RX671 Group

Using FLL to Boost HOCO Frequency Accuracy

Introduction

This application note explains the use of the frequency locked loop (FLL) function to boost the frequency accuracy of the high-speed on-chip oscillator (HOCO) and describes the provided sample code that uses the clock frequency accuracy measurement circuit (CAC) to measure the HOCO frequency with the FLL enabled and confirm that the frequency error is within the acceptable range.

Target Device

RX671 Group

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

Contents

1. Specifications

Enabling the FLL boosts the HOCO frequency accuracy by using the subclock to apply correction to the HOCO. The sample program accompanying this application note confirms that the HOCO frequency error is within the acceptable range before using the HOCO with FLL enabled for the main processing. In addition, this application note shows how to use the CAC to measure the HOCO frequency.

2. Operation Confirmation Environment

2.1 Operation Confirmation Conditions

The operation of the sample code accompanying this application note has been confirmed under the conditions listed in Table 2.1.

Table 2.1 Operation Confirmation Conditions

Item	Description
MCU used	R5F5671EHDFB (RX671 Group)
Operating frequencies	• HOCO: 20 MHz • Subclock: 32.768 kHz Oscillator drive capacity (standard CL) • System clock (ICLK): 20 MHz (HOCO × 1)
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics e² studio Version: 2021-07
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.03.00
	Compile option Default settings of integrated development environment
BSP module version	Version 6.11
CAC module version	Version 1.9.0
iodefine.h version	Version 1.00
Endian order	Little endian
Operating mode	Single-chip mode
Processor mode	Supervisor mode
Sample code version	Version 1.00
Board used	Renesas Starter Kit+ for RX671 (product No.: RTK55671EDC10000BJ)

3. Related Application Note

An application note related to this application note is listed below. Refer to it in conjunction with this document.

RX Family Board Support Package Module Using Firmware Integration Technology (R01AN1685EJ)

4. Software

4.1 Operation Overview

The sample program accompanying this application note uses code output by Smart Configurator.

The BSP clock_source_select function is used to make FLL settings. After this, the CAC is used to measure the HOCO frequency and the frequency accuracy is checked. Table 4.1 lists the clocks used by the CAC.

Frequency measurement starts after a reset and is performed four times consecutively. CAC_BUSY is stored in the variable g_cac_status while measurement is taking place.

The mean of the four measurements is used as the measurement result for confirming whether the frequency error is within the acceptable range. The program described in this application note uses “within ±0.25% of the clock frequency” as the acceptable range for the HOCO frequency. The method used to check whether the frequency error is within the acceptable range is to compare the measurement result to the values set in the CAC upper-limit value setting register (CAULVR) and CAC lower-limit value setting register (CALLVR).

The clock frequency error is within the acceptable range when CALLVR register value ≤ measurement result ≤ CAULVR register value, so such a result is judged to be normal. In this case, CAC_OK is stored in the variable g_cac_status.

The clock frequency error is outside the acceptable range when measurement result < CALLVR register value or measurement result > CAULVR register value, so such a result is judged to be abnormal. In this case, CAC_ERROR is stored in the variable g_cac_status.

Table 4.1 Clocks Used by CAC

Clock	Frequency
Measurement reference clock	0.000032 MHz (subclock (32.768 kHz) × 1/1,024)
Measurement target clock	0.625 MHz (HOCO (20 MHz) × 1/32)

4.2 BSP Settings

Table 4.2 and Figure 4.1 show the BSP settings specified in Smart Configurator. Note that the default values are used for settings not listed in Table 4.2.

Table 4.2 Modified Clock Settings

Clock	Frequency	Setting Value
Subclock	Frequency	32.768 kHz
	Oscillator drive capacity	Standard CL
HOCO	Frequency	20 MHz
	Enable FLL function	Enabled
System clock	Frequency	20 MHz
	Clock source	HOCO × 1

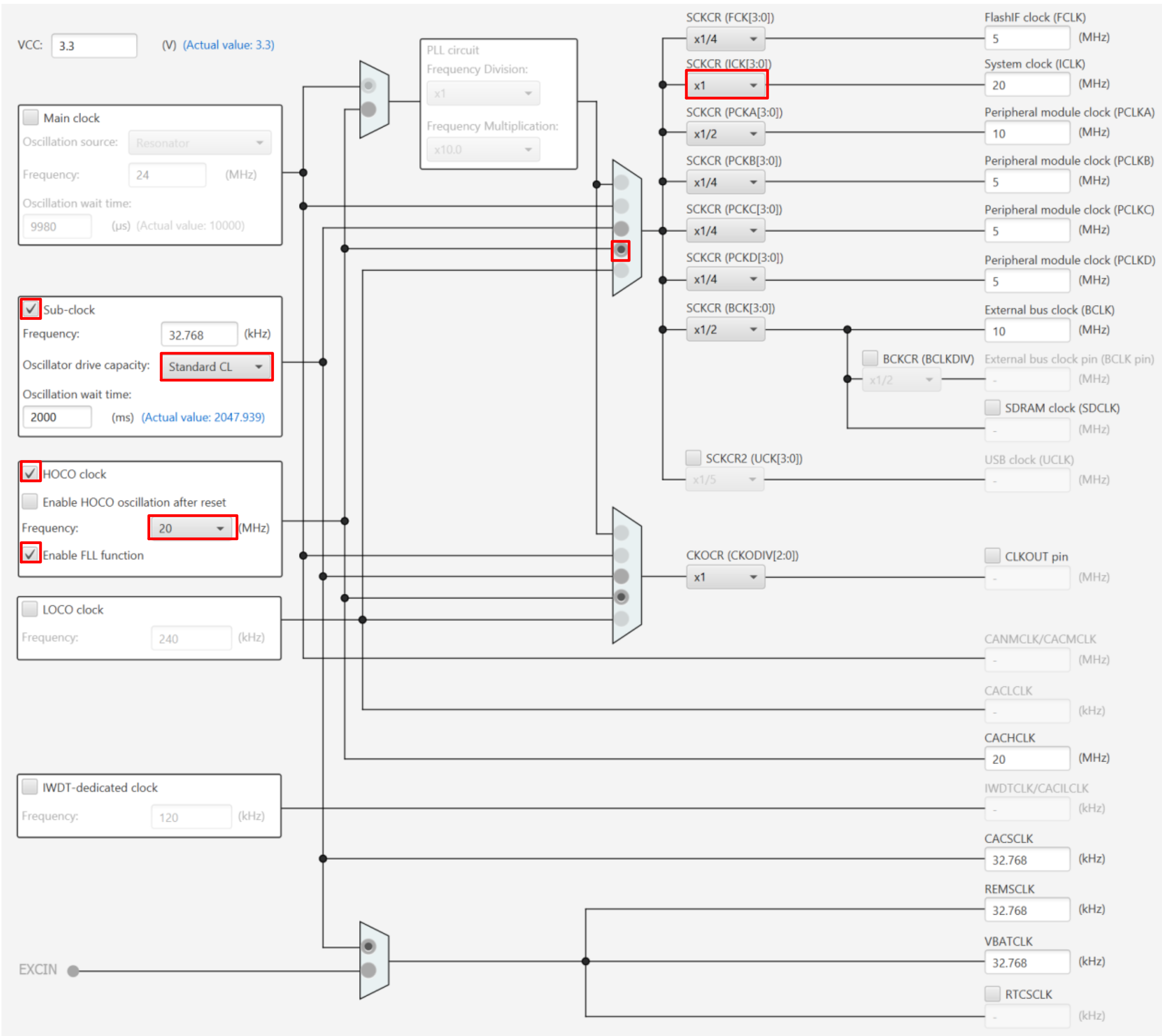


Figure 4.1 Clock Settings in Smart Configurator

4.3 CAC Settings

Table 4.3 and Figure 4.2 show the CAC settings specified in Smart Configurator.

Table 4.3 Modified CAC Settings

Main Item	Subitem	Setting Value
Measurement reference setting	Clock select	Sub-clock
	Frequency	1/1,024 (0.000032 MHz)
	Digital filter selection	Disabled
	Valid edge	Rising
Measurement target setting	Clock select	HOCO clock
	Frequency	1/32 (0.625 MHz)
Counter value comparison setting	Maximum positive deviation	0.25%
	Maximum negative deviation	0.25%
Interrupt setting	Enable frequency error interrupt (FERRI)	Unchecked
	Enable measurement end interrupt (MENDI)	Level 15 (highest)
	Enable overflow interrupt (OVFI)	Unchecked

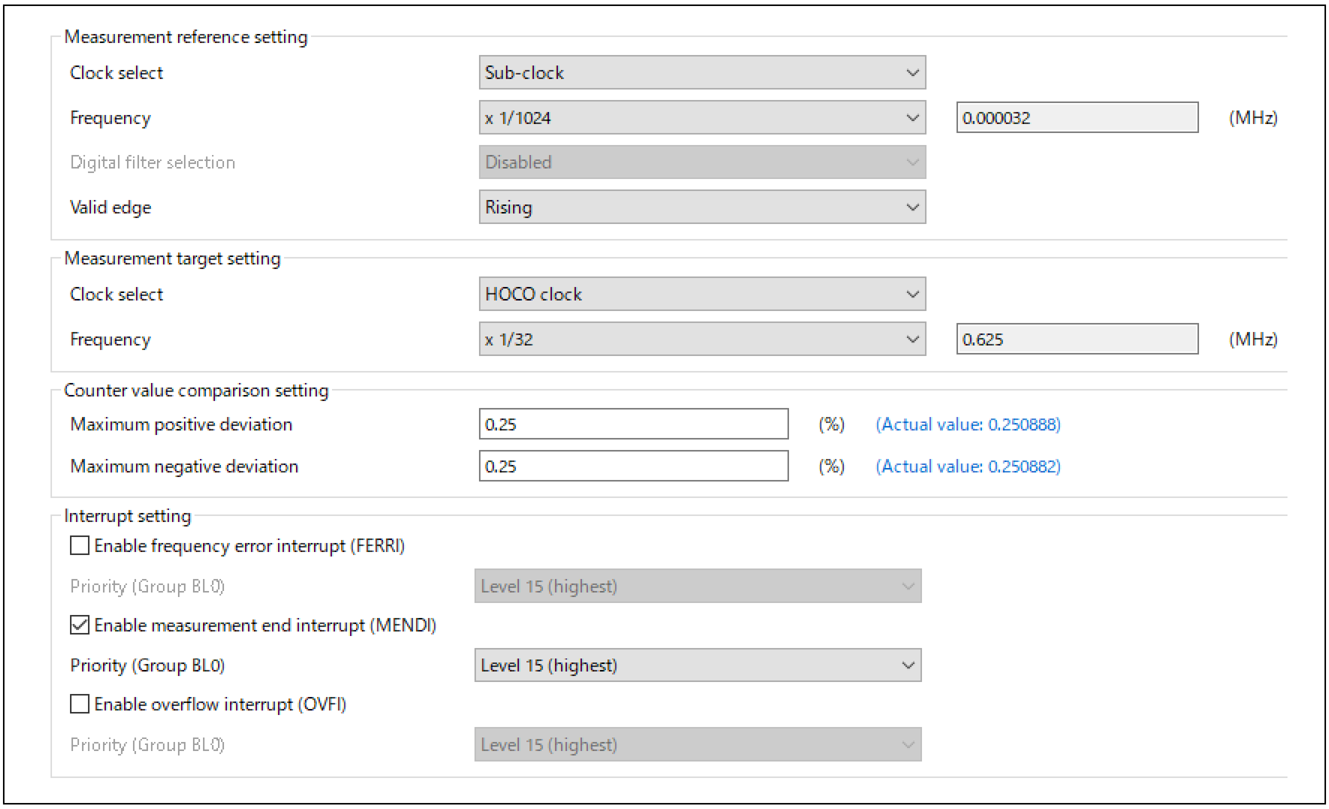


Figure 4.2 CAC Settings in Smart Configurator

4.4 File Structure

Table 4.4 lists the file containing the sample code. Note that files generated by the integrated development environment are omitted.

Table 4.4 Sample Code File

File Name	Description	Remarks
main.c	Main processing	

4.5 Option Setting Memory

Table 4.5 lists the status of the option setting memory used by the sample code. Settings should be optimized for the user system as necessary.

Table 4.5 Option Setting Memory Used by Sample Code

Symbol	Address	Setting Value	Description
MDE	FE7F 5D03h to FE7F 5D00h	FFFF FFFFh	Little-endian
OFS0	FE7F 5D07h to FE7F 5D04h	FFFF FFFFh	Voltage monitor 0 reset disabled after a reset HOCO oscillation disabled after a reset
OFS1	FE7F 5D0Bh to FE7F 5D08h	FFFF FFFFh	IWDT stopped after a reset

4.6 Constants

Table 4.6 and Table 4.7 list the constants used in the sample code.

Table 4.6 Constants Used in Sample Code (Modifiable by User)

Constant Name	Setting Value	Description
CHECK_CNT	4	Frequency measurement count*1

Note: 1. Only set a value between 1 and 255.

Table 4.7 Constants Used in Sample Code (Not Modifiable by User)

Constant Name	Setting Value	Description
CAC_ERROR	0x00	Frequency measurement result is abnormal.
CAC_OK	0x01	Frequency measurement result is normal.
CAC_BUSY	0x02	Frequency measurement is in progress.

4.7 Variables

Table 4.8 and Table 4.9 list the global variables used in the sample code.

Table 4.8 Global Variables Used in Sample Code

Type	Variable Name	Description	Used by Function
volatile uint8_t	g_cac_status	Frequency measurement status 0x00: Frequency measurement result is abnormal. 0x01: Frequency measurement result is normal. 0x02: Frequency measurement is in progress.	R_CAC_Start r_Config_CAC_mendi_interrupt

Table 4.9 Static Variables Used in Sample Code

Type	Variable Name	Description	Used by Function
static uint8_t	s_measure_counter	Frequency measurement counter	R_CAC_Start r_Config_CAC_mendi_interrupt
static uint32_t	s_result_buffer	Frequency measurement result storage buffer	R_CAC_Start r_Config_CAC_mendi_interrupt
static uint32_t	s_result_average	Mean frequency measurement result storage buffer	R_CAC_Start r_Config_CAC_mendi_interrupt

4.8 Functions

Table 4.10 lists the functions used in the sample code.

Table 4.10 Functions

Function Name	Description	Contained in File
main	Main processing	main.c
R_CAC_Start	Clock frequency measurement start	Config_CAC_user.c

4.9 Function Specifications

The sample code function specifications are listed below.

main	
Outline	Main processing
Header	None
Declaration	void main (void)
Description	This function starts clock frequency measurement.
Arguments	None
Return value	None

R_CAC_Start	
Outline	Clock frequency measurement start processing
Header	Config_CAC.h
Declaration	void R_CAC_Start (void)
Description	Performs processing to start clock frequency measurement.
Arguments	None
Return value	None

r_Config_CAC_mendi_interrupt	
Outline	CAC measurement end interrupt handler
Header	Config_CAC.h
Declaration	void r_Config_CAC_mendi_interrupt (void)
Description	Judges whether the HOCO’s frequency error is within the acceptable range based on the measurement result.
Arguments	None
Return value	None

4.10 Flowcharts

4.10.1 Main Processing

Figure 4.3 is a flowchart of the main processing.

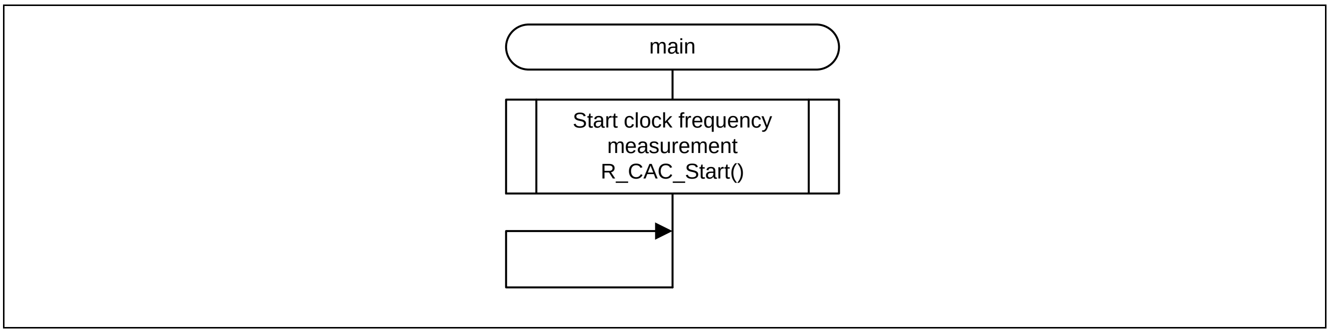


Figure 4.3 Main Processing

4.10.2 CAC Measurement Start Processing

Figure 4.4 is a flowchart of the CAC measurement start processing.

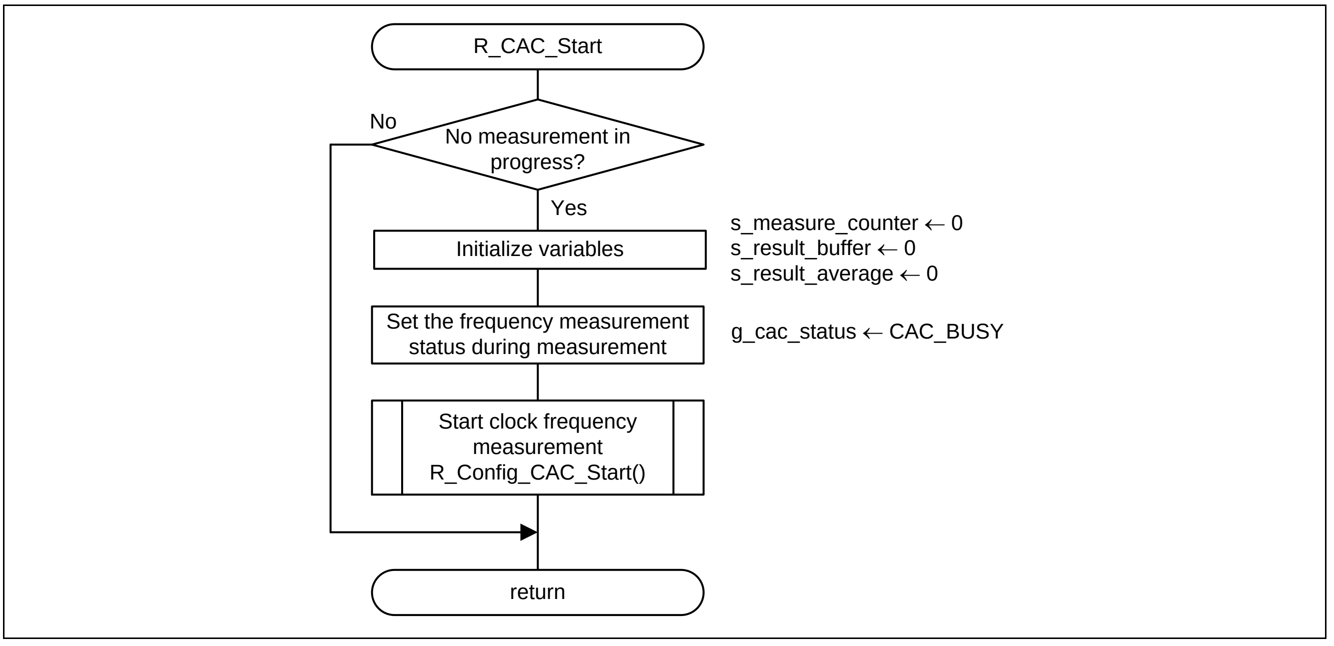


Figure 4.4 CAC Measurement Start Processing

4.10.3 CAC Measurement End Interrupt

Figure 4.5 is a flowchart of the CAC measurement end interrupt handler.

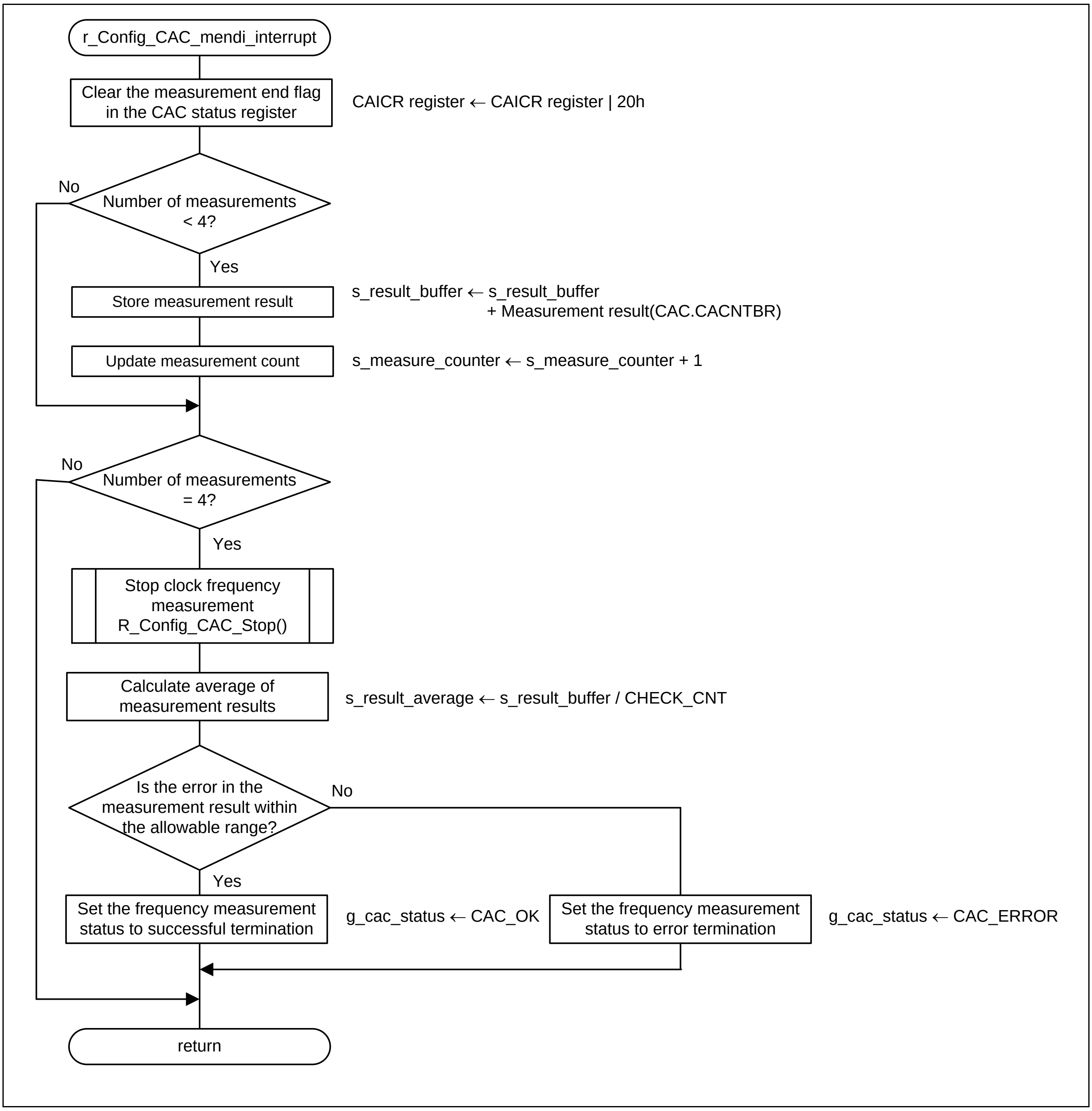


Figure 4.5 CAC Measurement End Interrupt

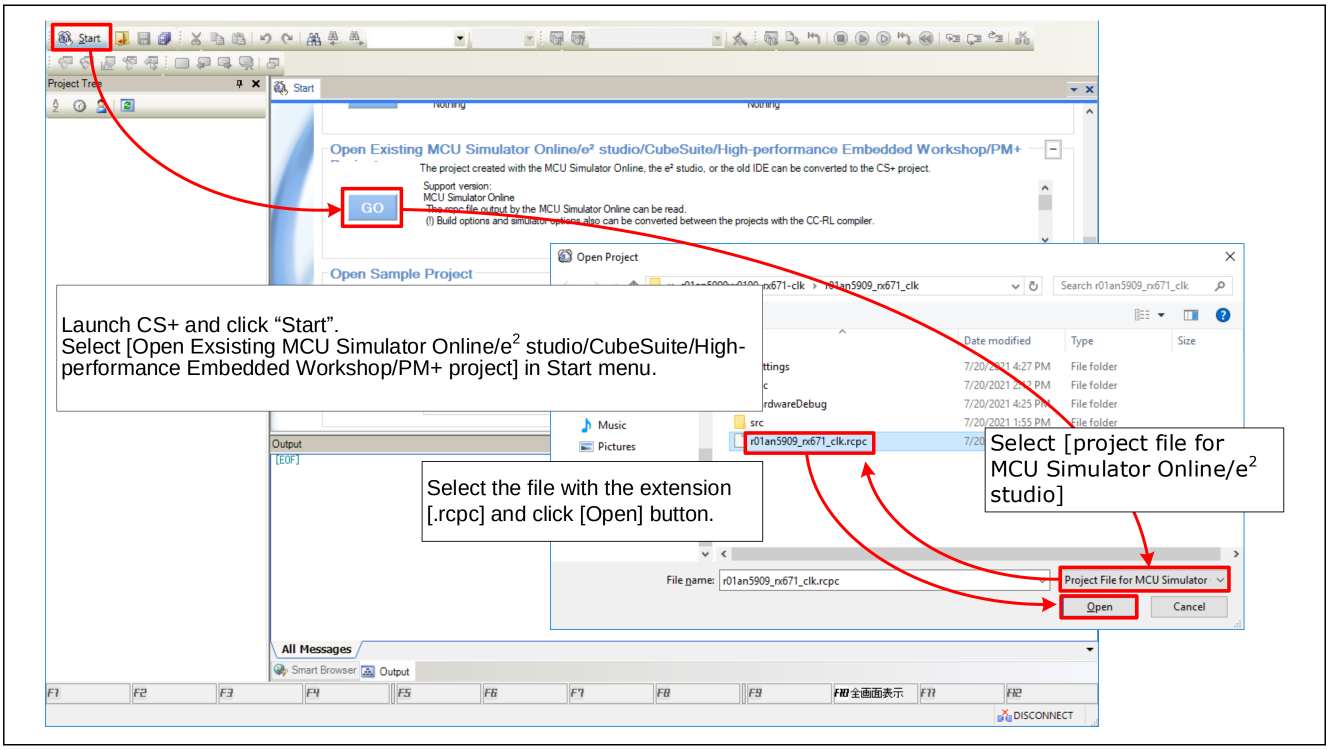
4.11 Procedure for Using Sample Code in CS+

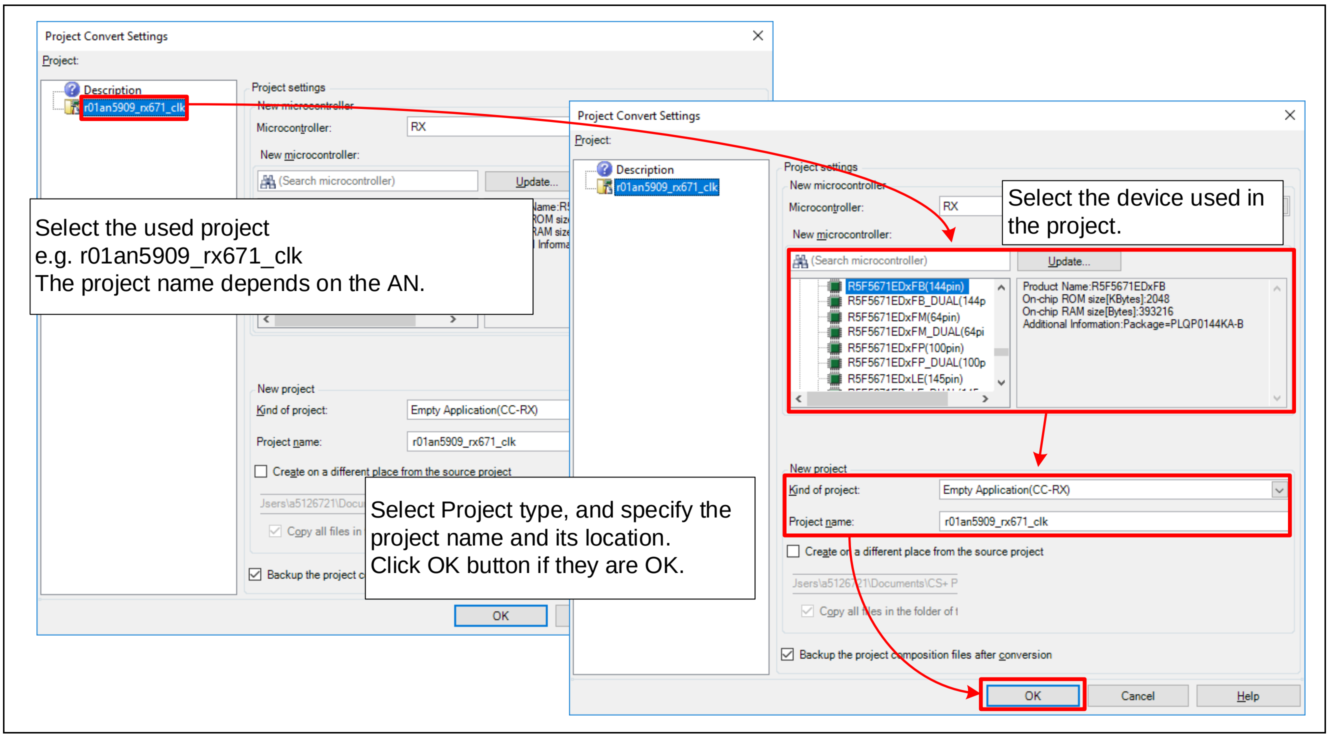
This application notes support the following development tools.

- e² studio Version: 2021-07 and RX Compiler CC-RX V3.03.00
- CS+ V8.06.00 and RX Compiler CC-RX V3.03.00

Using this application note with CS+

This application note contains a project only for e2 studio. When you use this project with CS+, import the project to CS+ by following procedures.





5. Reference Documents

User's Manual: Hardware
RX671 Group User's Manual: Hardware (R01UH0899EJ)
(The latest version can be downloaded from the Renesas Electronics website.)

User's Manual: Software
Smart Configurator User's Manual: RX API Reference (R20UT4360EJ)
(The latest version can be downloaded from the Renesas Electronics website.)

User's Manual: Development Tools
RX Family: CC-RX Compiler User's Manual (R20UT3248EJ)
(The latest version can be downloaded from the Renesas Electronics website.)

Technical Updates/Technical News
(The latest information can be downloaded from the Renesas Electronics website.)

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Aug. 25, 2021	—	First edition issued

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/.