Renesas Synergy™ Platform

CAC HAL Module Guide

Introduction

This module guide will enable you to effectively use a module in your own design. Upon completion of this guide, you will be able to add this module to your own design, configure it correctly for the target application and write code, using the included application project code as a reference and efficient starting point. References to more detailed API descriptions and suggestions of other application projects that illustrate more advanced uses of the module are available in the Renesas Synergy Knowledge Base (as described in the References section at the end of this document), and should be valuable resources for creating more complex designs.

The CAC HAL module provides a high-level API for clock accuracy control applications and uses the Clock Frequency Accuracy Measurement Circuit (CAC) peripheral on a Synergy MCU. This is particularly useful function when implementing a fail-safe mechanism for reliability-oriented applications. A user-defined callback can be created to respond to various error indications.

Contents

1. CAC HAL Module Features

The CAC HAL module API interfaces with a clock frequency-measurement circuit capable of monitoring the clock frequency based on a reference-signal input. The reference signal may be an externally supplied clock source or one of several available internal clock sources. An interrupt request may optionally be generated by a completed measurement, a detected frequency error, or a counter overflow. A digital filter is available for an externally supplied reference clock, and dividers are available for both internally supplied measurement and reference clocks. Edge-detection options for the reference clock are configurable as rising, falling, or both.

The frequency of the following clocks can be measured:

- Clock output from main-clock oscillator (main clock)
- Clock output from sub-clock oscillator (sub clock)
- Clock output from high-speed on-chip oscillator (HOCO clock)
- Clock output from mid-speed on-chip oscillator (MOCO clock)
- Clock output from low-speed on-chip oscillator (LOCO clock)
- Clock output from IWDT-dedicated on-chip oscillator (IWDTCLK clock)
- Peripheral module clock (PCLKB).

The measurement clock is monitored using a reference clock. The reference clock may be an external clock (supplied on the CACREF input pin) or one of the following internal clocks:

- Clock output from main-clock oscillator (main clock)
- Clock output from sub-clock oscillator (sub clock)
- Clock output from high-speed on-chip oscillator (HOCO clock)
- Clock output from mid-speed on-chip oscillator (MOCO clock)
- Clock output from low-speed on-chip oscillator (LOCO clock)
- Clock output from IWDT-dedicated on-chip oscillator (IWDTCLK clock)
- Peripheral module clock (PCLKB)

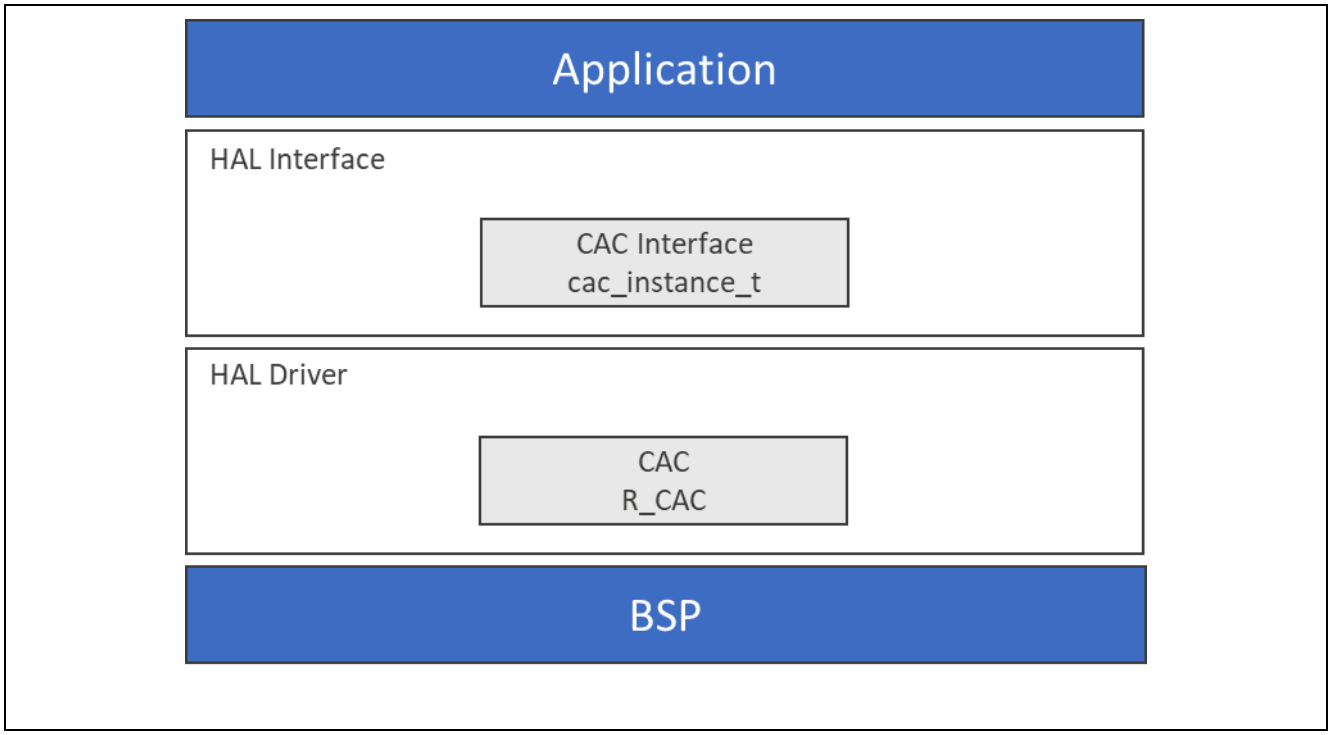

Figure 1. CAC HAL Module Block Diagram

2. CAC HAL Module APIs Overview

The CAC HAL module defines APIs for opening, closing, reading, starting, stopping and resetting the CAC. A complete list of the available APIs, an example API call and a short description of each can be found in the following table. A table of status return values follow the API summary table.

Table 1. CAC HAL Module API Summary

Function Name	Example API Call and Description
open	`g_cac0.p_api->open(g_cac0.p_ctrl, g_cac0.p_cfg)` Open function for CAC device.
read	`g_cac0.p_api->read(g_cac0.p_ctrl, &cac0_status, &cac0_counter)` Read function for CAC peripheral.
close	`g_cac0.p_api->close(g_cac0.p_ctrl)` Close function for CAC device.
stopMeasurement	`g_cac0.p_api->stopMeasurement(g_cac0.p_ctrl)` Ends a measurement for the CAC peripheral.
startMeasurement	`g_cac0.p_api->startMeasurement(g_cac0.p_ctrl)` Begin a measurement for the CAC peripheral.
reset	`g_cac0.p_api->reset(g_cac0.p_ctrl)` Reset function for CAC device.
versionGet	`g_cac0.p_api->versionGet(&cac0_version)` Get the CAC API and code version information.

Note: For details on operation and definitions for the function data structures, typedefs, defines, API data, API structures and function variables, review the *SSP User's Manual* API References for the associated module.

Table 2. Status Return Values

Name	Description
SSP_SUCCESS	API Call Successful
SSP_ERR_INVALID_ARGUMENT	One or more configuration options are invalid
SSP_ERR_NOT_OPEN	Open has not been successfully called
SSP_ERR_ASSERTION	Null provided for p_ctrl, p_cfg and others
SSP_ERR_INVALID_POINTER	Interrupt specified with NULL callback
SSP_ERR_HW_LOCKED	Hardware lock for CAC peripheral is already taken
SSP_ERR_INVALID_CAC_REF_CLOCK	Measured clock rate smaller than reference clock rate

Note: Lower-level drivers may return common error codes. Refer to the *SSP User's Manual* API References for the associated module for a definition of all relevant status return values.

3. CAC HAL Module Operational Overview

The CAC HAL module measures the operation and accuracy of a selected clock. Once the measurement is requested, counting begins on the first valid edge detected for the reference clock and ends on the next valid edge. A valid edge can be configured to be rising, falling, or both. The count is incremented at each cycle of the measurement clock after it has passed through the divider circuit that divides the clock by 1, 4, 8, or 32. An internally supplied reference clock also passes through a divider circuit that can divide the clock by 32, 128, 1024, or 8192. An externally supplied reference clock does not pass through a divider circuit — but it may pass through a digital filter, if it is configured to do so.

For example, if the sub-clock is specified as the measurement clock (32 kHz) and a divisor of 1 is specified, then the counter increments at a 32 kHz rate. If a reference clock of 1 kHz is provided, then after one cycle of the reference clock you would expect the counter to be 32. This is where the CAC upper and lower-limit settings are examined. Part of the setup for a CAC measurement is the specification of an upper and lower limit for the measurement. When a measurement is complete, the CAC compares the counter contents to the limits configured for the measurement. If the counter is both ≤ upper limit and ≥ lower limit, then the measurement has completed without error and the measured frequency is operating within the defined limits.

If the counter fails to meet these requirements, then a frequency error is indicated. A completed measurement may be identified by making API calls to poll the driver, or by establishing a callback function.

3.1 CAC HAL Module Operational Notes

Continuous Mode

The CAC module may be operated in either a single or continuous measurement mode. In continuous mode, the measuring process is restarted after each completed measurement. In non-continuous, or single measurement mode, measurement stops after the first completed measurement. The continuous_mode configuration member controls this feature.

Interrupts and Callbacks

When a measurement is completed, and a callback is provided by the user (with one or more interrupts enabled), the CAC HAL module invokes the callback (`p_callback`) with the argument `cac_callback_args_t`, indicating the event `cac_event_t`. If interrupts are not enabled, the API supports checking the measurement status to poll whether the measurement is complete (read), which provides both the status of the measurement and the current value of the CAC counter register.

The CAC driver supports three interrupts:

- A frequency error interrupt occurs when a measurement completes, and the CAC counter register value is outside of the range that was specified as part of the CAC driver open. The configuration @reg ferr_interrupt_enabled member provided in the `open` call must be enabled for this interrupt to be generated.
- An overflow error interrupt occurs when the CAC counter register overflows its maximum (0xFFFF) value. The configuration ovf_interrupt_enabled member provided in the `open` call must be enabled for this interrupt to be generated.
- A measurement complete interrupt occurs when a measurement completes, and the CAC counter register value is within the range that was specified as part of the CAC driver open. The configuration mei_interrupt_enabled member provided in the `open` call must be enabled for this interrupt to be generated.

Reset

The `reset` API can be used to reset the overflow, measurement end, and frequency error interrupt flags after the measurement has been stopped to eliminate any false triggers.

3.2 CAC HAL Module Limitations

Refer to the most recent *SSP Release Notes* for any additional operational limitations for this module.

4. Including the CAC HAL Module in an Application

This section describes how to include the CAC HAL module in an application using the SSP configurator.

Note: It is assumed that you are familiar with creating a project, adding threads, adding a stack to a thread, and configuring a block within the stack. If you are unfamiliar with any of these items, review the first few chapters of the *SSP User's Manual* to learn how to manage these important steps in creating SSP-based applications.

To add the CAC driver to an application, simply add it to a thread using the stacks selection sequence given in the following table. (The default name for the CAC is `g_cac0`. This name can be changed in the associated Properties window.)

Table 3. CAC HAL Module Selection Sequence

Resource	ISDE Tab	Stacks Selection Sequence
g_cac0 CAC Driver on r_cac	Threads > HAL/Common	New Stack > Driver> Monitoring > Clock Accuracy Circuit Driver on r_cac

When the CAC HAL module on r_cac is added to the thread stack as shown in the following figure, the configurator automatically adds any needed lower-level modules. Any drivers that need additional configuration information are box text highlighted in Red. Modules with a Gray band are individual modules that stand alone.

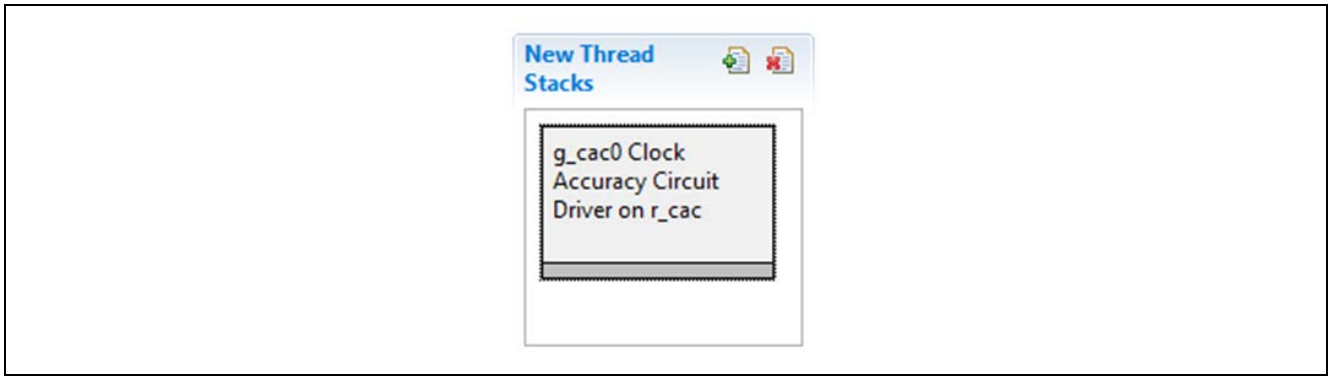

Figure 2. CAC HAL Module Stack

5. Configuring the CAC HAL Module

The CAC HAL module must be configured by the user for the desired operation. The SSP configuration window automatically identifies (by highlighting the block in red) any required configuration selections, such as interrupts or operating modes, which must be configured for lower-level modules for successful operation. Only those properties that can be changed without causing conflicts are available for modification. Other properties are 'locked' and unavailable for changes. These properties are identified with a lock icon for the 'locked' property in the Properties window in the ISDE. This approach simplifies the configuration process and makes it much less error-prone than previous 'manual' approaches to configuration. The available configuration settings and defaults for all the user-accessible properties are given in the Properties tab within the SSP configurator and are shown in the following tables for easy reference.

One of the properties most often identified as requiring a change is the interrupt priority. This configuration setting is available within the Properties window of the associated module. Simply select the indicated module to view the Properties window. The interrupt settings are often toward the bottom of the properties list, so scroll down until they become available. In the Properties window, note that the interrupt priorities indicate the validity of the setting based on the targeted MCU (CM4 or CM0+). This level of detail is not included in the configuration properties tables, but easily visible in the ISDE when configuring the interrupt priority levels.

Note: You may want to open your ISDE, create the module and explore the property settings in parallel with looking over the configuration table settings given below. This will help orient you and can be a useful 'hands-on' approach to learning the ins and outs of developing with SSP.

Table 4. Configuration Settings for the CAC HAL Module on r_cac

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	Controls whether to include code for API parameter checking.
Name	g_cac0	Identifies this instance
Continuous Measurement Operation	Enabled, Disabled (Default: Enabled)	If enabled, measurement continuously restarts after completing
Measurement Complete Interrupt	Enabled, Disabled (Default: Enabled)	Enabling allows the CAC driver to generate an interrupt when a measurement is complete, provided the CAC MEASUREMENT END interrupt is enabled in the ICU.

ISDE Property	Value	Description
Overflow Interrupt	Enabled, Disabled (Default: Enabled)	Enabling allows the CAC driver to generate an interrupt when a CAC overflow occurs, provided the CAC OVERFLOW interrupt is enabled in the ICU.
Frequency Error Interrupt	Enabled, Disabled (Default: Enabled)	Enabling allows the CAC driver to generate an interrupt when a frequency error occurs, provided the CAC FREQUENCY ERROR interrupt is enabled in the ICU.
Upper Limit Threshold	0 – 0xFFFF (Default: 0)	Top end of allowable range for measurement completion
Lower Limit Threshold	0 – 0xFFFF, must be < Upper Limit threshold> (Default: 0)	Bottom end of allowable range for measurement completion
Reference Clock Source	Main Oscillator, Sub-clock, HOCO, MOCO, LOCO, PCLKB, IWDT (Default: Main Oscillator)	Reference clock source
Reference Clock Divider	32,128,1024,8192 (Default: 32)	Reference clock divider
Reference Clock Edge Detect	Rising, Falling, Both (Default: Rising)	Reference clock edge detection
Reference Clock Digital Filter	Disabled, Sampling clock = Measuring freq, Sampling clock = Measuring freq/4, Sampling clock = Measuring freq/16 (Default: Disabled)	Reference clock digital filter
Measurement Clock Source	Main Oscillator, Sub-clock, HOCO, MOCO, LOCO, PCLKB, IWDT (Default: HOCO)	Measurement clock source
Measurement Clock Divider	1,4,8,32 (Default: 1)	Measurement clock divider
Callback	Null	Function name for callback
Frequency Error Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid), Disabled (Default: Disabled)	CAC frequency error interrupt priority
Measurement End Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid), Disabled (Default: Disabled)	CAC measurement end interrupt priority

ISDE Property	Value	Description
Overflow Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid), Disabled (Default: Disabled)	CAC overflow interrupt priority

Note: The example values and defaults are for a project using the Synergy S7G2 MCU Group. Other MCUs may have different default values and available configuration settings.

In some cases, settings other than the defaults for the module can be desirable. For example, it might be useful to select different clock sources and dividers.

5.1 CAC HAL Module Clock Configuration

Clocks are selected from the Clock tab as needed by the application.

5.2 CAC HAL Module Pin Configuration

The pins for the CAC HAL module are selected as shown in the following table. The pin settings are shown in the subsequent table. If CACREF is used as the input pin of the reference clock, you must configure this pin.

Table 5. Pin Selection Sequence for the CAC HAL Module

Resource	ISDE Tab	Pin selection Sequence
CAC HAL Module	Pins	Peripherals > Monitoring: CAC > CAC0

Note: The selection sequence assumes that CAC0 is the desired hardware target for the driver.

Table 6. Pin Configuration Settings for the CAC HAL Module

Pin Configuration Property	Value	Description
Operation Mode	Disabled, External Reference (Default: Disabled)	Select enable as the operation mode for CAC
CACREF	None, P204 (Default: None)	CACREF pin

Note: The example values are for a project using the Synergy S7G2 MCU Group and the SK-S7G2 Kit. Other Synergy Kits and other Synergy MCUs may have different available pin configuration settings.

6. Using the CAC HAL Module in an Application

You can use the following optional steps prior to the main application:

1. Start reference clock and measurement clock using the CGC @ref cgc_api_t::clockStart API if needed. For the started clock, use the CGC @ref cgc_api_t::clockCheck API to confirm oscillation stability or active state
2. Get the API and code version information using the CGC @ref cac_api_t::versionGet if needed.

Typical steps in using the CAC HAL module in an application include:

1. Initialize the CAC HAL module using the @ref cac_api_t::open API
2. Start a measurement using the @ref cac_api_t::startMeasurement API
3. Poll using the @ref cac_api_t::read function to look for the measurement result or get measurement status and result using callback function called in ISR
4. Stop the measurement using the @ref cac_api_t::stopMeasurement API
5. Reset the overflow, measurement end, and frequency error interrupt flags using the @ref cac_api_t::reset API
6. Close the CAC HAL module if no more measurements are needed using the @ref cac_api_t::close API.

The following diagram shows common steps in a typical operation flow using the CAC HAL module.

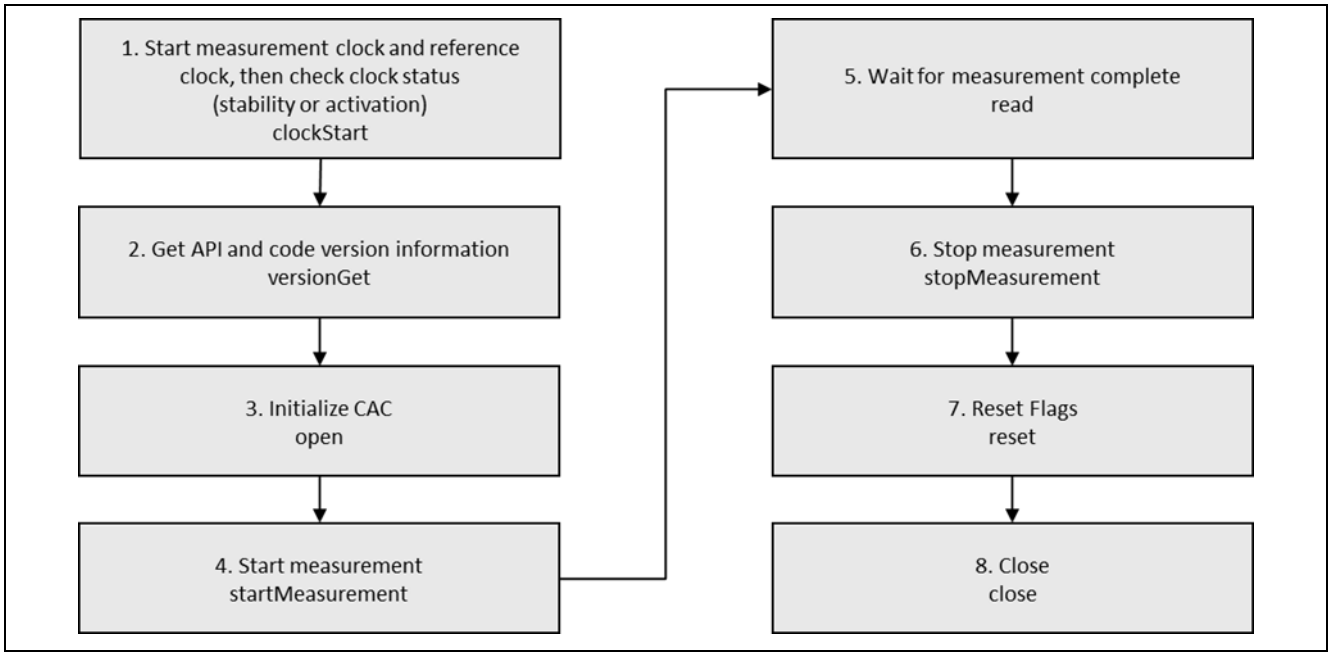

Figure 3. Flow Diagram of a Typical CAC HAL Module Application

7. CAC HAL Module Application Project

The application project associated with this module guide demonstrates the steps shown in a full design. The project can be found using the link provided in the References section at the end of this document. You may want to import and open the application project within the ISDE and view the configuration settings for the CAC HAL module. You can also read over the code (in `cac_hal_mg.c`), which is used to depict the CAC APIs in a complete design.

The application project demonstrates the typical use of the CAC APIs. The application project `cac_hal_entry` initializes the CAC HAL module, starts the measurement and waits for the measurement to complete polling in the main loop. The measurement results and status are read using the `read` API and printed on the Debug Console using the common semi-hosting function. After the measurement is complete, stop the measurement, and close the CAC HAL module. The following table identifies the target versions for the associated software and hardware used by the application project.

Table 7. Software and Hardware Resources Used by the Application Project

Resource	Revision	Description
e^2 studio	v5.3.1 or later	Integrated Solution Development Environment
SSP	v1.2.0 or later	Synergy Software Platform
IAR EW for Synergy	v7.71.2 or later	IAR Embedded Workbench® for Renesas Synergy™
SSC	v5.3.1 or later	Synergy Standalone Configurator
SK-S7G2	v3.0 to v3.1	Starter Kit

The following diagram shows a simple flow of the application project:

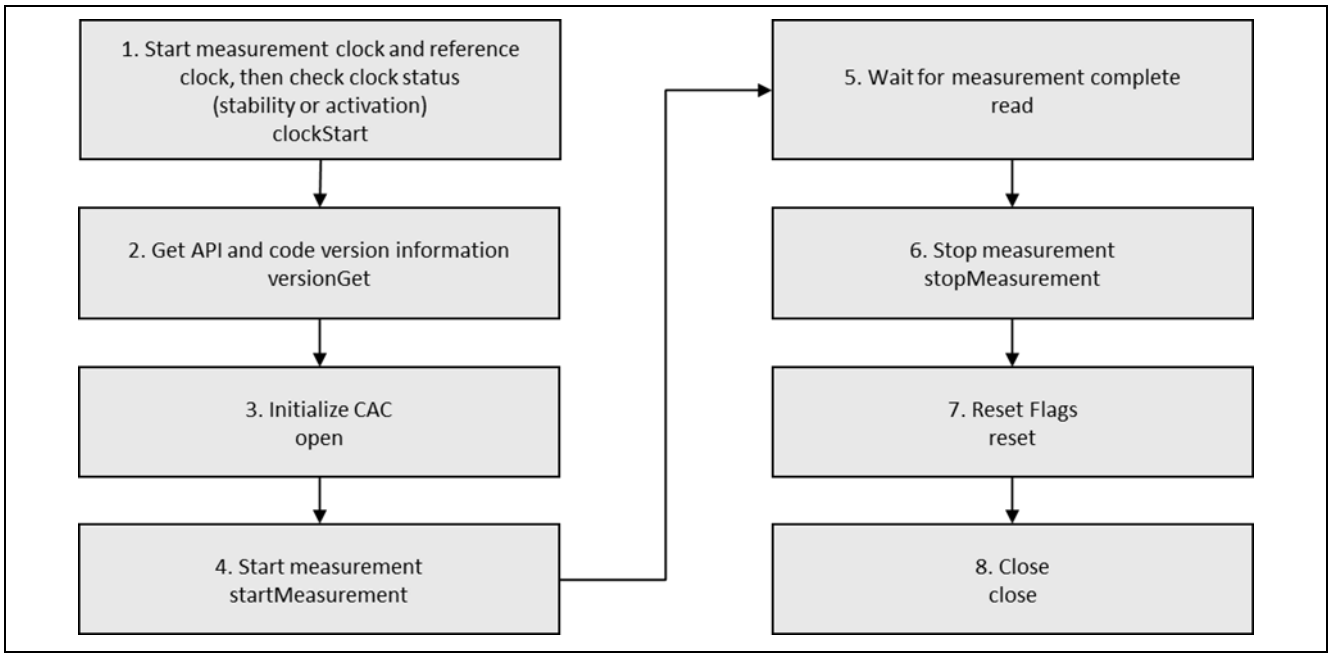

Figure 4. CAC HAL Module Application Project Flow Diagram

The complete application project can be found using the link provided in the References section at the end of this document. The `cac_hal_mg.c` file resides in the project once it has been imported into the ISDE. You can open this file within the ISDE and follow along with the description provided to help identify key uses of APIs.

The first section of `cac_hal_mg.c` has the header files which reference the CAC instance-structure and a code section which allows semi-hosting to display results using printf(). The next section is the entry function for the main program-control section. Before initializing the CAC HAL module, the measurement clock and the reference clock must be started first. The stability of the HOCO, Main OSC, and PLL, as well as the activate state of LOCO, MOCO, and Sub clock should then be checked before starting the CAC to measure frequency accuracy. If all clocks have been activated before, such as having been already started by the cgc module, the start step can be skipped.

If the API and code version information are useful for the application, use the `versionGet` API to obtain these values. To make the CAC module effective, initialize the CAC module using the `open` API. Then, start the measurement using the `startMeasurement` API and wait for measurement to complete in a `do while` loop. Whether the measurement completes or generates an error, the values of the status register and the counter buffer register are stored into the user-defined variables. If semi-hosting is defined, these values are printed on the Debug Console. After one measurement completes, use the `stopMeasurement` API to stop CAC measurement, then use the `reset` API to reset the overflow, measurement end, and frequency error interrupt flags. Finally, use the `close` API to close the CAC HAL module.

Note: This description assumes that you are familiar with using printf() the Debug Console with the Synergy Software Package. If you are unfamiliar with this, refer to the How do I Use Printf() with the Debug Console in the *Synergy Software Package* listed in the References section at the end of this document. Alternatively, you can see results using the watch variables in the debug mode.

To support the required operations and the physical properties of the target board and MCU a few key properties are configured in this application project. The properties with values set for this specific project are listed in the following table. You can open the application project and view these settings in the Properties window as a hands-on exercise.

Table 8. CAC Configuration Settings for the Application Project

ISDE Property	Value Set
Name	g_cac0
Continuous Measurement Operation	Enabled
Measurement Complete Interrupt	Disabled

ISDE Property	Value Set
Overflow Interrupt	Disabled
Frequency Error Interrupt	Disabled
Upper Limit Threshold*	6,990
Lower Limit Threshold*	6,663
Reference clock source	Main Oscillator
Reference clock divider	8,192
Reference clock edge detect	Rising
Reference clock digital filter	Disabled
Measurement clock source	HOCO
Measurement clock divider	1
Callback	NULL
Frequency Error Interrupt Priority	Disabled
Measurement End Interrupt Priority	Disabled
Overflow Interrupt Priority	Disabled

Note: Without FLL

8. Customizing the CAC HAL Module for a Target Application

Some configuration settings are normally changed by the developer from those shown in the application project. For example, the user can easily change the configuration settings for the CAC reference and measurement clocks by updating the corresponding clock in the Clocks tab. The CAC upper-limit threshold and lower-limit threshold can also be changed easily by inputting new values in Properties window.

9. Running the CAC HAL Module Application Project

To run the CAC HAL module application project and to see it executed on a target kit, you can simply import it into your ISDE, compile, and run debug.

Note: The following steps are described in sufficient detail for someone experienced with the basic flow through the Renesas Synergy™ development process. If these steps are unfamiliar, review the first few sections in the *SSP User's Manual*, as described in the Reference section at the end of this document.

To create and run the CAC HAL module application project, simply follow these steps:

1. Import and build the example project provided in the package according to the *Renesas Synergy™ Project Import Guide* (r11an0023eu01121-synergy-ssp-import-guide.pdf).
2. Connect to the host PC using the USB cable (use J19 DEBUG_USB connector).
3. Start to debug the application.
4. The output can be viewed in the Renesas Debug Console

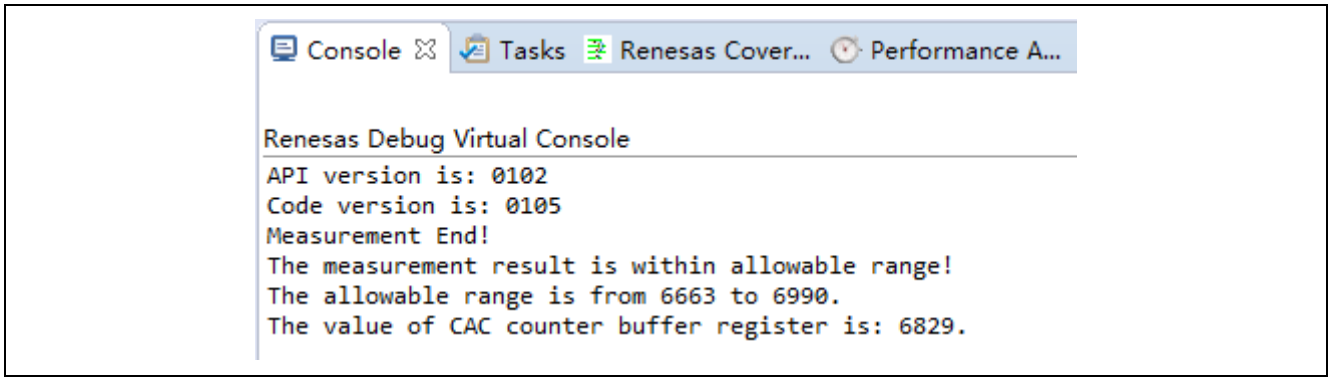

Figure 5. Example Output from the CAC HAL Module Application Project

Note: In the application project, the measurement clock is HOCO divided by 1 and the reference clock is the main oscillator divided by 8,192. From the Clocks tab, XTAL is 24 MHz and HOCO is 20 MHz. Using

HOCO to measure the main oscillator, the theoretical value should be (1/(24 MHz/8,192))/(1/20 MHz) or about 6,826.67. Considering the allowed deviation shown in the electrical characteristics in the datasheet, a pair of values for both upper-limit threshold and lower-limit threshold is set.

10. CAC HAL Module Conclusion

This module guide has provided all the background information needed to select, add, configure, and use the module in an example project. Many of these steps were time consuming and error-prone activities in previous generations of embedded systems. The Renesas Synergy Platform makes these steps much less time consuming, and removes the common errors, like conflicting configuration settings or the incorrect selection of lower-level drivers. The use of high-level APIs (as demonstrated in the application project) illustrates additional development time savings by allowing work to begin at a high level, and avoiding the time required in older development environments to use or, in some cases, create, lower-level drivers.

11. CAC HAL Module Next Steps

After you have mastered a simple CAC HAL module project, you may want to exercise a more complex example. You can adjust the LOCO, MOCO, or HOCO user-trimming control register to improve on-chip clock accuracy. You can refer to the CGC HAL module guide to be familiar with clock operation and practice on your own.

12. CAC HAL Module Reference Information

SSP User Manual: Available in html format in the SSP distribution package and as a pdf from the Synergy Gallery.

Links to all the most up-to-date `r_cac` module reference materials and resources are available on the Synergy Knowledge Base: https://en-support.renesas.com/search/r_cac%20Module%20Guide%20Resources.

Website and Support

Visit the following vanity URLs to learn about key elements of the Synergy Platform, download components and related documentation, and get support.

Synergy Software	www.renesas.com/synergy/software
Synergy Software Package	www.renesas.com/synergy/ssp
Software add-ons	www.renesas.com/synergy/addons
Software glossary	www.renesas.com/synergy/softwareglossary
Development tools	www.renesas.com/synergy/tools
Synergy Hardware	www.renesas.com/synergy/hardware
Microcontrollers	www.renesas.com/synergy/mcus
MCU glossary	www.renesas.com/synergy/mcuglossary
Parametric search	www.renesas.com/synergy/parametric
Kits	www.renesas.com/synergy/kits
Synergy Solutions Gallery	www.renesas.com/synergy/solutionsgallery
Partner projects	www.renesas.com/synergy/partnerprojects
Application projects	www.renesas.com/synergy/applicationprojects
Self-service support resources:	
Documentation	www.renesas.com/synergy/docs
Knowledgebase	www.renesas.com/synergy/knowledgebase
Forums	www.renesas.com/synergy/forum
Training	www.renesas.com/synergy/training
Videos	www.renesas.com/synergy/videos
Chat and web ticket	www.renesas.com/synergy/resourcelibrary

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Sep.07.17	—	Initial release
1.01	Jan.07.19	10	Simplified section 9 by removing project creation instructions

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 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.
5. 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.
6. 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.
7. 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.
8. 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.
9. 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.
10. 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.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. 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.4.0-1 November 2017)

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/.