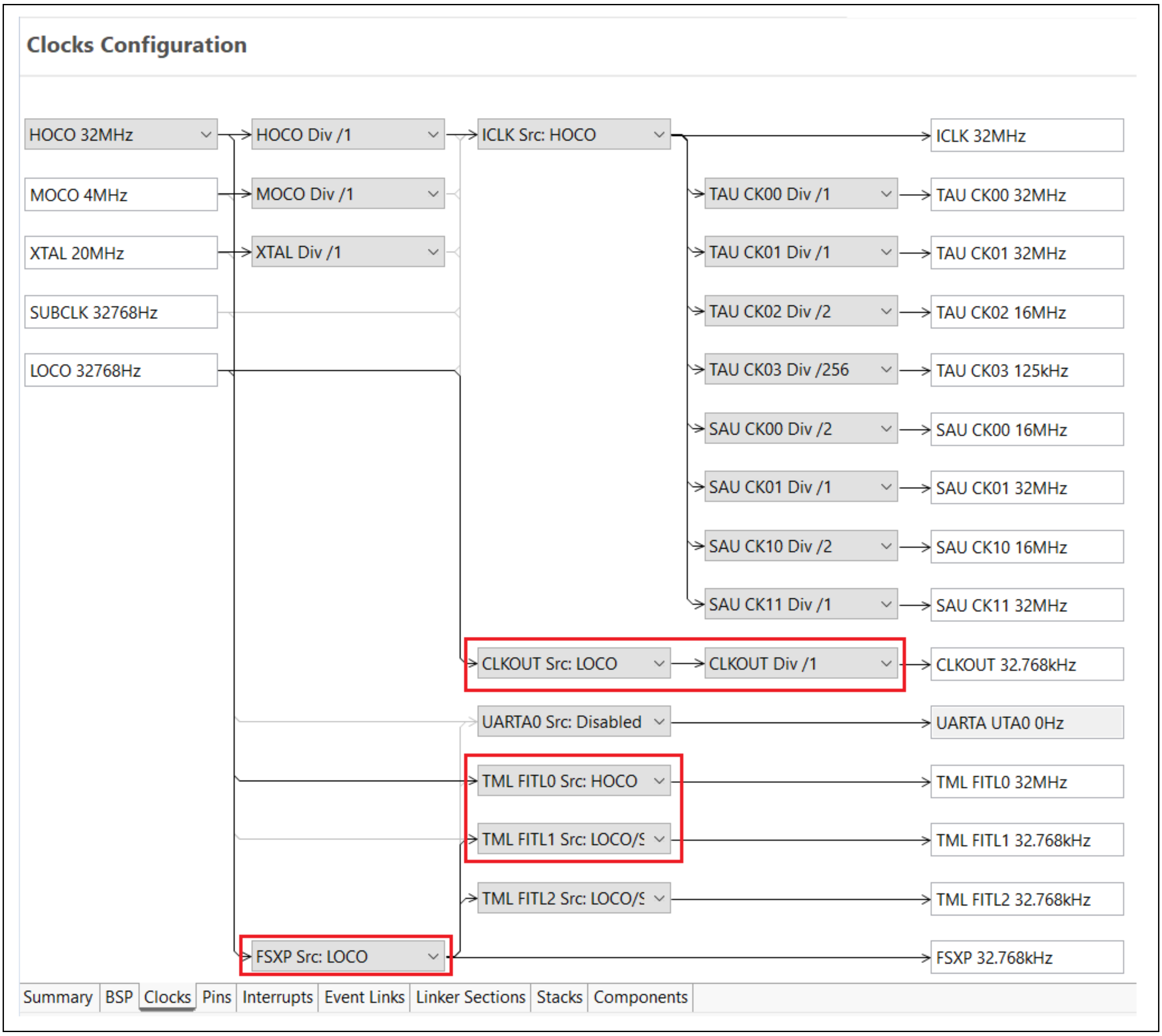
Table 1-1 lists the peripheral functions to be used and their uses.

Table 1-1 Primary Peripheral Function and Use

Primary Peripheral Function	Use
P200 (S1), pin input edge detection interrupt (R_ICU)	This is used for selecting the mode and starting the correction process.
32-bit interval timer channel (R_TML) used in 16-bit capture mode: Channels 0 and 1 are connected to operate as a 16-bit counter using the count source, channels 2 and 3 are connected to operate as a 16-bit counter using the capture clock, and the connected counters are used for capture operation.	Used for calibration of LOCO/MOCO
Timer Array Unit channel 0 (R_TAU)	It is utilized for debouncing user switches and for differentiating between short and long presses.

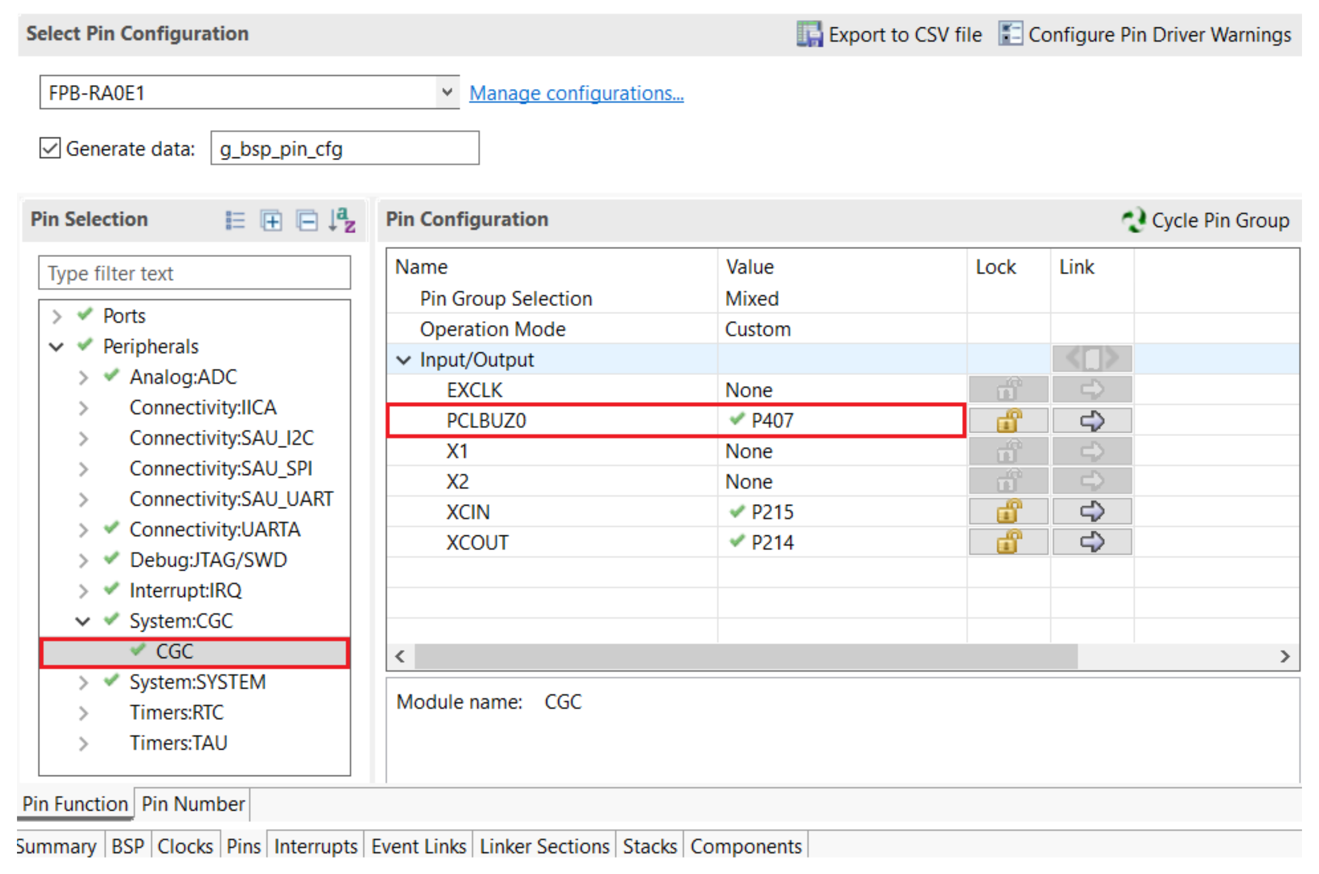
Figure 1-1 provides the selection of clock settings. Select LOCO as the FXSP source and the TML32 FITL1 source.

Figure 1-1 Example of Clock Settings



Set LOCO as the CLKOUT source, as indicated in Figure 1-1, and configure P407 as the PCLBUZ0 pin, as shown in Figure 1-2. This configuration will enable LOCO output on P407. You can use an oscilloscope to confirm the LOCO clock correction.

Figure 1-2 Example of PCLBUZ0 Pin Setup



1.2 Outline of Operation

The following outlines the settings for peripheral functions.

(1) Initialization of External Interrupt

The settings are shown in Table 1-2.

Table 1-2 Initial Setting Conditions of External Interrupt (IRQ0)

Item	Description
Valid edge	Falling edge
Pin	P200

(2) Initialization of 32-bit Interval Timer TML32 In 16-bit Capture Mode

When using the 16-bit capture mode for channels 0 and 1, the counter value is stored in the interval timer capture register 00 (ITLCAP00) upon receiving a designated capture trigger.

Refer to Figure 1-3 for the register settings for the 16-bit capture mode.

Figure 1-3 Registers and Settings Used in 16-bit Capture Mode

Register name (symbol)	Bit	Setting
Interval timer compare register 00 (ITLCMP00)	Bits 15 to 0	Specify 16-bit compare values for channels 0 and 1.
Interval timer compare register 01 (ITLCMP01)*1	Bits 15 to 0	Specify 16-bit compare values for channels 2 and 3.
Interval timer control register 0 (ITLCTL0)	EN0	Specify whether to start or stop counting in channels 0 and 1.
	EN1	Set to 0.
	EN2	Specify whether to start or stop counting in channels 2 and 3.
	EN3	Set to 0.
	MD[1:0]	Set to 01b.
Interval timer frequency division registers 0 (ITLFDIV00)	FDIV0[2:0]	Select the count clock for channel 0.
	FDIV1[2:0]	Set to 000b.
Interval timer frequency division registers 1 (ITLFDIV01)	FDIV2[2:0]	Set to 000b.
	FDIV3[2:0]	Set to 000b.
Interval timer clock select register 0 (ITLCSEL0)	ISEL[2:0]	Select the count clock for the interval timer in channels 0 and 1.
	CSEL[2:0]	Select the count clock for the interval timer for capturing in channels 2 and 3.
Interval timer capture control register 0 (ITLCC0)	CAPEN	Set to 1.
	CAPCCR	Specify whether to clear or hold the counter value in channels 0 and 1 after the completion of capturing.
	CTRS[1:0]	Select a capture trigger.

Note 1. Channels 2 and 3 can only be used in 16-bit counter mode when an interrupt on compare match with ITLCMP01 is not to be used as a capture trigger.

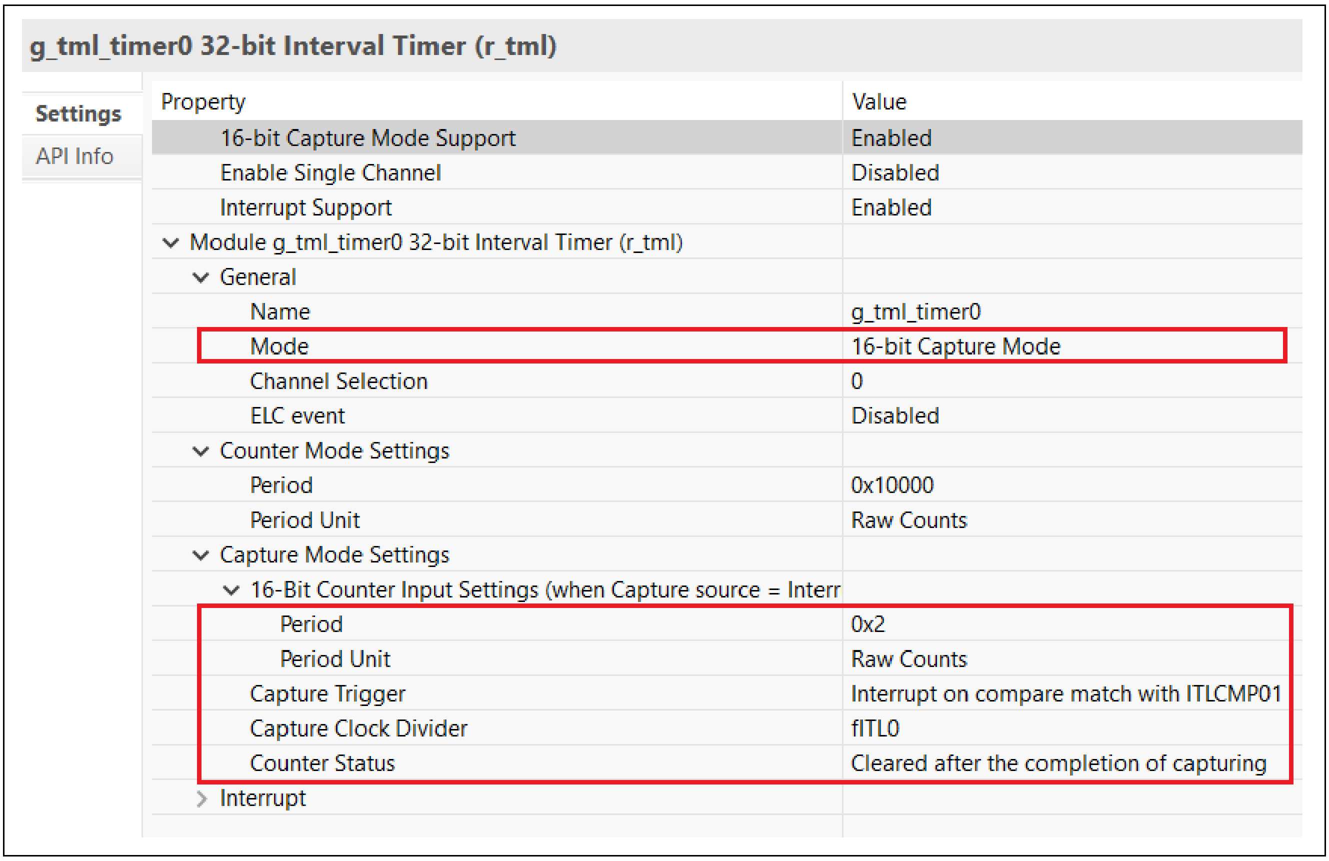
The settings in the example code are shown in Table 1-3.

Table 1-3 Initial Settings for the 32-bit Interval Timer (Configured in 16-bit Capture Mode for Channels 0 and 1)

Item	Description
Operating mode	16-bit capture mode
Operating clock	HOCO (32 MHz)
Count source division ratio	HOCO (non-divided)
Capture trigger	Interrupt on compare match with ITLCMP01
Capture clock	FSXP/MOCO
Interval of capture trigger	FSXP(LOCO): 2 raw counts/MOCO: 128 raw counts

Figure 1-4 provides an example of the TML32 timer settings using Renesas FSP.

Figure 1-4 Example of TML32 Timer Settings for LOCO Correction



(3) Initialization of Timer Array Unit

The settings are shown in Table 1-4.

Table 1-4 Initial Setting Conditions of Timer Array Unit Channel 0

Item	Description
Operating mode	Interval timer
Operating clock	CK00
Count source division ratio	None-divided
Interval period	10ms

Figure 1-5 provides an example of the Timer Array Unit settings in Renesas FSP.

Figure 1-5 Example of Timer Array Unit Settings

g_tau_debounce_timer0 Timer, Independent Channel, 16-bit and 8-bit Timer Operation (r_tau)

Settings | API Info

Property	Value
Common	
Parameter Checking	Default (BSP)
Interrupt Support	Enabled
Pin Output Support	Disabled
Pin Input Support	Disabled
Extra Input Mode Support	Disabled
8-Bit Mode Support	Disabled
Module g_tau_debounce_timer0 Timer, Independent Channel, 16-	
General	
Name	g_tau_debounce_timer0
Channel	0
Function	Interval Timer
Bit Timer Mode	16-bit timer
Operation Clock	CK00
Period	10
Period Unit	Milliseconds

1.3 Description of Calibration Methods

This section describes the two calibration methods used in this application note.

First, configure the 32-bit interval timer TML32 to operate in 16-bit capture mode for measuring the periods of the low-speed on-chip oscillator (LOCO) and the middle-speed on-chip oscillator (MOCO).

Select the interrupt that occurs on a compare-match with ITLCMP01 as the capture trigger. This trigger generates an event every 2 raw counts using the clock from the low-speed on-chip oscillator (32.768 kHz) and every 128 raw counts using the clock from the middle-speed on-chip oscillator (4 MHz).

Next, select a high-speed on-chip oscillator clock (32 MHz) for the counting clock. Set LOCO as the FXSP source. You can then select either MOCO or LOCO as the TML32 FITL1 source.

To enhance accuracy, measure the subsystem clock cycle four times. Sum the four captured values to identify any error in the frequency of the LOCO or MOCO oscillation clocks.

The target frequency ranges should be within ±2 LSB, which corresponds to ±0.6% of the LIOTRM register and ±0.3% of the MIOTRM register, respectively.

Table 1-5 presents the calculated count values for the four captures.

Table 1-5 Range of LOCO Count Values

LOCO Clock Frequency		Count Value Obtained through Four Times of Capture (Calculated Value)
32.768 kHz		7812.5
32.768 kHz – 0.6%	32.571 kHz	7859.658
32.768 kHz + 0.6%	32.965 kHz	7765.905

Table 1-5 presents the established target range for the count value, derived from four capture instances, which are set between 7766 and 7859.

When the frequency accuracy of the high-speed on-chip oscillator is within ±1.0%, the frequency accuracy of the low-speed on-chip oscillator will be within ±1.6%.

A count value of 7765 or lower signifies that the LOCO clock is operating slower than the target frequency. In contrast, a count value of 7860 or higher indicates that the LOCO clock is operating faster than the target frequency. Based on the count value, determine whether to adjust the LIOTRM register value to speed up or slow down the clock. Calibration should be performed by incrementing the LIOTRM register value by ±1. Calibration is complete when the count value falls within the target range.

Figure 1-6 provides an example of the calibration process of LOCO.

Figure 1-6 Example of LOCO Calibration

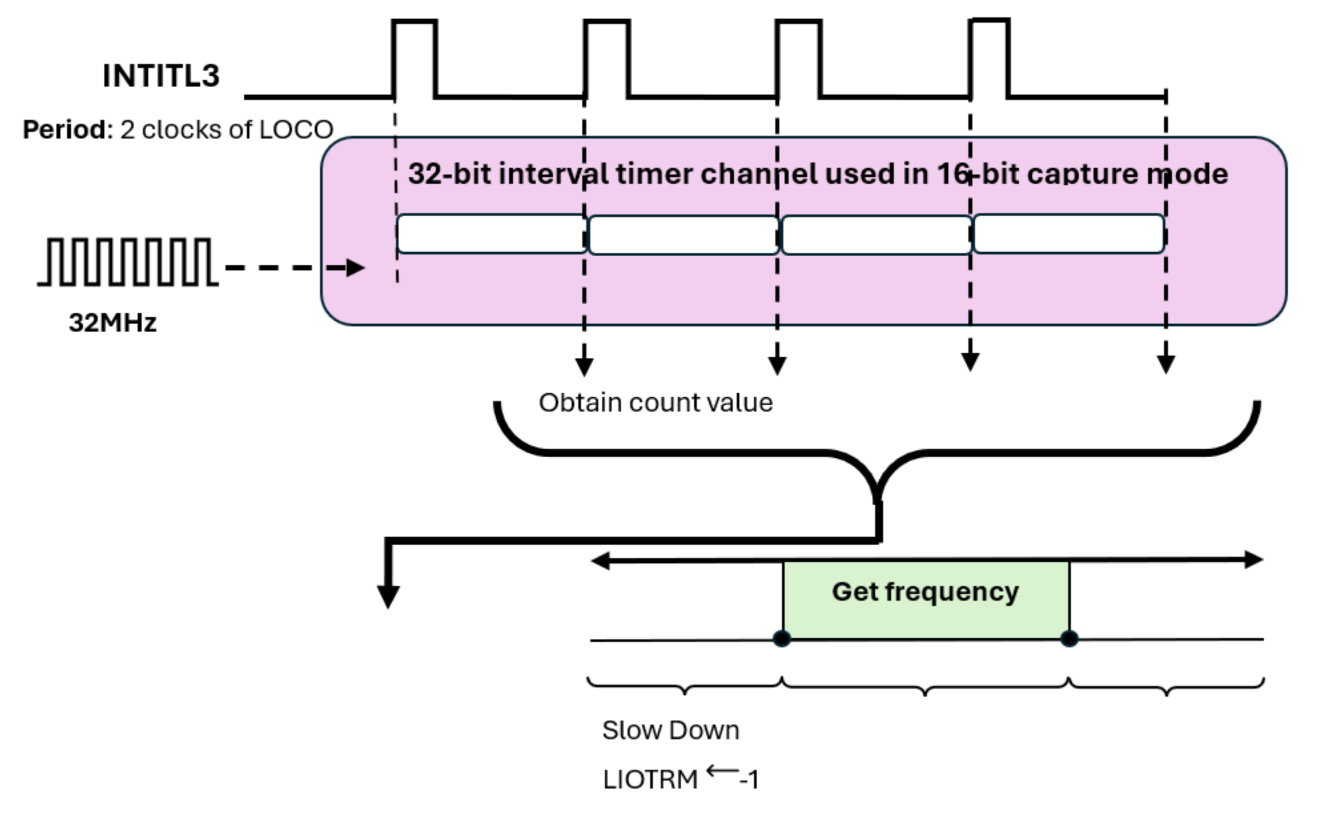


Table 1-6 Range of MOCO Count Values

MOCO Clock Frequency		Count Value Obtained through Four Times of Capture (Calculated Value)
4MHz		4096
4MHz – 0.3%	3.988MHz	4108.325
4MHz + 0.3%	4.012MHz	4083.749

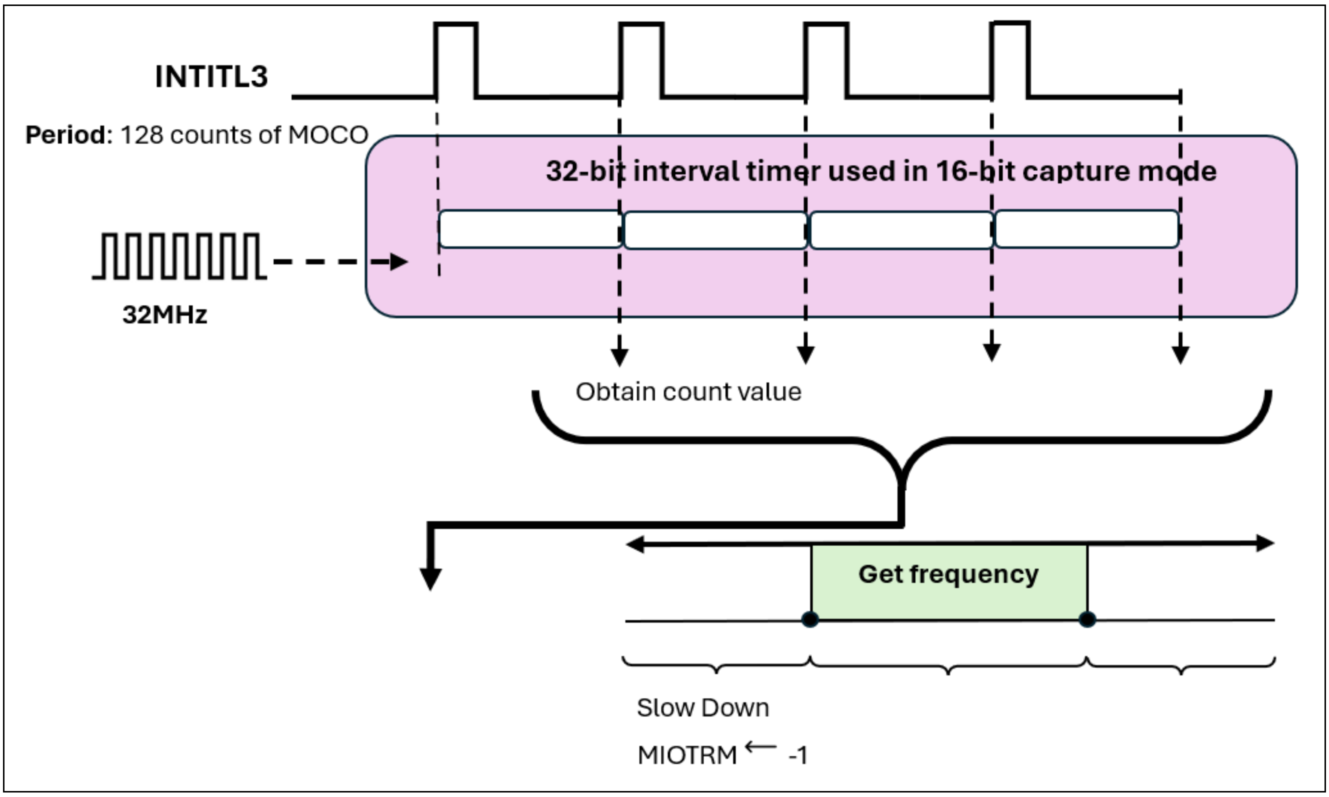
Table 1-6 presents the established target range for the count value, derived from four capture instances, which are set between 4084 and 4108.

When the frequency accuracy of the high-speed on-chip oscillator is within ±1.0%, the frequency accuracy of the medium-speed on-chip oscillator will be within ±1.3%.

A count value of 4083 or lower signifies that the MOCO clock is operating slower than the target frequency. In contrast, a count value of 4109 or higher indicates that the MOCO clock is operating faster than the target frequency. Based on the count value, determine whether to adjust the MIOTRM register value to speed up or slow down the clock. Calibration should be performed by incrementing the MIOTRM register value by ±1. Calibration is complete when the count value falls within the target range.

Figure 1-7 provides an example of the calibration process of MOCO.

Figure 1-7 Example of MOCO Calibration

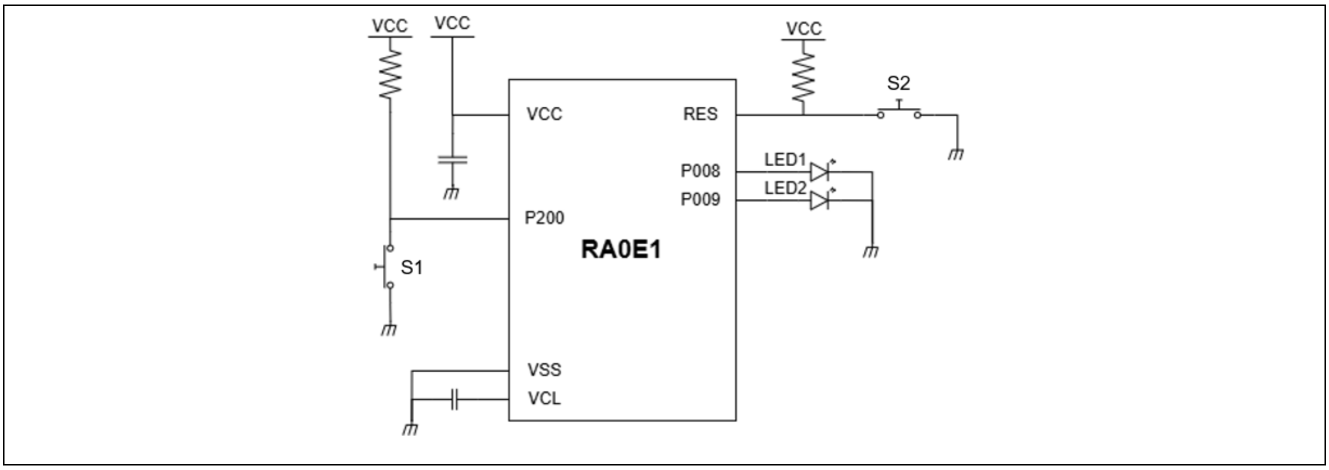


2. Hardware Descriptions

2.1 Example of Hardware Configuration

Figure 2-1 shows an example of the hardware configuration used in the application note.

Figure 2-1. Hardware Configuration



Caution: 1. The purpose of this circuit is only to provide the connection outline, and the circuit is simplified accordingly. When designing and implementing an actual circuit, provide proper pin treatment and make sure that the hardware's electrical specifications are met (connect the input-only ports separately to VDD or VSS via a resistor).

2. Connect any pins whose name begins with EVSS to VSS and any pin whose name begins with EVDD to VDD, respectively.

3. VDD must be held at not lower than the reset release voltage (VLVD) that is specified as LVD.

Table 2-1 lists the pins used and their functions.

Table 2-1 Operation Confirmation Conditions

Pin Name	I/O	Description
P200/S1	Input	User switch: Press the switch to select the correction mode (LOCO/MOCO correction) by holding it down for more than 2 seconds, then initiate the calibration by pressing it briefly.
P008/LED1	Output	LOCO correction mode LED: Indicate the status in LOCO correction mode: WAIT (blinking), START (off), and COMPLETE (solid green, on).
P009/LED2	Output	MOCO correction mode LED: Indicate the status in MOCO correction mode: WAIT (blinking), START (off), and COMPLETE (solid green, on).

Caution: In this application note, only the used pins are processed. When designing your circuit, make sure the design includes sufficient pin processing and meets electrical characteristic requirements.

3. Software Explanation

The overview of the demo system and example code is illustrated in Figure 3-1.

1. When the power is turned on, LED1 blinks to indicate LOCO correction mode. If you press the FPB switch (S1) for 2 seconds or more, LED2 will blink, signaling MOCO correction mode. Pressing S1 again for 2 seconds or more will switch the system back to LOCO correction mode.

2. When the switch is pressed briefly, the correction of the selected OCO (LOCO/MOCO) will begin. The LED will turn off during correction.

3. When the correction is complete, the LED will light up.

Figure 3-1. The Demo System and Sample Code Overview

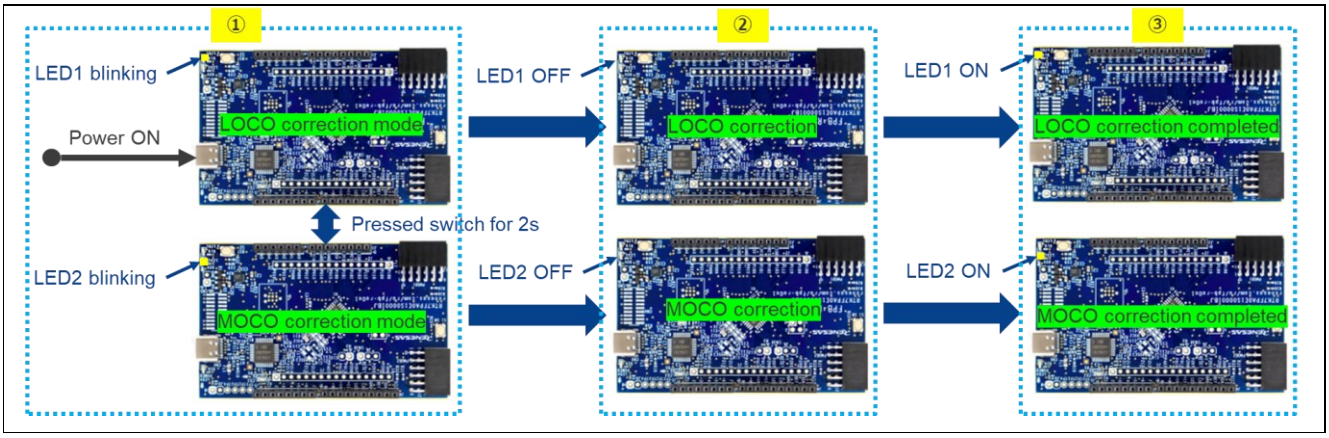


Figure 3-2 illustrates the primary loop of the example project, responsible for processing user selections and initiating the correction process.

Figure 3-2. A Snapshot of the Main Loop in the Example Code

```
/* Main loop */
while (1)
{
    /* Toggle user LEDs: LOCO/MOCO correction mode */
    led_blinky (leds, oco_type, oco_sts);
    /* User select OCO trimming*/
    switch (g_sw_press)
    {
        case SHORT_PRESS: /* Start calibration process */
        {
            /* Set OCO process status */
            oco_sts = START;
            /* Start the trimming process*/
            R_Main_Calibrate_OCO(oco_type);
            /* Set OCO process status */
            oco_sts = COMPLETE;

            /* Clear user sw status */
            g_sw_press      = NO_PRESS;
        }
        break;
        case LONG_PRESS: /* Change OCO correction mode*/
        {
            /* Turn off current user LED */
            err = R_IOPORT_PinWrite(&g_ioport_ctrl, (bsp_io_port_pin_t)leds.p_leds[oco_type], BSP_IO_LEVEL_LOW);
            /* Handle error */
            APP_ERR_TRAP(err);
            /* Change OCO trimming type */
            oco_type ^= MOCO;
            /* Set OCO process status */
            oco_sts = WAIT;

            /* Clear user sw status */
            g_sw_press      = NO_PRESS;
        }
        break;
        case NO_PRESS:
        default:
            break;
    }
}
```

Start MOCO before detecting an error and making corrections; it stops automatically by default. Figure 3-3 illustrates the code needed to initiate MCO.

Figure 3-3. Example Code to Start MOCO

```
fsp_err_t MOCO_Start(void)
{
    fsp_err_t err     = FSP_SUCCESS;

    /* Unlock CGC and LPM protection registers. */
    R_SYSTEM->PRCR    = (uint16_t) BSP_PRV_PRCR_UNLOCK;

    /* Start MOCO */
    R_SYSTEM->MOCOCR = (uint8_t)(0x00);

    /* Lock CGC and LPM protection registers. */
    R_SYSTEM->PRCR    = (uint16_t) BSP_PRV_PRCR_LOCK;

    /* Give some delays for oscillation stabilization */
    R_BSP_SoftwareDelay(1U, BSP_DELAY_UNITS_MICROSECONDS);

    return  err;
}
```

3.1 List of Primary Constants

Table 3-1 lists primary constants that are used in the example code.

Table 3-1 Primary Constants in the Example Code

Constant Name	Setting Value	Description
LMIOTRM_MAX	11111111B	Maximum value of the LIOTRM/ MIOTRM register
LMIOTRM_MIN	00000000B	Minimum value of the MIOTRM/ LIOTRM register
CCNT_HOCO_LOCO_MAX	7860	Upper threshold of clock count for LOCO correction
CCNT_HOCO_LOCO_MIN	7765	Lower threshold of clock count for LOCO correction
CCNT_HOCO_MOCO_MAX	4108	Upper threshold of clock count for MOCO correction
CCNT_HOCO_MOCO_MIN	4084	Lower threshold of clock count for MOCO correction

3.2 List of Primary Functions

Table 3-2 shows a list of primary functions.

Table 3-2 Primary Functions in the Example Code

Function Name	Outline
R_Main_Calibrate_OCO	Calibration process
R_Trimming_OCO	Perform LOCO/MOCO clock correction

3.3 Primary Function Flowcharts

The OCO correction process includes two functions: R_Trimming_OCO and R_Main_Calibrate_OCO. Further details about these functions will be outlined in the upcoming sections.

This example code initializes the TML32 module using FSP configuration and APIs. However, most of the LOCO and MOCO frequency detection and correction procedures rely on direct register access. This approach enables the frequency correction function to achieve more deterministic results.

3.3.1 Calibration Process

Figure 3-4 shows the flowchart of the calibration process.

Figure 3-4. Flowchart of LOCO/MOCO Calibration Process(1/2)

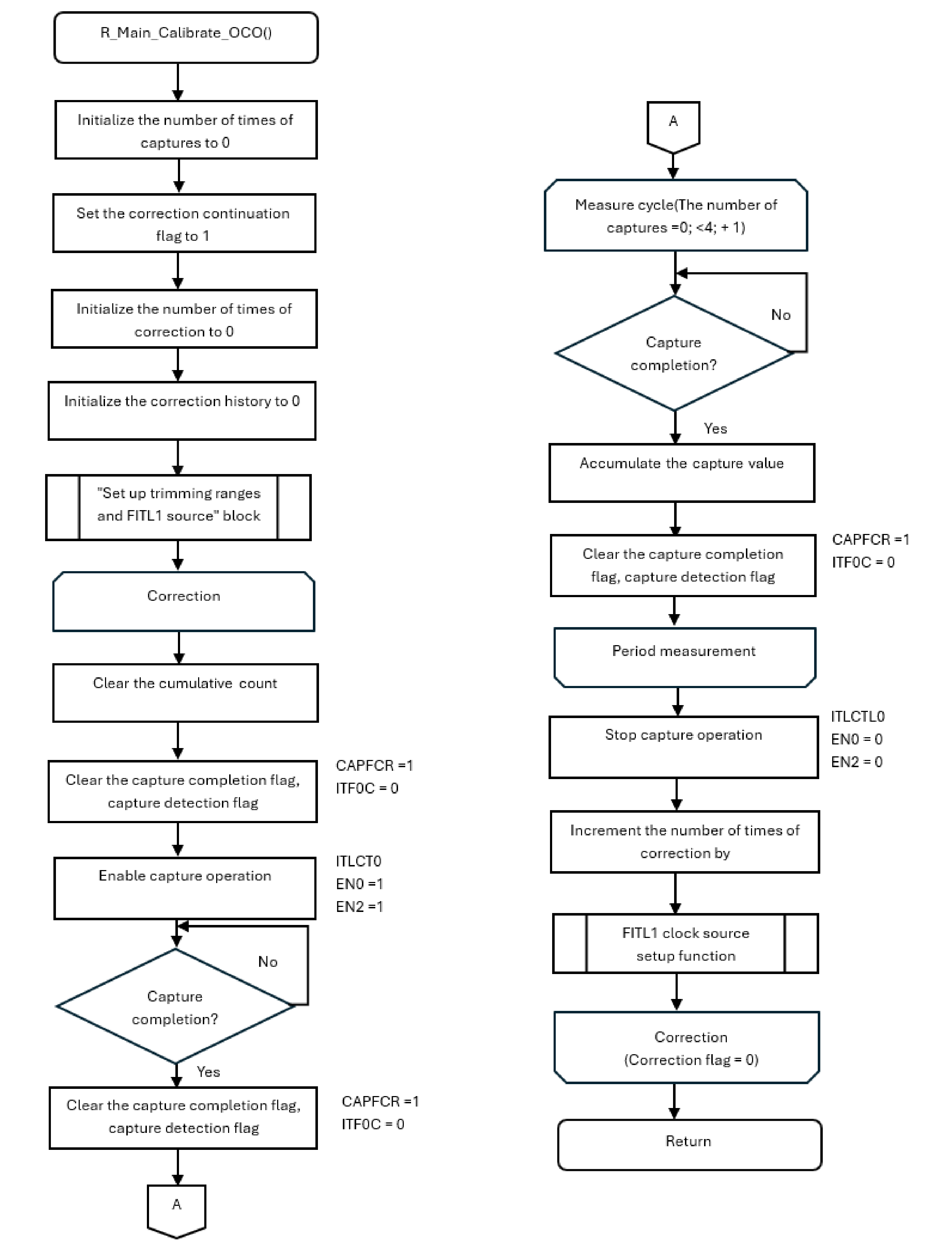
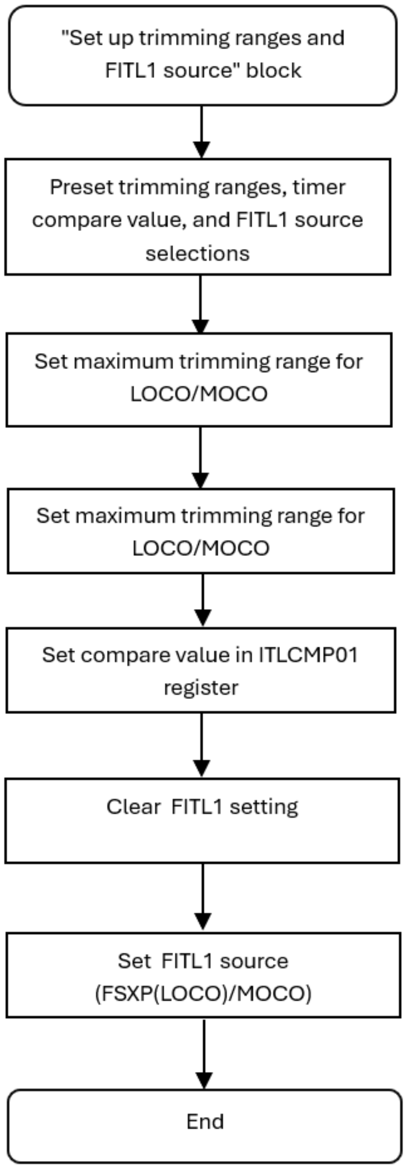


Figure 3-5. Flowchart of LOCO/MOCO Calibration Process(2/2)



3.3.2 LOCO and MOCO Correction Function

Figure 3-6, Figure 3-7, and Figure 3-8 present the flowcharts for the LOCO and MOCO correction function.

Figure 3-6. LOCO/MOCO Correction Function (1/3)

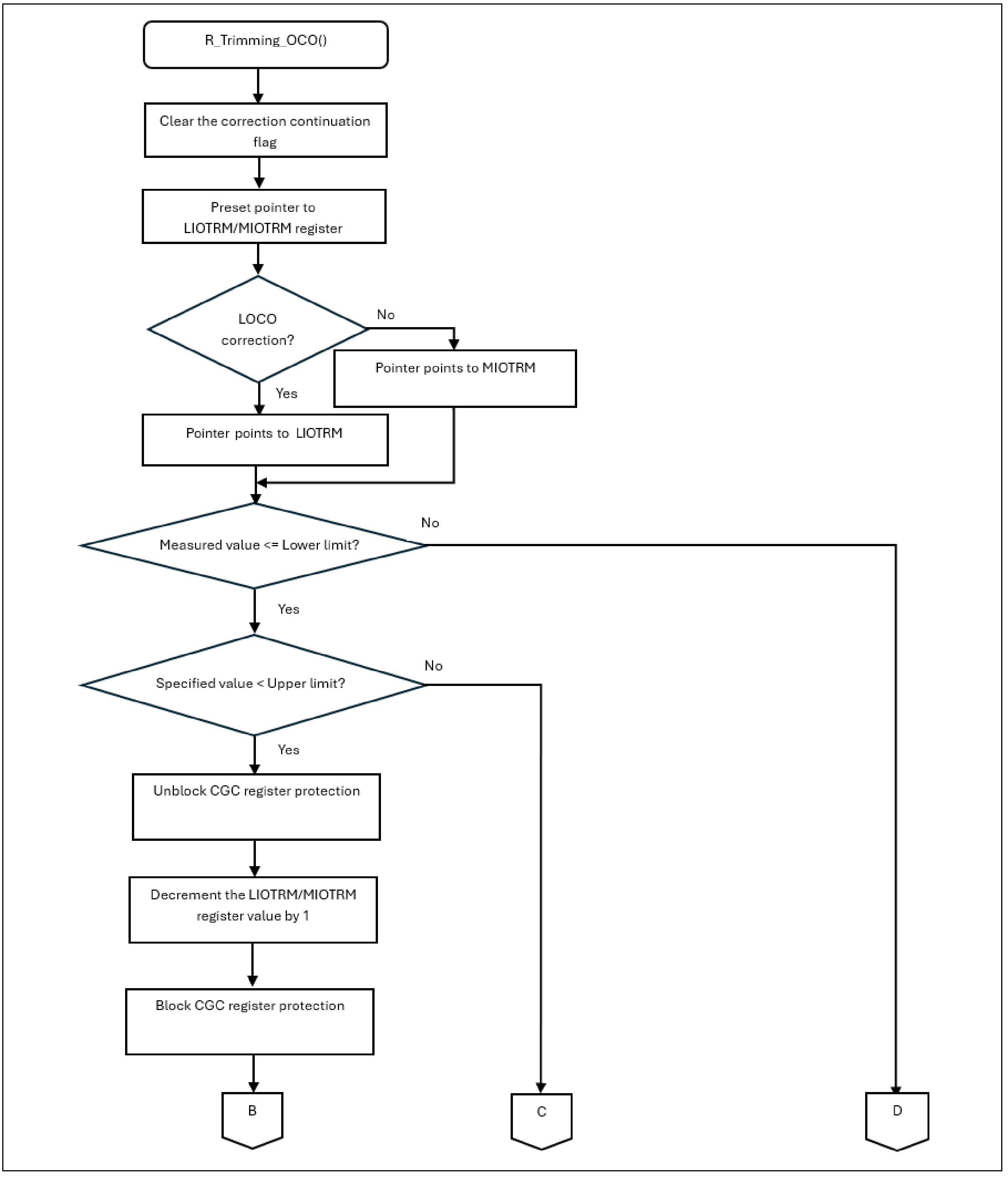


Figure 3-7. LOCO/MOCO Correction Function (2/3)

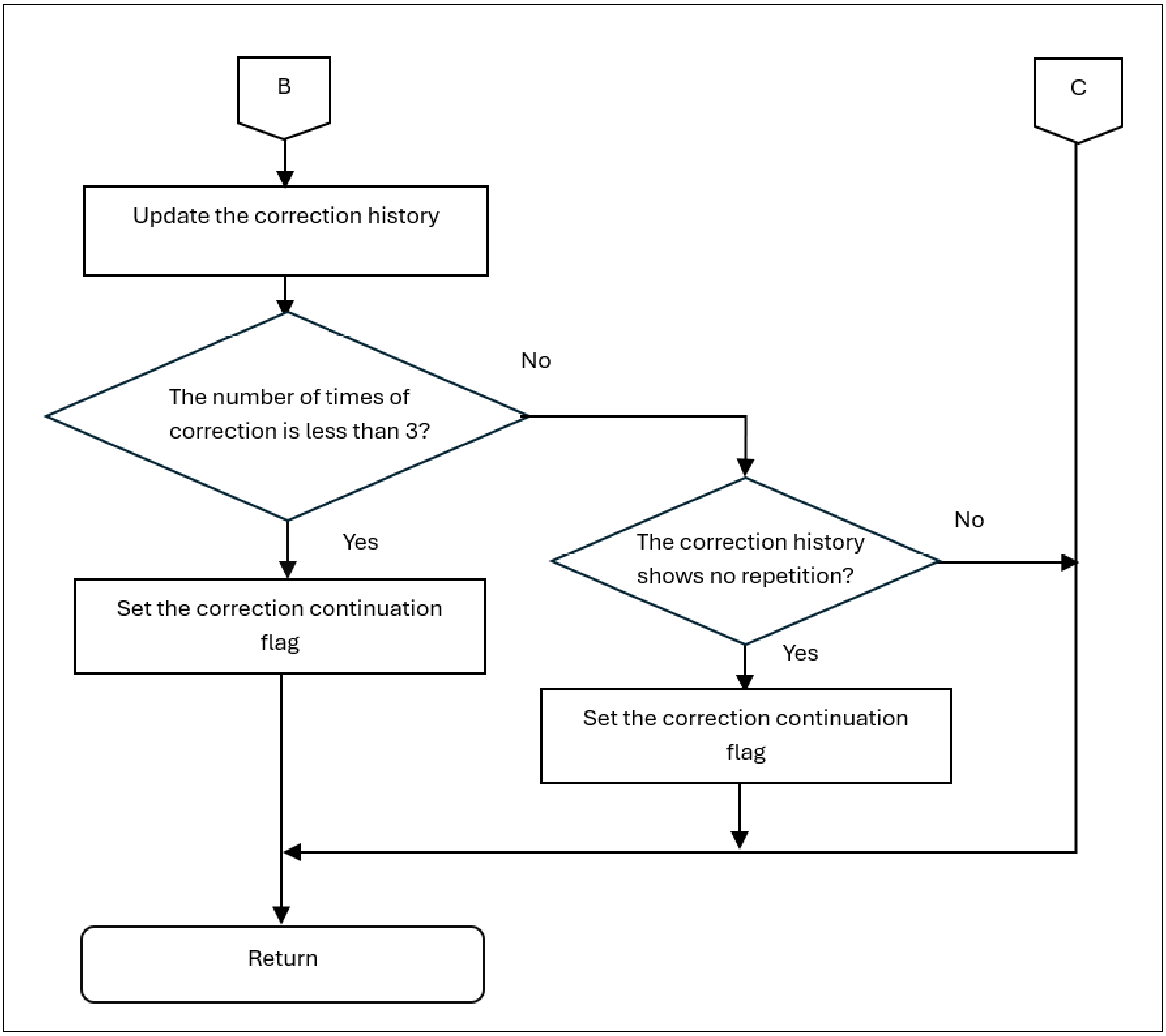
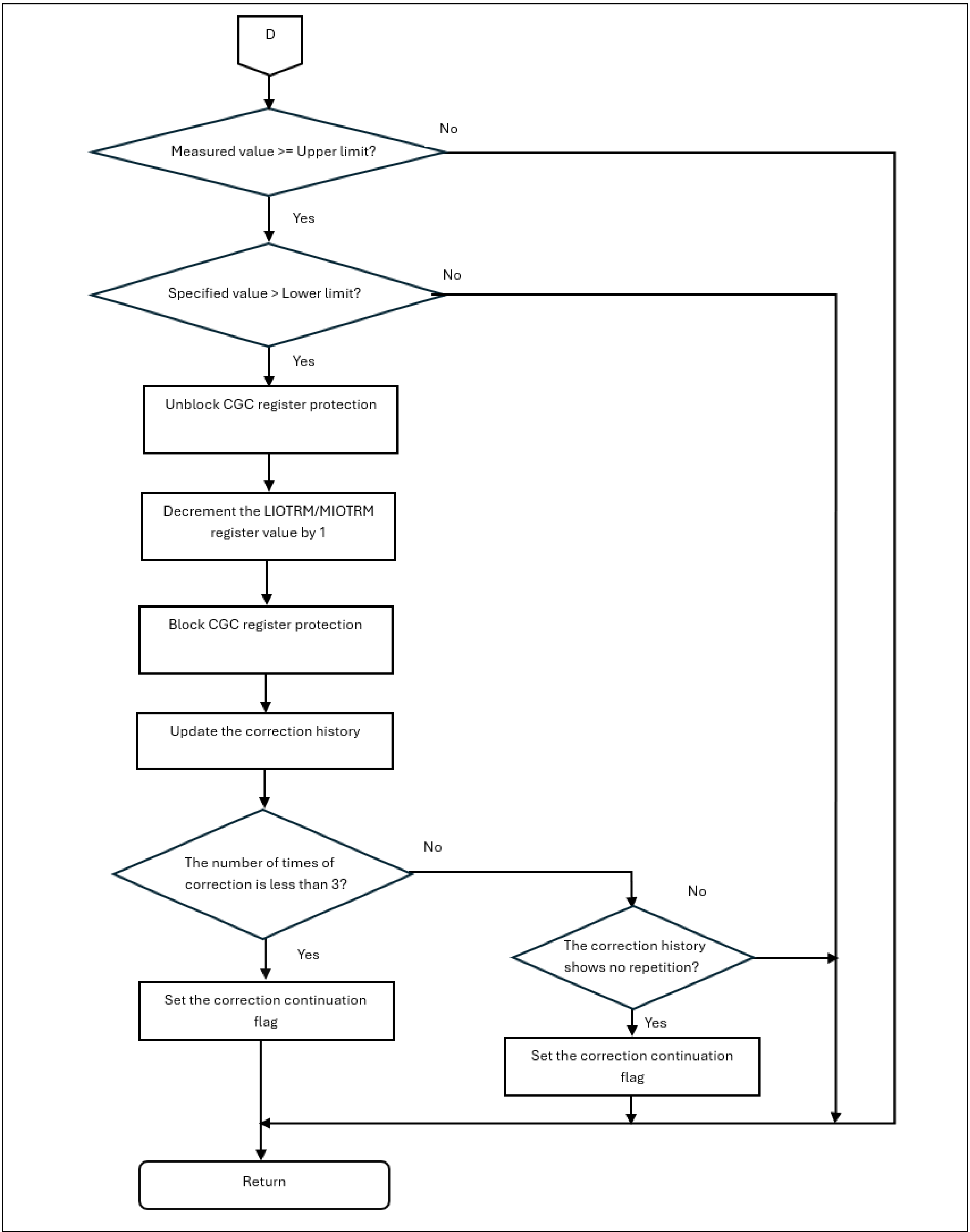


Figure 3-8. LOCO/MOCO Correction Function (3/3)

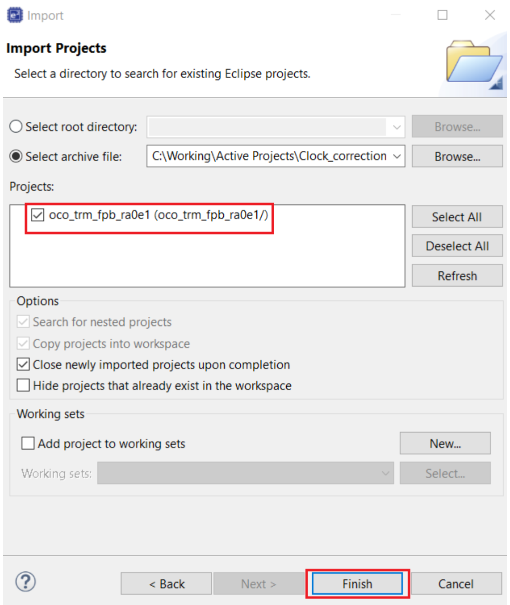


4. Verify The Example Project

4.1 Import The Project

1. Launch e²studio IDE.
2. Select any workspace in Workspace launcher.
3. Close the **Welcome** window.
4. Select **File > Import**.
5. Select **Existing Projects into Workspace** from the **Import** dialog box.
6. FPB-RA0E1, select archive file “oco_trm_fpb_ra0e1.zip”; for FPB-RA0E2, select archive file “oco_trm_fpb_ra0e2.zip”. Both are archived in the file named oco_trm_fpb_ra0e.zip.
7. Select both developed project samples on each core as shown below, click **Finish**

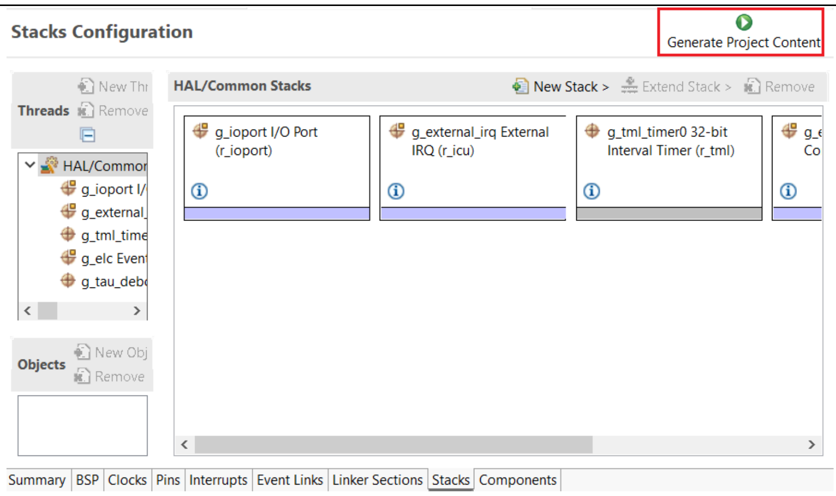
Figure 4-1. Import Application Project into Workspace



4.2 Build Project

Launch configuration.xml in the project and click “Generate Project Content” to generate project content.

Figure 4-2. Example of Generating Project Content



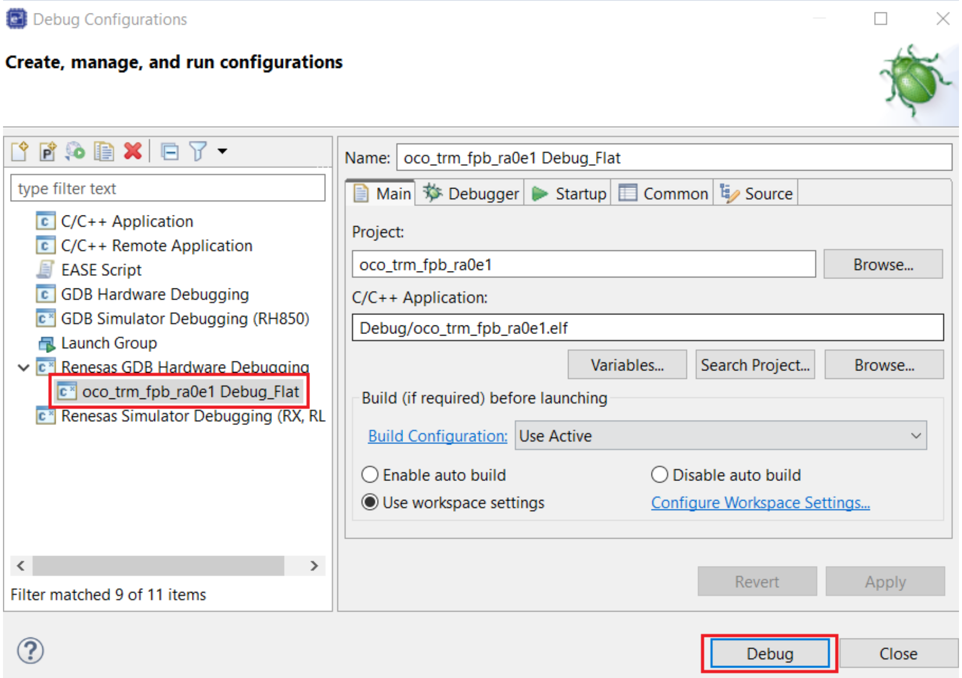
After generating the project content, right-click on the project and select "Build Project."

Ensure that the building is successful by checking the building log console.

4.3 Download and Run Project

Connect CN6 on the FPB-R0E1 board to the USB port on your PC. From the menu, go to **Run**>**Debug Configurations** and select oco_trm_fpb_ra0e1 Debug_Flat to launch the project.

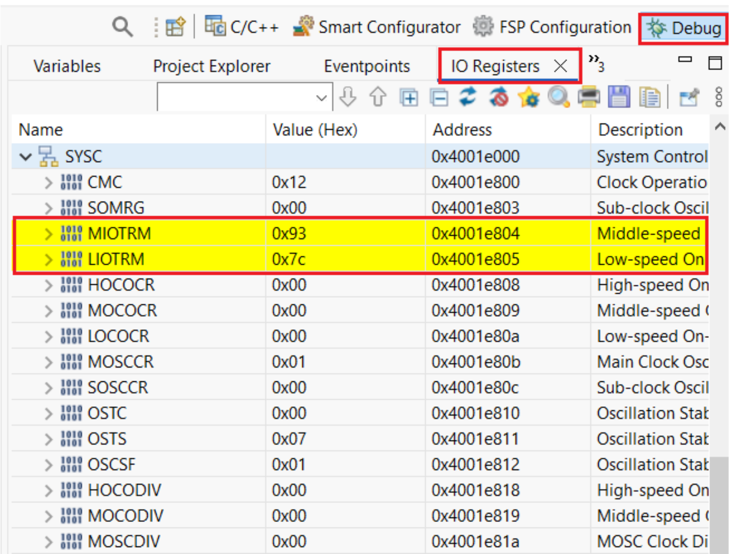
Figure 4-3. Example of Starting Debugging Example Project



4.4 Verify Frequency Correction Result

Refer to Software Explanation for detailed instructions on operating the example project. While debugging the example project, use the IO Registers tab to check the register values for LIOTRM and MIOTRM. Make sure to compare these register values with the default settings in the MCU User Manual.

Figure 4-4. Example of LIOTRM and MIOTRM Registers after Frequency Correction



Since LOCO serves as the output source for P407, its signal can be captured with an oscilloscope to verify the frequency after correction compared to the frequency before correction, as illustrated in Figure 4-5 and Figure 4-6.

Figure 4-5. Example of LOCO Output on P407 after Frequency Correction

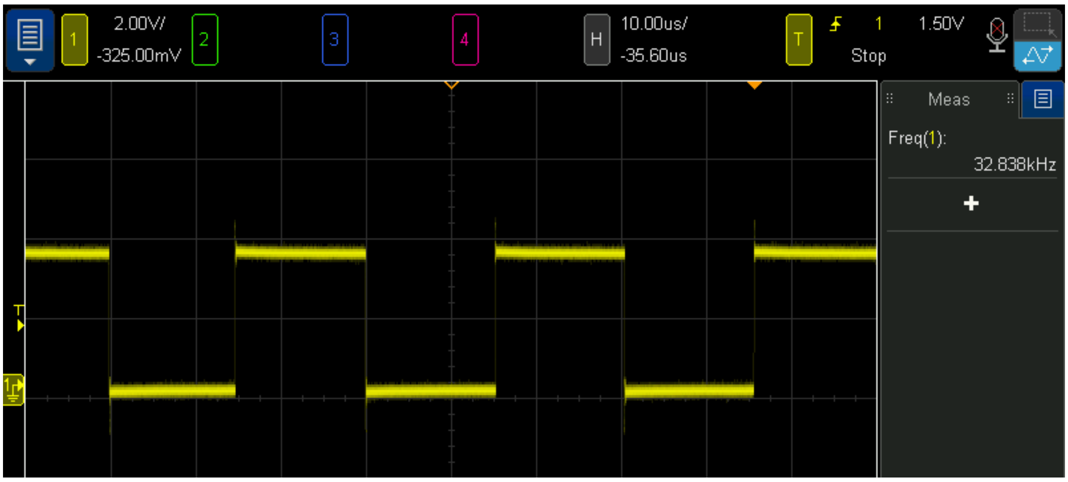
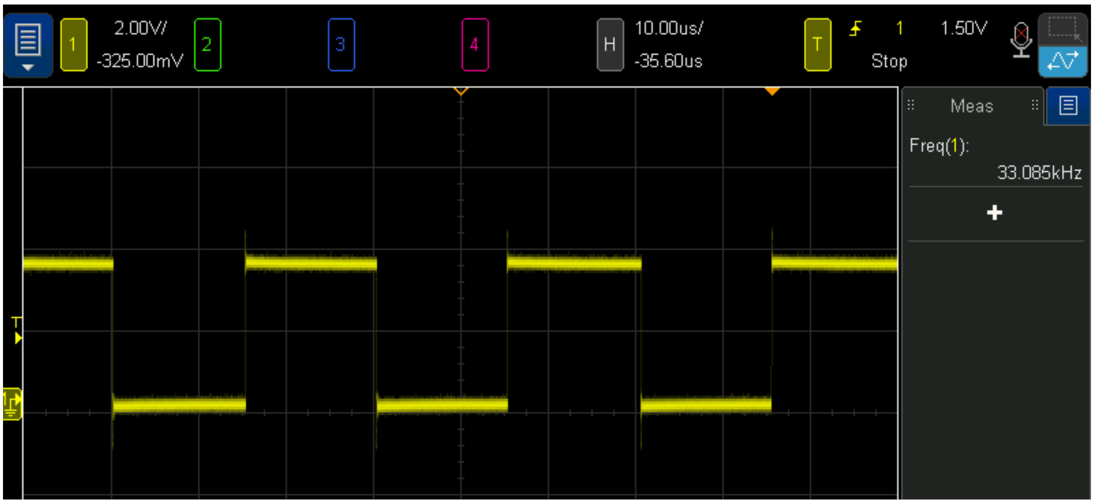


Figure 4-6. Example of LOCO Output on P407 before Frequency Correction



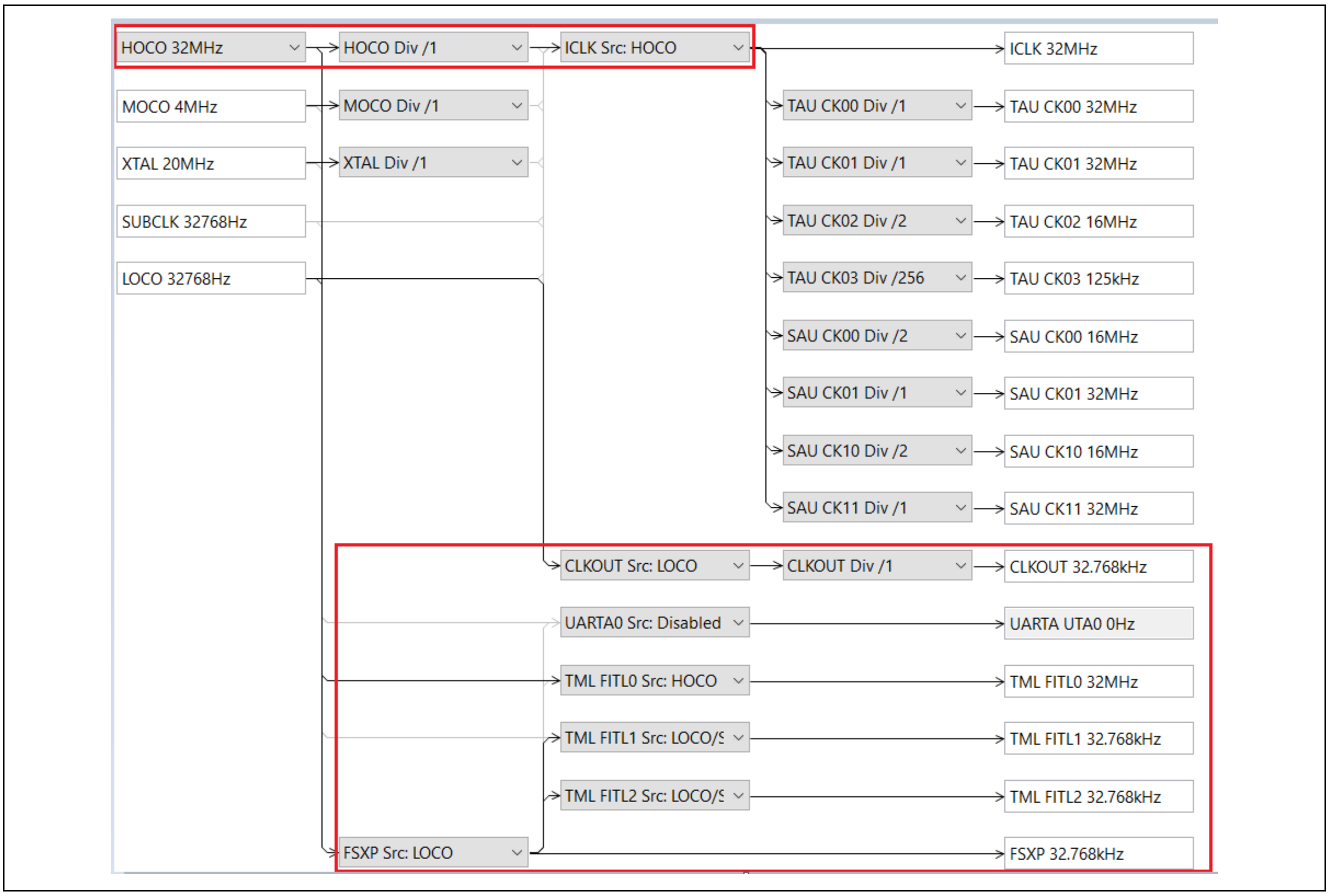
5. Porting the Example Project to a Different RA0 MCU

The process for porting the LOCO and MOCO trimming example project (oco_trm_fpb_ra0e1) to another RA0 MCU, which includes a Timer Array Unit (TAU) and a 32-bit Interval Timer (TML32), is outlined below.

(1) Setting up clock configuration, as shown in Figure 5-1

- HOCO 32MHz is the ICLK and Peripheral clock.
- LOCO is FSXP source.
- HOCO is the TML FITL0 source.
- LOCO/FSXP is the TML FITL1 source.
- LOCO/FSXP is the TML FITL2 source.

Figure 5-1. Example of RA0 MCU Clock Configuration



(2) Change the FSP configuration for the user switch button, as shown in Figure 5-2, Figure 5-3 and Figure 5-4

- Update the IRQ pin in the Pins configuration.
- Update the configuration for the r_icu module.
- Modify the IRQ number and the macro definition for the pin associated with the user switch.

Figure 5-2. Example of Pin Configuration for the User Switch

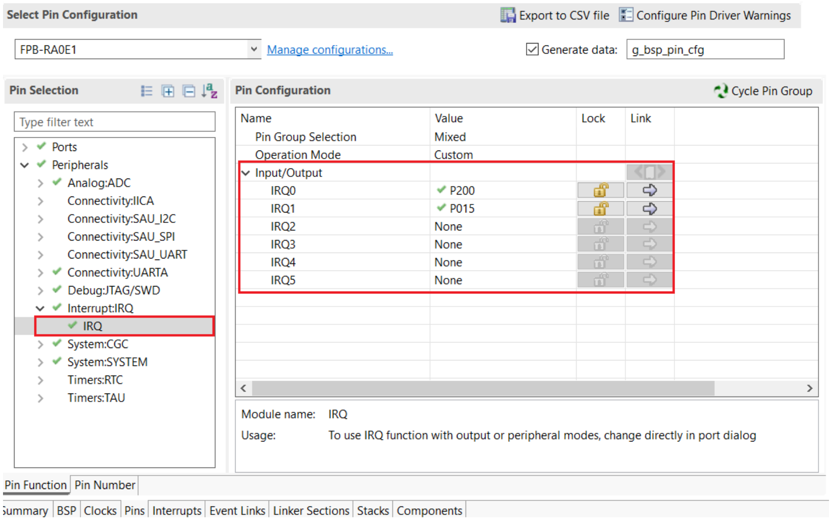
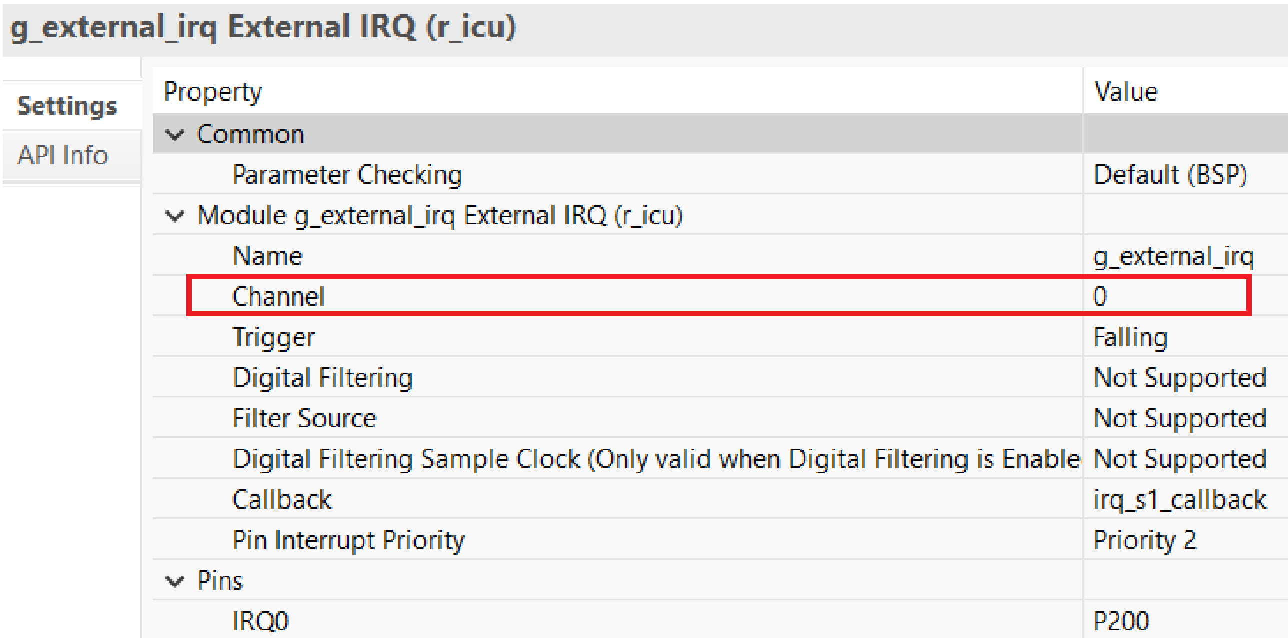


Figure 5-3. Example of r_icu Settings



Property	Value
Common	
Parameter Checking	Default (BSP)
Module g_external_irq External IRQ (r_icu)	
Name	g_external_irq
Channel	0
Trigger	Falling
Digital Filtering	Not Supported
Filter Source	Not Supported
Digital Filtering Sample Clock (Only valid when Digital Filtering is Enable	Not Supported
Callback	irq_s1_callback
Pin Interrupt Priority	Priority 2
Pins	
IRQ0	P200

Figure 5-4. A Snapshot of Macro Definitions for the User Switch

```
/* External IRQ channel for specific boards */
#define USER_SW_IRQ_NUMBER              (0x00)

#define USER_SW_DEBOUNCE_LIMIT          (10U)
#define USER_SW_LONG_PRESS_THRESHOLD    (300U)
#define USER_SW_PIN                     BSP_IO_PORT_02_PIN_00
```

(3) Create a new pin assignment for the LEDs if needed, as shown in Figure 5-5

Figure 5-5. A Snapshot of Pin Definitions for the LEDs

```
/** Array of LED IOPORT pins. */
static const uint16_t g_bsp_prv_leds[] =
{
    (uint16_t) BSP_IO_PORT_00_PIN_08,   ///< LED1
    (uint16_t) BSP_IO_PORT_00_PIN_09,   ///< LED2
};
```

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	June.30.25	-	Initial version
		-	

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/.