Renesas Synergy™ Platform

CGC 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 CGC HAL module is a high-level API for clock-control applications and is implemented on r_cgc. The CGC HAL module configures and controls the clock-control functions of a Synergy MCU using the clock-control peripheral. Because every project requires a clock function, the CGC HAL module is added to a project by default. (The module is configured in the ISDE.) A user-defined callback can be created to signal when the main oscillator has stopped.

Contents

1. CGC HAL Module Features

The CGC HAL module supports the configuration and control of the various clocking functions on the Renesas Synergy MCU. Key features include:

- Select the system clock source
  - HOCO (high-speed on-chip oscillator), MOCO (middle-speed on-chip oscillator), LOCO (low-speed on-chip oscillator), Main Clock, PLL, or Sub-Oscillator
- Configure the internal clocks and turn them on or off
- Configure the output clocks
- Set up the Oscillation Stop Detection feature
- Set up clock divisors on each of the up to six clock domains
- Some Synergy MCUs also support controllable external clock outputs which may have independent divisors

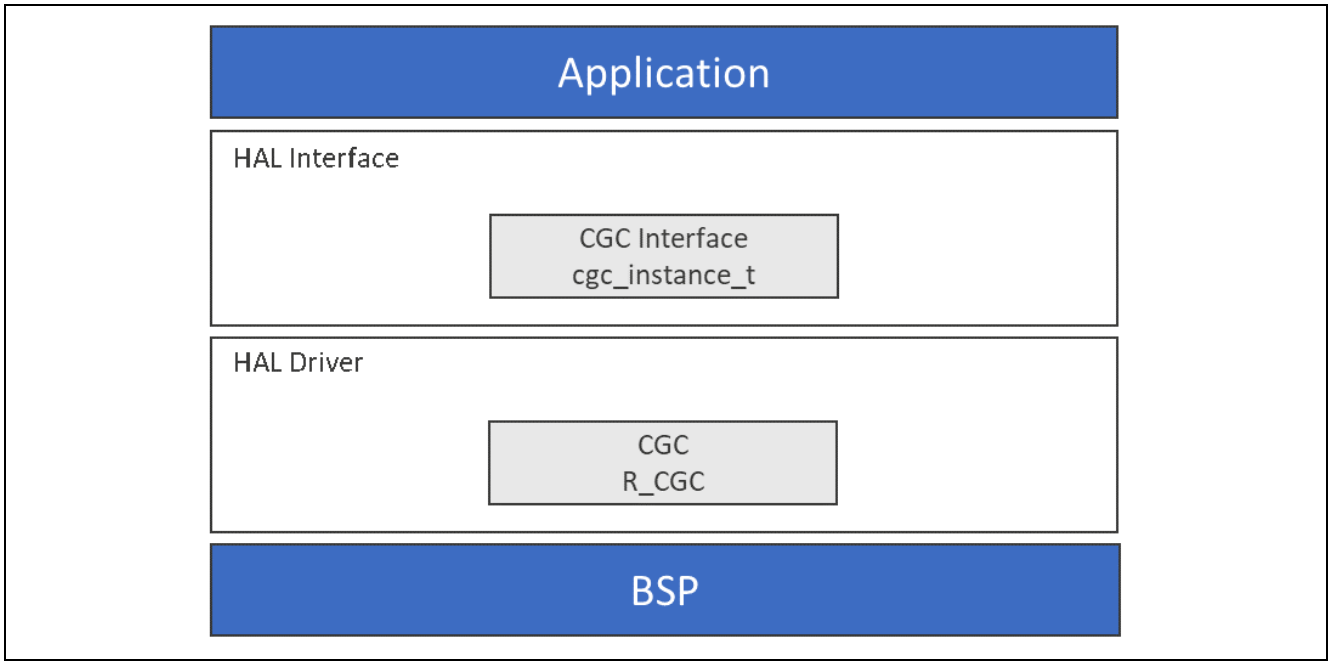

Figure 1. CGC HAL Module Block Diagram

2. CGC HAL Module APIs Overview

The CGC HAL module defines APIs for initializing, starting, controlling and stopping the MCU clock. A complete list of the available APIs, an example API call and a short description of each can be found in the following API Summary table. A table of status return values are listed after the API summary.

Table 1. CGC HAL Module API Summary

Function Name	Example API Call and Description
.init	`g_cgc.p_api->init();` Initial clock configuration called by BSP automatically.
.clocksCfg	`g_cgc.p_api->clocksCfg(&p_clock_cfg);` The BSP calls this function at startup, but it can also be called from the application to change clocks at runtime.
.clockStart	`g_cgc.p_api->clockStart(clock_source, &p_clock_cfg);` Start a clock.
.clockStop	`g_cgc.p_api->clockStop(clock_source);` Stop a clock.
.systemClockSet	`g_cgc.p_api->systemClockSet(clock_source, &p_clock_cfg);` Set the system clock.

Function Name	Example API Call and Description
.systemClockGet	`g_cgc.p_api->systemClockGet(&clock_source, &clock_config);` Get the system clock information.
.systemClockFreqGet	`g_cgc.p_api->systemClockFreqGet(&clock_source, &frequency_hz);` Return the frequency of the selected clock.
.clockCheck	`g_cgc.p_api->clockCheck(clock_source);` Check the stability of the selected clock.
.oscStopDetect	`g_cgc.p_api->oscStopDetect(callback, enable);` Configure the Main Oscillator stop detection.
.oscStopStatusClear	`g_cgc.p_api->oscStopStatusClear();` Clear the oscillator stop detection flag.
.busClockOutCfg	`g_cgc.p_api->busClockOutCfg (divider);` Configure the bus clock output secondary divider. The primary divider is set using the BSP clock configuration and the systemClockSet function (S7G2 and S3A7 MCU only).
.busClockOutEnable	`g_cgc.p_api->busClockOutEnable ();` Enable the bus clock output (S7G2 and S3A7 MCU only).
.busClockOutDisable	`g_cgc.p_api->busClockOutDisable ();` Disable the bus clock output (S7G2 and S3A7 MCU only).
.clockOutCfg	`g_cgc.p_api->clockOutCfg(clock_source, clock_dividers);` Configure clockOut.
.clockOutEnable	`g_cgc.p_api->clockOutEnable();` Enable clock output on the CLKOUT pin. The source of the clock is controlled by clockOutCfg.
.clockOutDisable	`g_cgc.p_api->clockOutDisable();` Disable clock output on the CLKOUT pin. The source of the clock is controlled by clockOutCfg.
.lcdClockCfg	`g_cgc.p_api->lcdClockCfg(clock);` Configure the segment LCD Clock (S3A7 and S124 MCUs only).
.lcdClockEnable	`g_cgc.p_api->lcdClockEnable();` Enable the LCD clock (S3A7 and S124 MCUs only).
.lcdClockDisable	`g_cgc.p_api->lcdClockDisable();` Disables the LCD clock (S3A7 and S124 MCUs only).
.sdramClockOutEnable	`g_cgc.p_api->sdramClockOutEnable();` Enables the SDRAM clock output (S7G2 MCU only).
.sdramClockOutDisable	`g_cgc.p_api->sdramClockOutDisable();` Disables the SDRAM clock (S7G2 only).
.usbClockCfg	`g_cgc.p_api->usbClockCfg(divider);` Configures the USB clock (S7G2 only).
.systickUpdate	`g_cgc.p_api->ssystickUpdate(period_count, units);` Update the Systick timer.
.versionGet	`g_cgc.p_api->versionGet(&version);` Retrieve the API version with the version pointer.

Note: For more complete descriptions of 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_ABORTED	Attempt to update systick timer failed.

Name	Description
SSP_ERR_HARDWARE_TIMEOUT	Hardware timed out.
SSP_ERR_STABILIZED	Clock stabilized.
SSP_ERR_CLOCK_INACTIVE	Clock not turned on.
SSP_ERR_MAIN_OCO_INACTIVE	Main OCO off/unstable.
SSP_ERR_CLOCK_ACTIVE	Clock active.
SSP_ERR_NOT_STABILIZED	Clock source un-stabilized.
SSP_ERR_CLKOUT_EXCEEDED	Clock out exceeded.
SSP_ERR_NULL_PTR	Pointer null.
SSP_ERR_OSC_DET_ENABLED	Oscillation stop detection enabled.
SSP_ERR_OSC_STOP_DETECTED	The Oscillation stop detect status flag is set. Under this condition it is not possible to disable the Oscillation stop detection function.
SSP_ERR_OSC_STOP_CLOCK_ACTIVE	The Oscillation Detect Status flag cannot be cleared if the Main Osc or PLL is set as the system clock. Change the system clock before attempting to clear this bit.
SSP_ERR_INVALID_ARGUMENT	Invalid argument.
SSP_ERR_INVALID_MODE	Attempt to start a clock in a restricted operating power control mode.

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. CGC HAL Module Operational Overview

The CGC HAL module interface provides the ability to configure and use all the CGC HAL module's capabilities. Among those capabilities are the selection of several clock sources to use as the system clock source; additionally, the system clocks can be divided down to provide a wide range of frequencies for various system and peripheral needs.

Clock stability can be checked, and clocks may also be stopped to save power when they are not needed. The API has a function to return the frequency of the system and system peripheral clocks at run time. There is also a feature to detect when the main oscillator has stopped, with the option of calling a user-provided callback function.

The CGC HAL module to configure and control clock features:

- Configure any of the available clocks (HOCO, MOCO, LOCO, Main Clock, PLL, Sub-Oscillator) as the system clock source
- Configure the internal clocks (ICLK, PCLK and so on)
- Switch the clocks on and off
- Configure the output clocks
- Set up the Oscillation Stop Detection feature

The Clock Generation Circuit peripheral features the following oscillators and clock generators:

- Main oscillator input of up to 24 MHz
- A 32.768 kHz sub-clock oscillator
- HOCO running at up to 64 MHz (depending on the device version)
- MOCO running at 8 MHz
- LOCO running at 32.768 kHz
- PLL circuit output running between 24 MHz and 240 MHz, depending on the device

Renesas Synergy microcontrollers have six internal clock domains. Each of them has independent divisors, but they are dependent upon the clock input selected in the System Clock Control Register. These are:

- ICLK – The core clock, for CPU, DMAC, ROM and RAM (max 32/48/240 MHz)
- PCLKA – Peripheral clock for modules including EtherC, EDMAC, USB2.0 HS, QSPI and SCIF (max 32/48/120MHz)
- PCLKB – Peripheral clock for modules like IIC, CAN, DAC12, RTC, USBFS, I/O Ports, WDT and IWDT (max 32/60 MHz)
- PCLKC – Peripheral clock for ADC12 conversion clock (max 64/60 MHz)
- PCLKD – Peripheral clock for GPT count clock (max 64/120 MHz)
- FCLK –Clock source for the flash memory (max 32/60 MHz)

In addition, some of the Synergy microcontrollers also support controllable external clock outputs, some of which have independent divisors. These are:

- CLKOUT – CLOCKOUT/BUZZER clock (max 24 MHz) (independent clock selector and divisor)
- BCLK – External bus clock to external bus controller (max 16/120 MHz)
- SDCLK – SDRAM clock (max 120 MHz)
- UCLK – USB clock (max 120 MHz) (independent divisor/selector on Synergy version 2)
- LCD_CLK – LCD Clock (independent clock selector but no divisor)

3.1 Changing the System Clock Peripheral Clock Divisors at Runtime

The CGC HAL module also has the option to change the system clock and clock tree settings at runtime via the `clocksCfg` API function. Choose New Stack>Driver>System>CGC Configuration Instance to create a configuration structure for use with the `clocksCfg` API function.

The `clocksCfg` function allows changes to the system clock, the peripheral clock dividers, the PLL multiplier and divider and the state-of-the-system clocks (stop/start; (HOCO, Main Oscillator, Subclock oscillator, and so on). The options in the following figure show an example where the system clock is being changed from HOCO to MOCO, and the peripheral clock dividers are also being updated. The options for each clock are start, stop, and none (meaning no change). Not all clocks are available on all MCUs. Not all peripheral clocks are available on all MCUs.

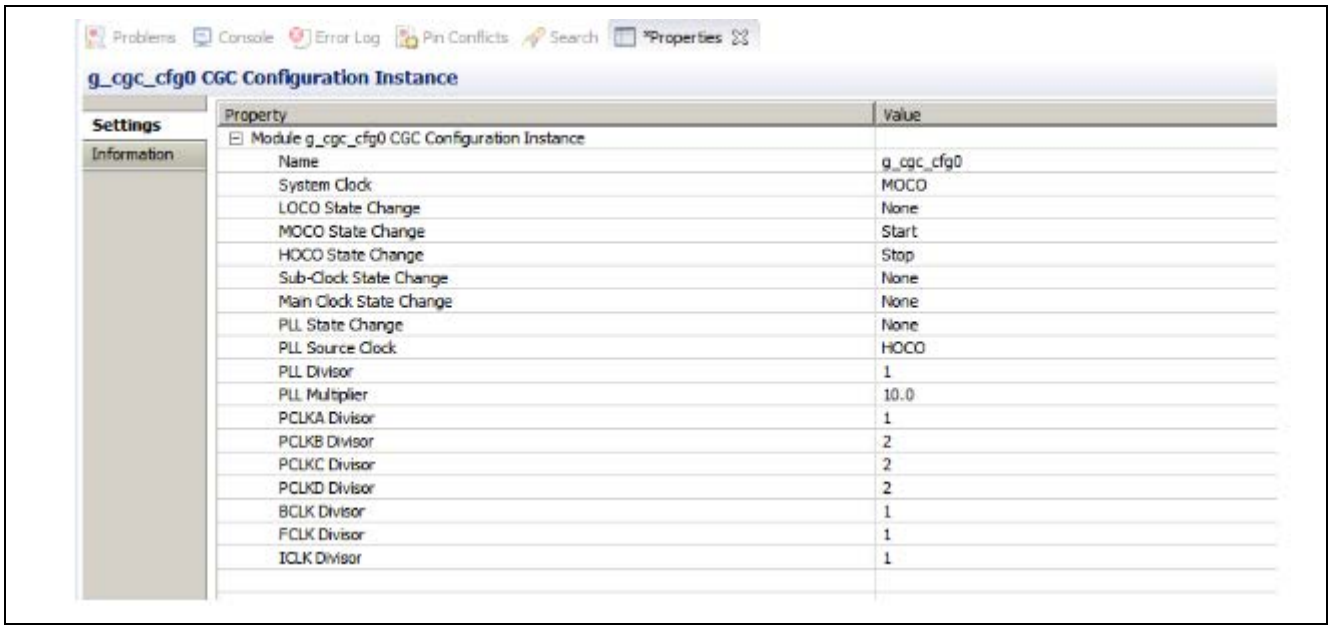

Figure 2. CGC Configuration Properties

The function call for the preceding example is:

```
g_cgc.p_api > clocksCfg(&g_cgc_cfg0);
```

3.2 Option Setting Memory

All Renesas Synergy microcontrollers include an Option-Setting Memory, this memory can be used to set the operating state of peripherals after a reset. The OFS can be used to set the state of the IWDT, WDT, LVD and CGC HOCO. The following table lists CGC HOCO parameters that can be configured by OFS registers:

Table 3. The OFS register set possibilities

Control	Description
HOCO oscillation enable	Automatically starts the HOCO after a Reset, if enabled
HOCO Frequency	S7 & S5 Series 16 MHz 18 MHz 20 MHz S3 & S1 Series 24 MHz 32 MHz 48 MHz 64 MHz

You can set the OFS register values through the properties dialog; the properties dialog is available on the Synergy Configuration editor when you select the BSP tab.

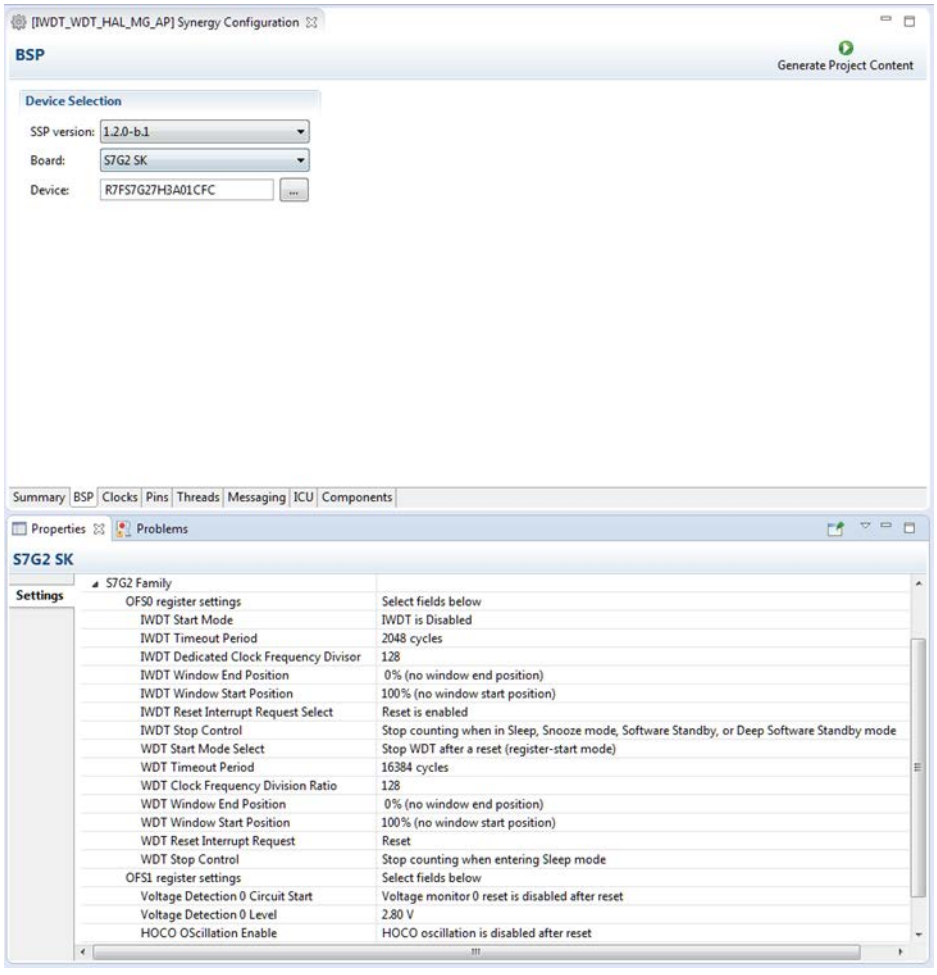

Figure 3. OFS Register Settings

The CGC HAL module also handles operating power control modes of the MCU since the Low Power Modes Version 2 HAL module will no longer handle operation-power control modes of the MCU.

3.3 CGC HAL Module Important Operational Notes and Limitations

3.3.1 CGC HAL Module Operational Notes

- The CGC HAL module has no dependencies with respect to the ThreadX® RTOS.
- The CGC HAL module is a core function of the MCU and is set after by the BSP initialization process it is quite possible that the CGC can be left unchanged. However, the CGC HAL module provides functions that change the clock configuration that can balance the requirements of operating speed and power consumption, depending on application requirements.
- The CGC peripheral of the Synergy microcontrollers support Oscillator Stop Detection. If enabled in the application, the Oscillator Stop Detection function automatically detects whether the Main/PLL clock has stopped and then switches operation to the MOCO. When enabling this functionality in the SSP, a callback function must be manually created by the user. The following steps detail this procedure.

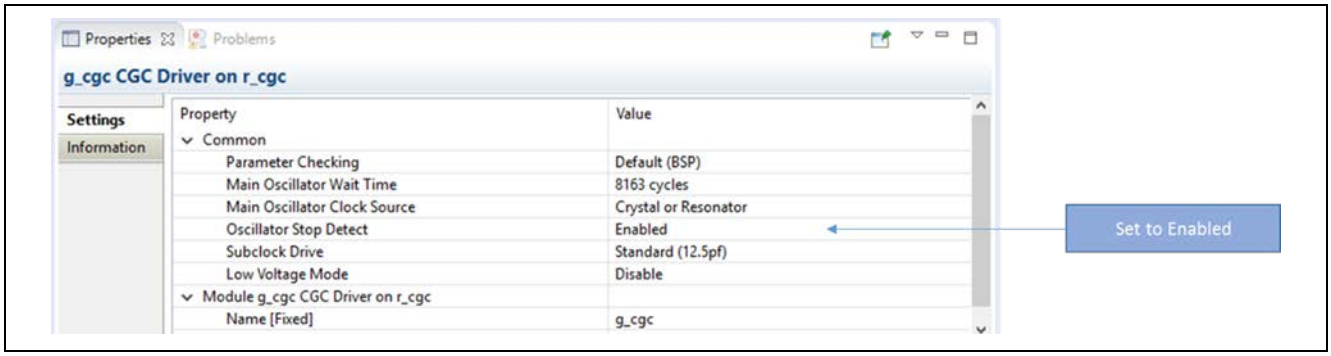

Figure 4. Oscillator Stop Detect Enable/Disable

In the application code, create a callback function. In this example, it is called `osc_stop_callback.`

```
void osc_stop_callback(cgc_callback_args_t * p_args)
{
      /* perform Oscillator Stop Detection processing */
}
```

Enable the oscillator stop detection by calling the API with the previously declared callback.

```
/* Enable the Osc Stop Detect functionality */
g_cgc.p_api->oscStopDetect( osc_stop_callback, true );
```

Enable the interrupt within the ICU

```
/* Osc Stop Detect is an NMI interrupt.  Enable the NMI in ICU */
R_ICU->NMIER_b.OSTEN = 1;
```

3.3.2 CGC HAL Module Limitations

Refer to the latest SSP release notes for limitations on the use of this module.

4. Including the CGC HAL Module in an Application

This section describes how to include the CGC 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 these items, refer to the first few chapters of the SSP User's Manual to learn how to manage each of these important steps in creating SSP-based applications.

The CGC Driver is automatically added to the HAL/Common thread, so it only needs to be added to a new thread if it has been removed. (The default name for the CGC is g_cgc. This name can be changed in the associated Properties window.)

Table 4. CGC HAL Module Selection Sequence

Resource	ISDE Tab	Stacks Selection Sequence
g_cgc CGC HAL on r_cgc	Threads	New Stack> Driver> System> CGC Driver on r_cgc

The CGC Driver on g_cgc is automatically added to the HAL/Common Stack, as shown in the following figure:

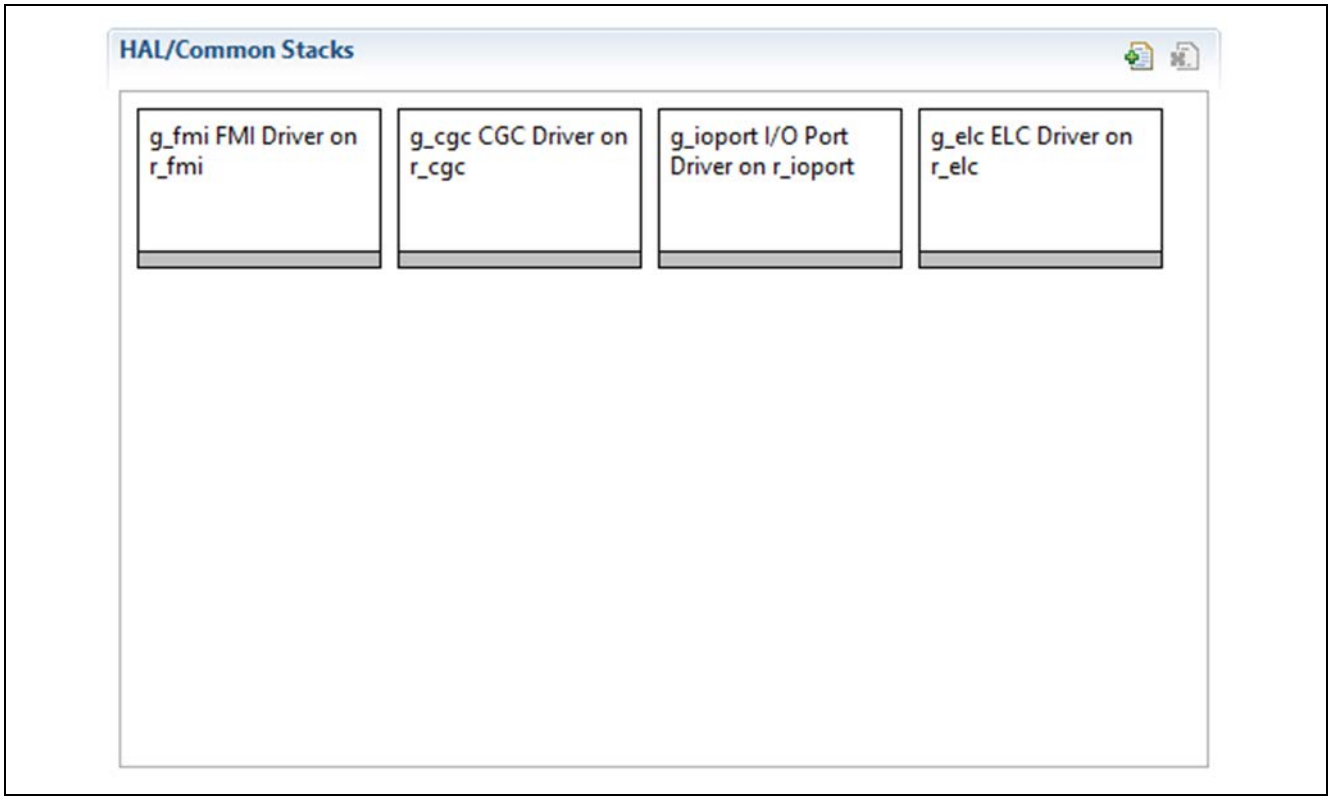

Figure 5. CGC HAL Module Stack

5. Configuring the CGC HAL Module

The CGC HAL module must be configured by the user for the desired operation. 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.

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

Table 5. Configuration Settings for the CGC HAL Module on r_cgc

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	Enable or disable the parameter error checking.
Main Oscillator Wait Time	3,35,67,131,259,547,1059,2147,4291,8163 cycles (Default: 8163 cycles)	Set to one of these values. It should be, at minimum, the main clock stabilization time length. This delay will be configured only if #define CGC_CFG_MAIN_OSC_CLOCK_SOURCE is set to 0, indicating that a resonator/ crystal is used. Set the main clock oscillation stabilization time to longer than or equal to the stabilization time recommended by the oscillator manufacturer.

ISDE Property	Value	Description
Main Oscillator Clock Source	External Oscillator, Crystal or Resonator (Default: Crystal or Resonator)	Set to 0 if a resonator, or crystal, is used. Set to 1 if an external oscillator input is used
Oscillator Stop Detect	Enabled, Disabled (Default: Enabled)	This allows the R_CGC_OscStopDetect function code to be generated if enabled. The user must call this function with a callback pointer to use this feature.
Subclock Drive	Standard (12.5pf), Middle (4.4pf) (Default: Standard)	This setting is for matching the subclock oscillator drive capacitance based on the crystal parameters #define CGC_CFG_SUBCLOCK_ DRIVE.

Note: The example values and defaults are for a project using the Synergy S7G2 MCU Family. Other MCUs may have different default values and available configuration settings.
In some cases, settings other than the defaults for the CGC HAL module can be desirable. For example, it might be useful to selectively turn clocks on or off or change frequency to optimize power and performance characteristics.

Note: Most of the property settings for modules are reasonably intuitive and usually they can be determined by inspection of the associated properties window from the SSP configurator.

5.1 CGC HAL Module Clock Configuration

The default CGC HAL module clock frequencies that are set by the BSP initialization process are configurable in the ISDE by using the Clocks tab in the configurator. Invalid selections are indicated in red when selected.

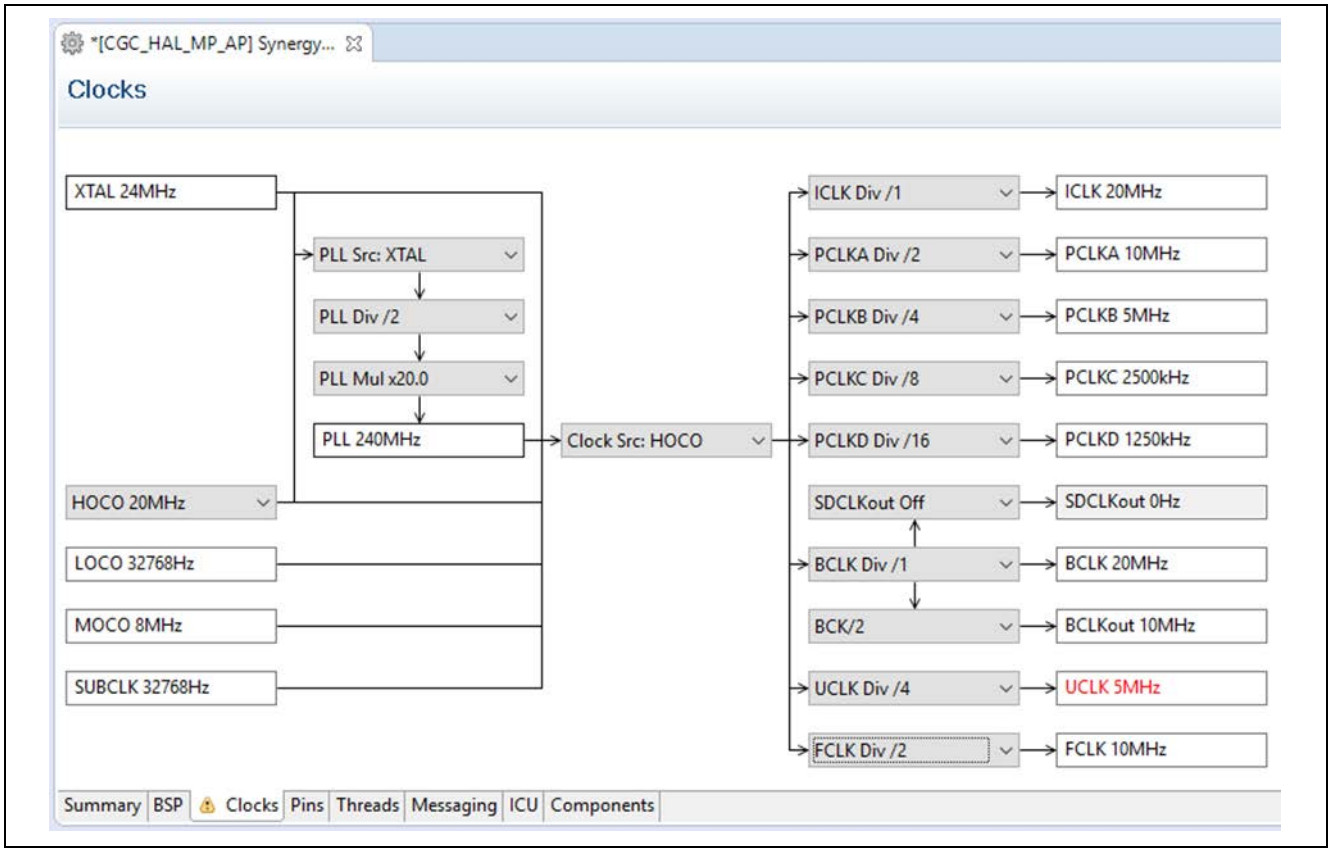

Figure 6. Default Clock Settings via the Clocks tab

In this example, the Clock Source is HOCO, and various clock dividers are chosen for the peripheral clocks. If a valid USB Clock (UCLK) cannot be achieved, it is highlighted in RED. It should be noted that this is only advisory, and the project will still build; as such, a clock frequency may be required.

5.2 CGC HAL Module Pin Configuration

The CGC peripheral module controls the output of BCLK and SDCLK signals. Use the Clocks tab to enable / disable this functionality. The BCLK_SDCLK I/O pin must be selected and configured as required via the Pins tab.

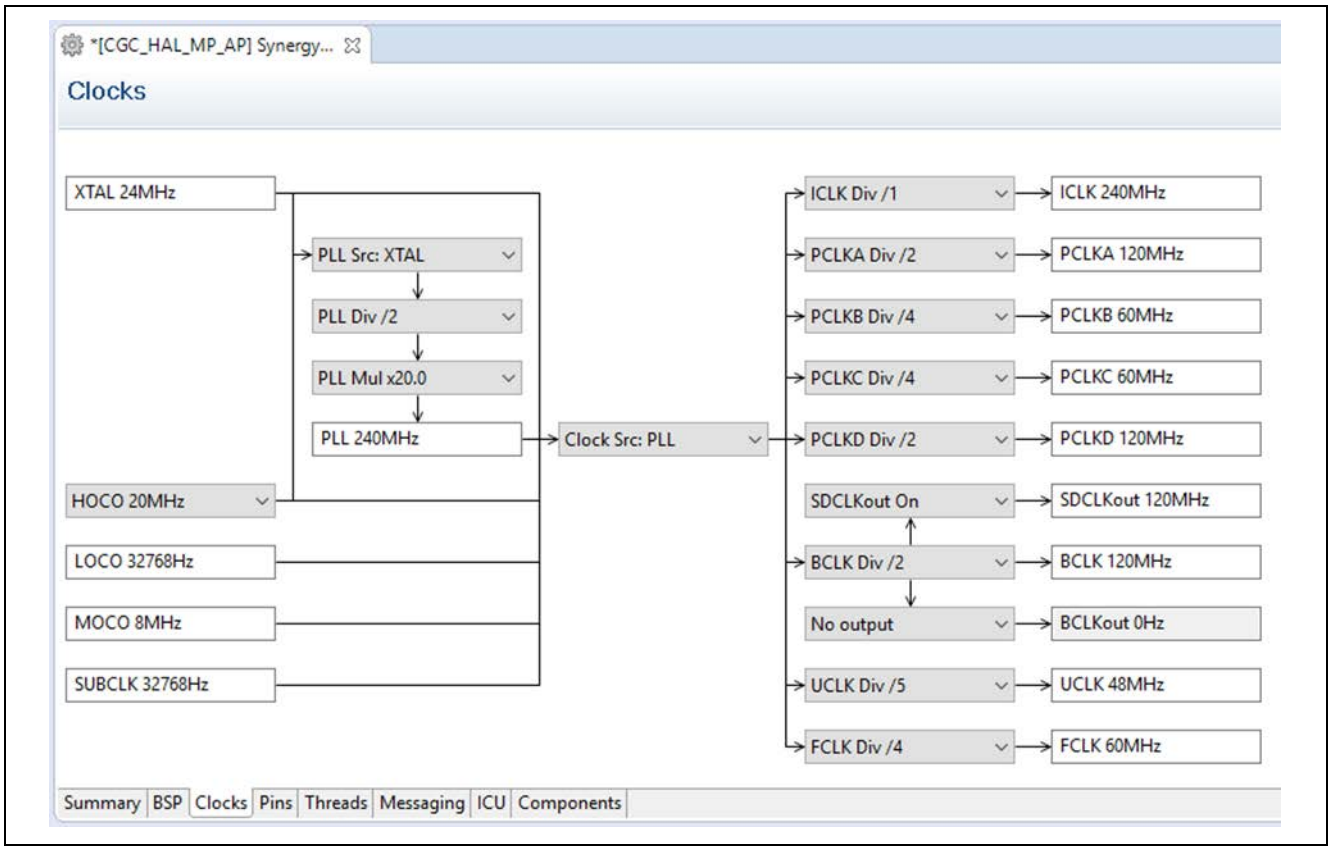

Figure 7. Enabling / Disabling SDCLK & BCLK via the Clock tab

In this example, SDRAM Clock is enabled, BUS Clock is disabled.

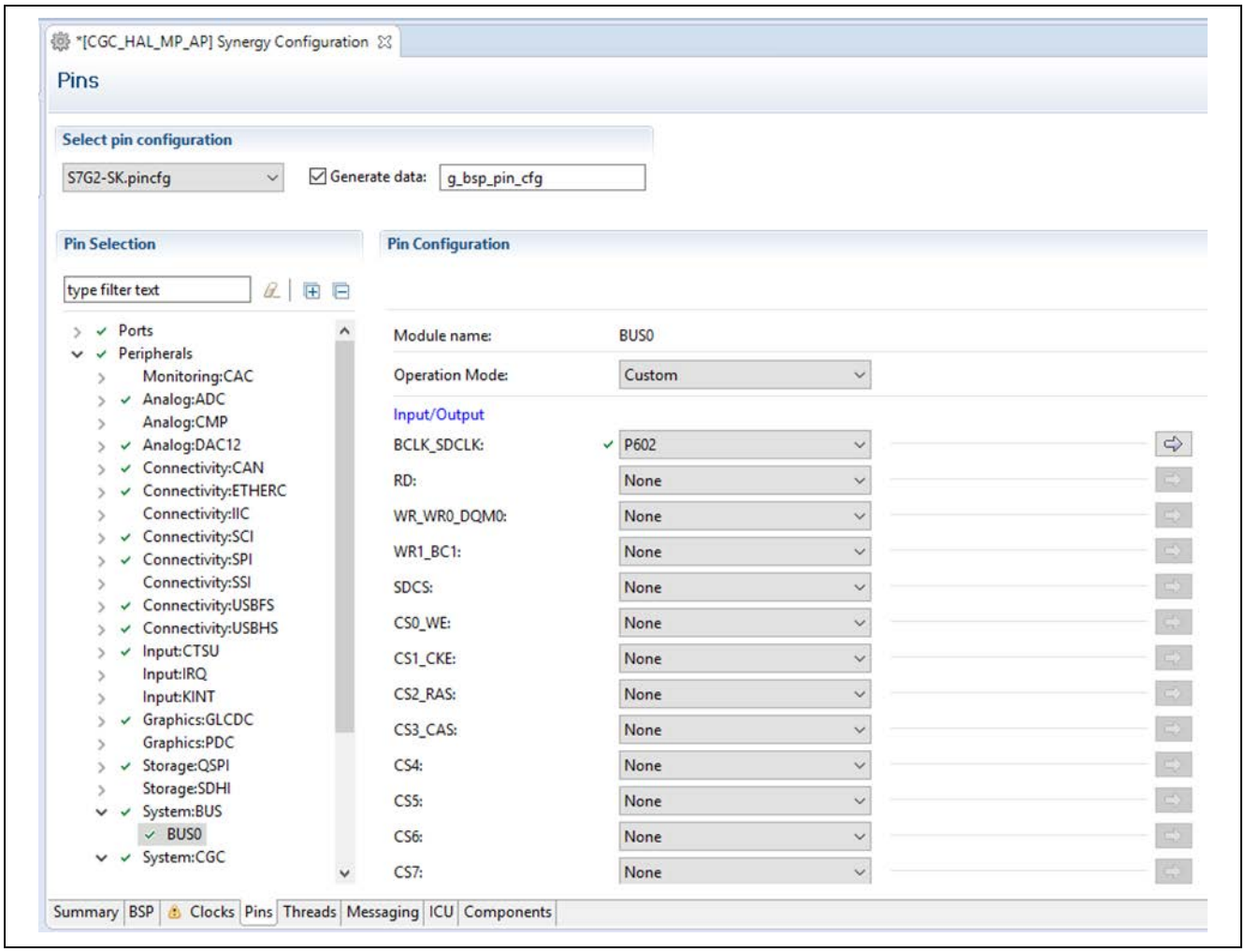

Figure 8. Enabling / Disabling SDCLK & BCLK via the Pins tab

In this example, SDRAM Clock / BUS Clock is enabled on pin P602.

Additional pin settings associated to the CGC allow for the enabling / disabling of the external oscillator pins, and setting the system Clock Out pin.

Renesas Synergy devices can run from its on-chip oscillators, in which case there is no requirement for the main clock external oscillator pins XTAL and EXTAL. These could be used as input pins by the application. The functionality of the sub clock external oscillator pins XCIN and XCOUT is fixed.

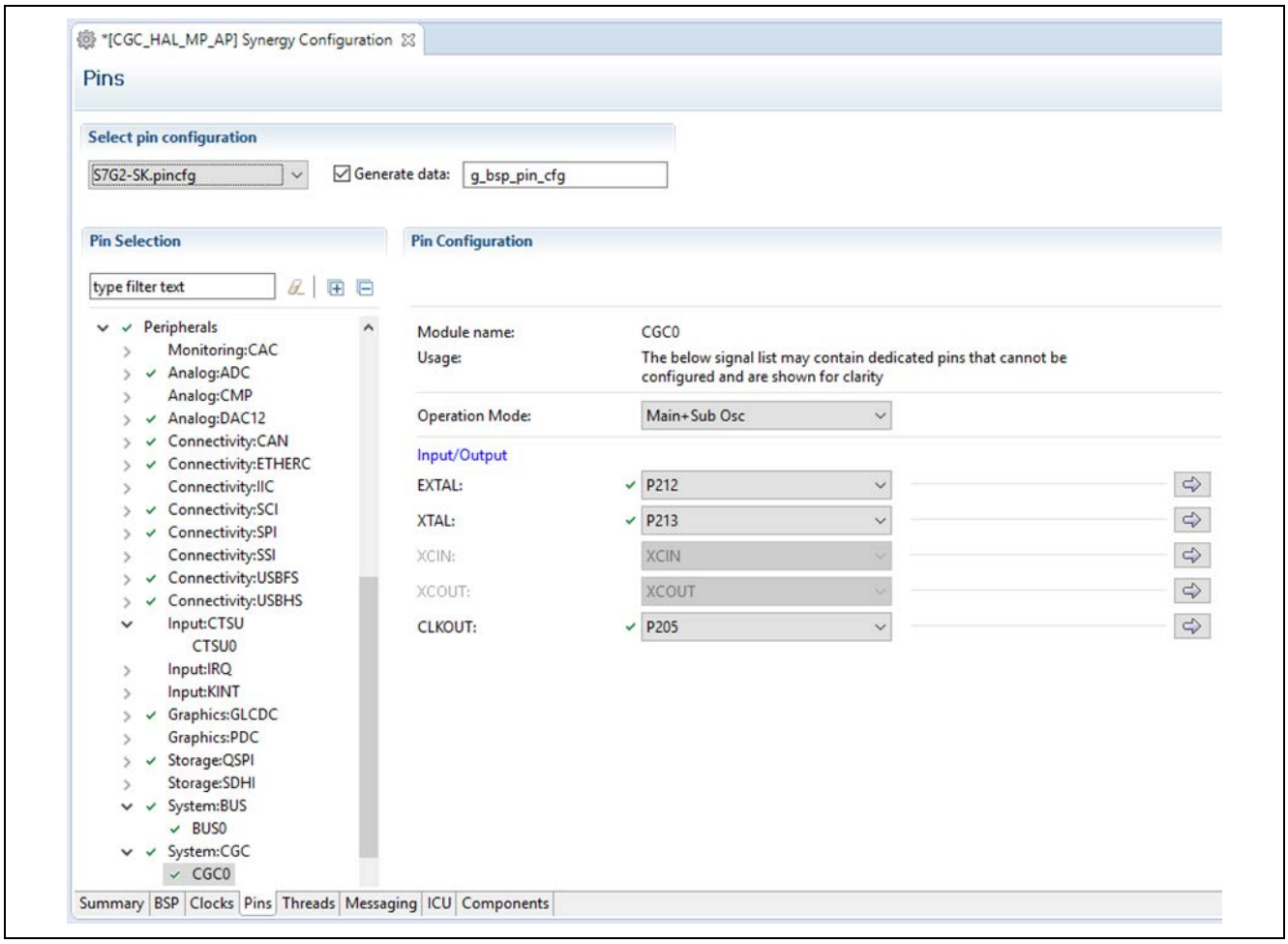

Figure 9. Enabling / Disabling EXTAL, XTAL and CLKOUT via the Pins tab

In this example, an external Main Oscillator is used via pins P212 and P213 and the CLKOUT is enabled on P205.

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

6. Using the CGC Module in an Application

The typical steps in using the CGC HAL module in an application are:

1. The CGC is automatically set after Reset.
2. Configure the clock as desired using the `clocksCfg` API.
3. Start clocks using the `clockStart` API.
4. Stop clocks using the `clockStop` API.
5. Other CGC functions as needed.

The following diagram illustrates common steps in a typical operational flow:

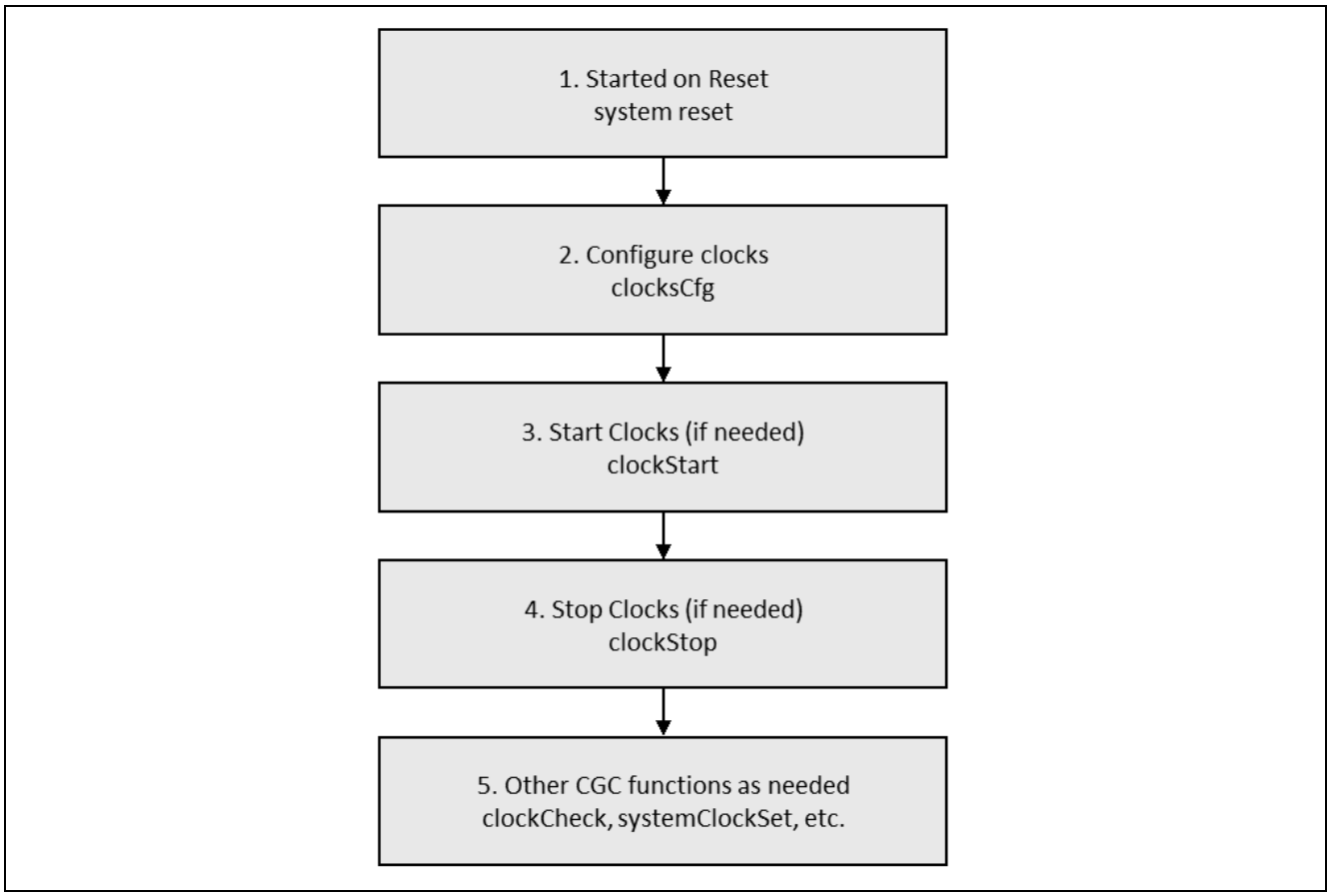

Figure 10. Flow Diagram of a Typical CGC HAL Module Application

7. The CGC HAL Module Application Project

The application project associated with this module guide demonstrates the steps in a full design. You may want to import and open the application project within the ISDE and view the configuration settings for the CGC HAL module. You can also read over the code (in `cgc_hal.c`) that illustrates CGC HAL module APIs in a complete design.

The application project demonstrates the use of the CGC APIs. The application project initializes a General Purpose Timer (GPT) to generate an interrupt, the initial frequency of which is based upon the default settings in the SSP configurator. The interrupt callback toggles one of three LEDs, depending on the chosen clock mode. The application modifies the clock generation circuit parameters, changing the clock frequency of the system, which is easily noticeable by the change in the blinking frequency of the LEDs. The application project also initializes the oscillator stop detect and two IRQ interrupts: IRQ10 and IRQ11. The oscillator stop detect ISR enters a while(1) loop to toggle the LEDs in an alternate on – off pattern that indicates to the user the application has detected an oscillator stop condition. The IRQ10 and IRQ11 ISR are generated by the user pushbuttons S5 and S4, respectively. These IRQs enable and disable the output of the system clock via the Clock Out pin.

Table 6. Software and Hardware Resources Used by the Application Project

Resource	Revision	Description
e^2 studio	5.3.1 or later	Integrated Solution Development Environment
SSP	1.2.0 or later	Synergy Software Platform
IAR EW for Renesas Synergy	7.71.2 or later	IAR Embedded Workbench® for Renesas Synergy™ (IAR EW for Synergy)
SSC	5.3.1 or later	Synergy Standalone Configurator
SK-S7G2	v3.0 to v3.1	Starter Kit

A simple flow diagram of the Application project is given in the following figures:

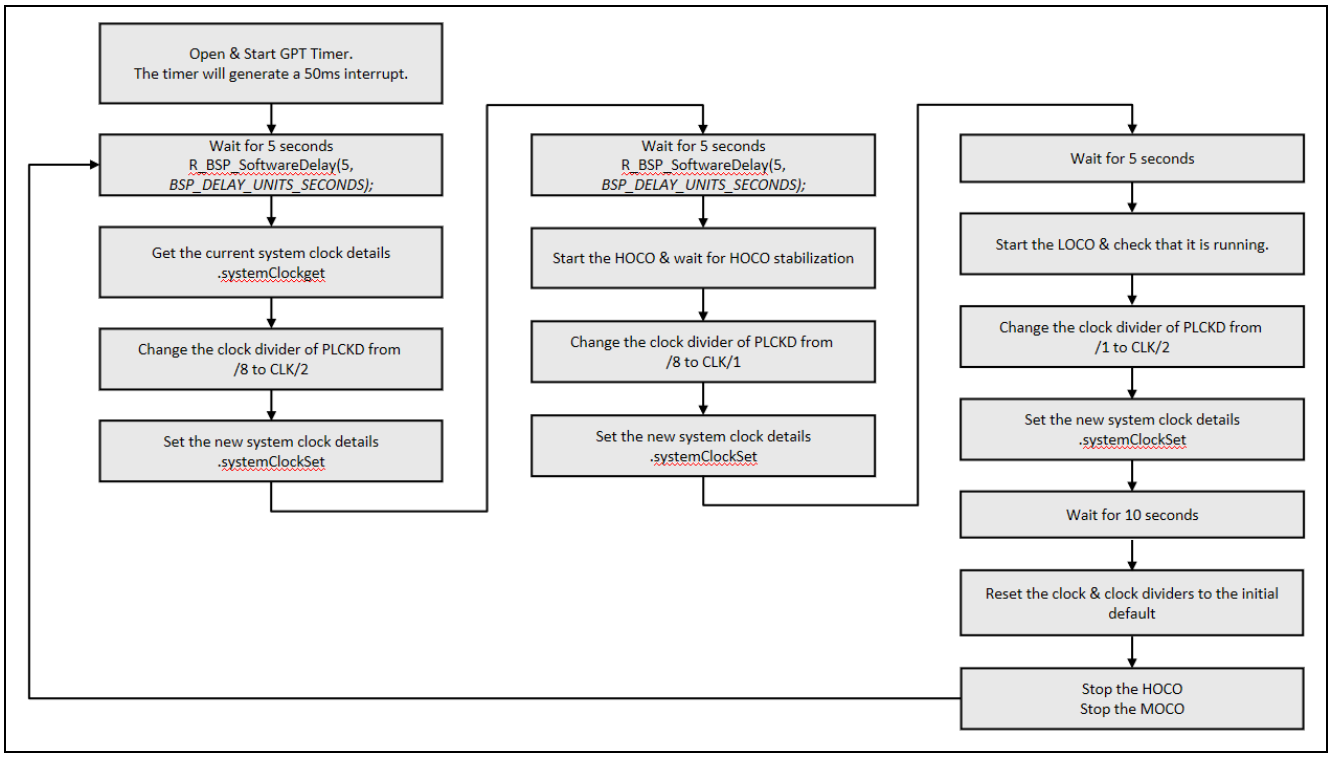

Figure 11. CGC HAL Module Application Project Flow Diagram

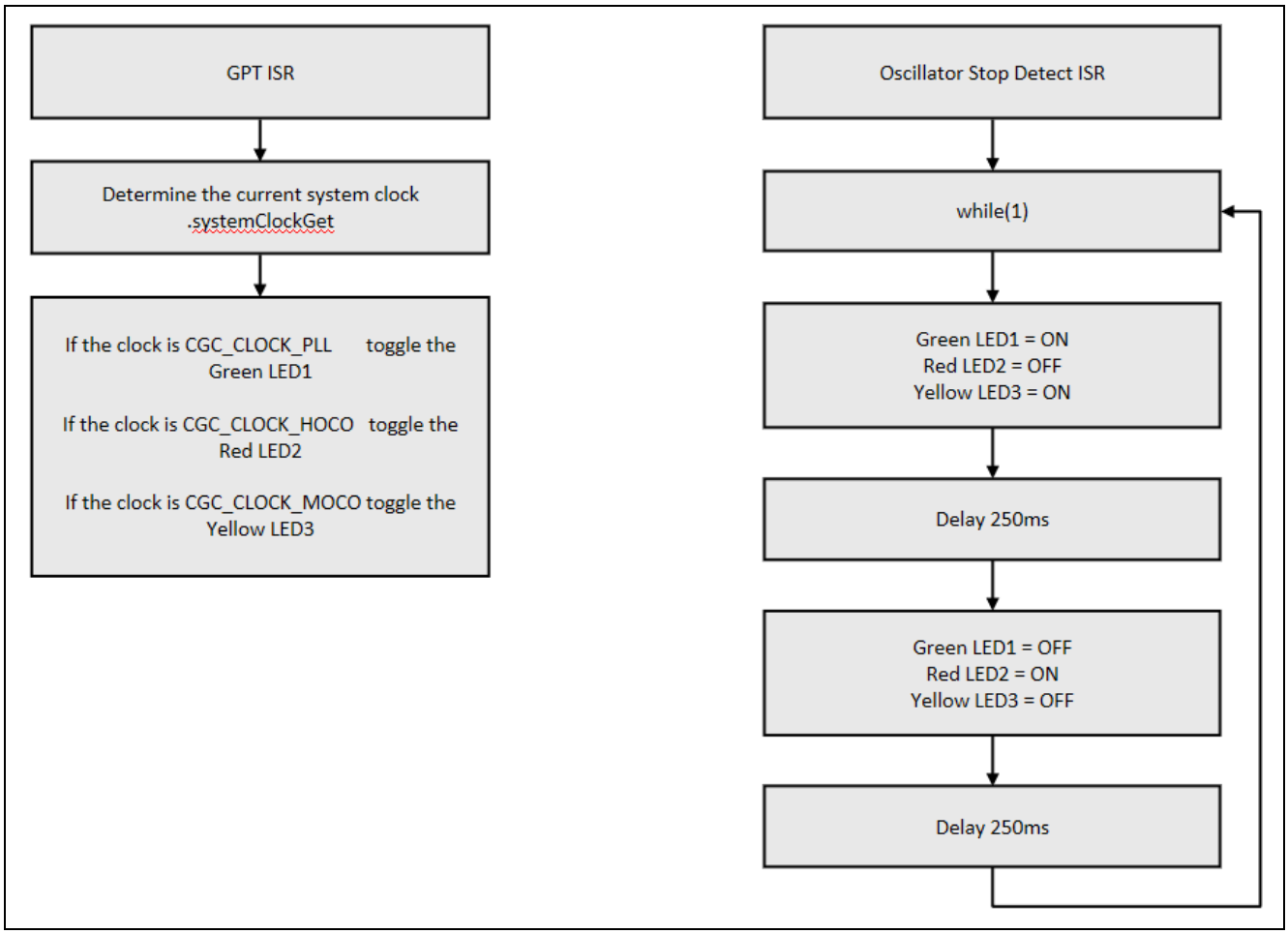

Figure 12. CGC Application Project GPT and Oscillator Stop Detect ISRs

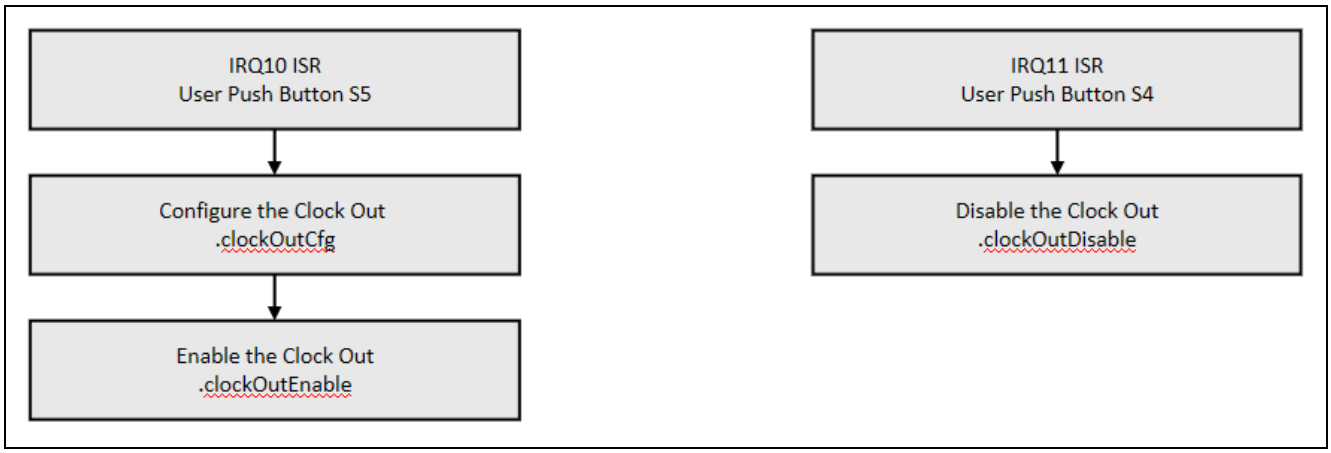

Figure 13. CGC Application Project User Push Button ISRs

The `cgc_hal.c` file in the project is located 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 application project starts execution with the initial system clock parameters as set in the clock configuration settings of the SSP. Note that the clock source is the PLL providing a CPU frequency of 240 MHz, and amongst others, a frequency of PLL/2 = 120 MHz for PCLKD (Peripheral Clock D). The GPT timer peripheral is clocked by PCLKD.

The default configuration of PCLKA is /2 but it is set to /4 for this example project. This is because PCLKD is also set to /4 during this example and, in this chip, the PCLKA divisor must never be set lower than the PCLKD divisor.

The first action is to open and start one of the GPT timers. GPT is configured to generate a 50 ms interrupt. The interrupt callback function toggles the LEDs.

The application then waits 5 seconds, in the interim the green LED blinks with a frequency of 10 Hz. The system clock configuration parameters are then read and the PCLKD divisor changes from /2 to /4. This new value is written to the CGC. The application then waits for 5 seconds, in the interim the green LED blinks with a frequency of 2.5 Hz.

Next, the clock source changes from the crystal oscillator with the PLL to HOCO, which is configured to run at 20 MHz. At the same time, the PCLKD divisor is changed from /4 to /1, which changes the PCLKD frequency. The application then waits 5 seconds, in the interim the red LED blinks with a frequency of 1.66 Hz.

The clock source next changes HOCO to MOCO, which is configured to run at 8 MHz. At this time the PCLKD divisor changes from /1 to /2, changing the PCLKD frequency. The application then waits 10 seconds, in the interim the yellow LED blinks with a frequency of 0.333 Hz.

The clock source is then set back to the starting defaults and the HOCO and LOCO are stopped. The process then repeats.

If at any time the external clock oscillator stops, Oscillator Stop Detect ISR generates and applications enter a while(1) loop; toggling the LEDs in the pattern LED1 = ON, LED2 = OFF, LED3 = ON, wait for 250 MS, LED1 = OFF,
LED2 = ON, LED3 = OFF, wait for 250MS.

If the user pushbutton S4 is pressed, Clock Out is enabled. If the application is running from the PLL (indicated by the Green LED1 toggling), then the output is the 24 MHz MAIN CLOCK. It is not possible to output the PLL clock via the clock out pin. If the application is running from HOCO (indicated by the RED LED2 toggling), then the output is the 20 MHz MAIN CLOCK. If the application is running from MOCO (indicated by the Yellow LED3 toggling), then the output is the 8 MHz MAIN CLOCK. If the user pushbutton S5 is pressed, Clock Out is disabled.

Note: To observe the clock out on port pin P205, such as on an oscilloscope, capacitor C63 must be removed. Otherwise, the clock-out signal is only observed as a voltage of different levels depending on the output clock. For more information, refer to Running the CGC HAL Module Application Project section.

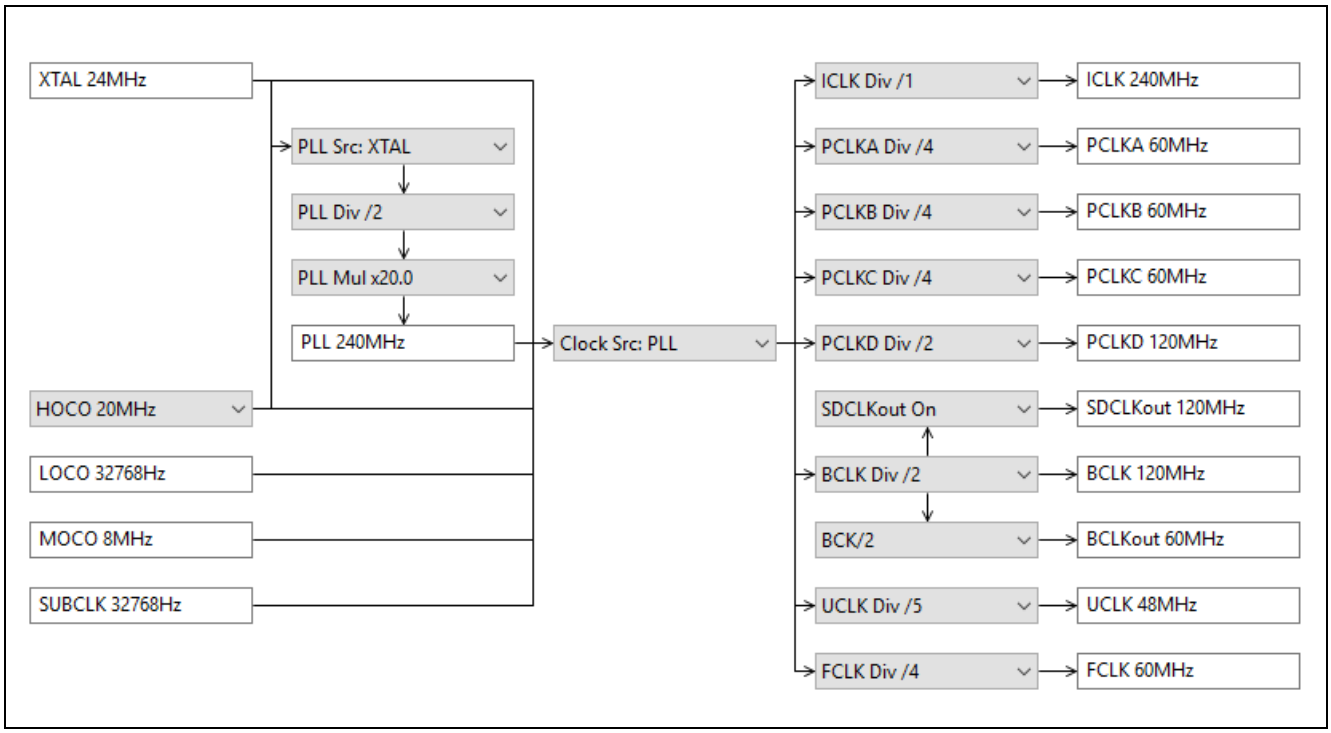

Figure 14. Default CGC settings for application project

As the CGC HAL module is in the HAL/Common Thread stack by default, typically there is no need to change any settings in its configuration. For the application project, a few CGC properties and additional modules must be configured. The following table lists properties with the values set for this specific project. You can also open the application project and view these settings in the Properties window as a hands-on exercise.

Table 7. GPT Timer HAL Module Configuration Settings for the Application Project

ISDE Property	Value Set
Name	g_timer
Channel	0
Mode	Periodic
Period Value	50
Period Unit	Milliseconds
Duty Cycle Value	50
Duty Cycle Unit	Unit Percent
Auto Start	True
GTIOCA Output Enabled	False
GTIOCA Stop Level	Pin Level Low
GTIOCB Output Enabled	False
GTIOCB Stop Level	Pin Level Low
Callback	g_timer_callback
Interrupt Priority	Priority 8(CM4: valid, CM0+: invalid)

Table 8. External IRQ HAL Module (IRQ10) Configuration Settings for the Application Project

ISDE Property	Value Set
Name	g_external_irq10
Channel	10
Trigger	Falling
Digital Filtering	Disabled
Interrupt enabled after initialization	True
Callback	cb_irq10
Interrupt Priority	Priority 8(CM4: valid, CM0+: invalid)

Table 9. External IRQ HAL Module (IRQ11) Configuration Settings for the Application Project

ISDE Property	Value Set
Name	g_external_irq11
Channel	11
Trigger	Falling
Digital Filtering	Disabled
Interrupt enabled after initialization	True
Callback	cb_irq11
Interrupt Priority	Priority 8(CM4: valid, CM0+: invalid)

Table 10. Pin configuration

Pin Selection Sequence	Setting
Ports > P6 > P600	Mode: Output mode (Initial Low)
Ports > P6 > P601	Mode: Output mode (Initial Low)
Ports > P6 > P602	Mode: Output mode (Initial Low)
Peripherals > Input:IRQ > IRQ10	P005
Peripherals > Input:IRQ > IRQ11	P006
Peripheral > System:CGC > CLCKOUT	P205 (On the SK-S7G2 board, pin P205 is also used as a CTSU pin and must be disabled. See the following entry)
Peripheral > Input:CTSU	Disabled

8. Customizing the CGC HAL Module for a Target Application

Since the CGC does not require any specific configuration, except defaults, developers can easily adjust the CGC to their own requirements. For example, it is possible to choose the system clock source from various options (external crystal oscillator, HOCO, LOCO, MOCO, or SUBCLK); PLL parameters (divisor, multiplier); or, clock divisors to adjust the clock frequencies, as needed.

9. Running the CGC HAL Module Application Project

To run the CGC 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.

See the package document, *Renesas Synergy Project Import Guide* (r11an0023eu0121-synergy-ssp-import-guide.pdf), for instructions on importing the project into e^2 studio or IAR EW for Synergy, and then build and run the application.

Note: The following steps are described sufficiently well for someone experienced with the basic flow through the Synergy development process. If these steps are unfamiliar, refer to the first few chapters of the SSP User's Manual for a description of how to accomplish these steps.

To create and run the CGC HAL module application project, use the following steps:

1. Import and build the example project included with this module guide
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 on the green, red, and yellow LEDs.
5. The clock out can be viewed on port pin P205. Note on SK-S7G2, the CTSU peripheral uses P205 and it is taken to ground via a 0.1µF capacitor (C63). To observe the clock out, such as on an oscilloscope, C63 must be removed. Otherwise, the clock out signal is only observed as a voltage of different levels, depending on the output clock:

Clock Out	Observed voltage on P205
Main Clock (24MHz)	1.64V
HOCO (20MHz)	1.50V
MOCO (8MHz)	1.48V

The clock's diagnostic output can be viewed on the debug console. The expected output from a successful run should look like the following text:

```
cgc_hal_module_guide_project

Press button S4 to enable Clock Out and S5 to disable Clock Out

Start the SUBCLOCK
5 second delay...
Setting PCLKD divisor to 4
5 second delay...
Changing input from PLL to HOCO
Waiting for HOCO to be stable...
HOCO is stable
Setting PCLKD divisor to 1
5 second delay...
Changing input from HOCO to MOCO
Waiting for MOCO to be stable...
MOCO is stable
Setting PCLKD divisor to 2
5 second delay...
Resetting clock settings to their original values
```

10. CGC 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 Synergy Platform makes these steps less time consuming and removes common errors, like conflicting configuration settings or incorrect selection of lower-level drivers. Use of high-level APIs (as demonstrated in the application project) illustrates the development time savings in 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. CGC HAL Module Next Steps

After you have mastered a simple CGC HAL module project, you may want to review a more complex example. In particular, exploring various power saving options available with the Synergy Platform, since many times these options are related to clock-control functions. Explore the Power Profiles and Low Power Mode-related module guides for additional examples related to clock control.

12. CGC 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_cgc module reference materials and resources are available on the Synergy Knowledge Base: https://en-support.renesas.com/search/r_cgc%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	May.22.17	–	Initial version
1.01	Aug.23.17	–	Update to Hardware and Software Resources Table
1.02	Jan.07.19	–	Changed the values of PCLKA and PCLKD divisors

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/.