Renesas Synergy™ Platform

AGT HAL Module Guide

R11AN0176EU0101
Rev.1.01
Dec 12, 2018

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 AGT HAL module is a high-level API for timing applications and is implemented on `r_agt`. The AGT HAL module uses the AGT peripheral on the Synergy MCU. A user-defined callback can be created to respond to a timer event.

The AGT HAL module configures a timer to a user-specified period. When the period elapses, any of the following events can occur:

- Interrupt the CPU, which calls a user-callback function (if provided)
- Toggle a port pin
- Transfer data using DMAC/DTC (if configured with transfer interface)
- Start another peripheral (if configured with events and peripheral definitions).

Contents

1. AGT HAL Module Features

- Multiple channels: 16-bit x 2 channels
  - Channel 1 can be clocked by the channel 0 underflow, creating a cascaded 32-bit timer
- Core clock:
  - Can be clocked using PCLKB, LOCO, or Fsub. When clocked by LOCO or Fsub, it can be used to wake up the MCU from sleep modes.

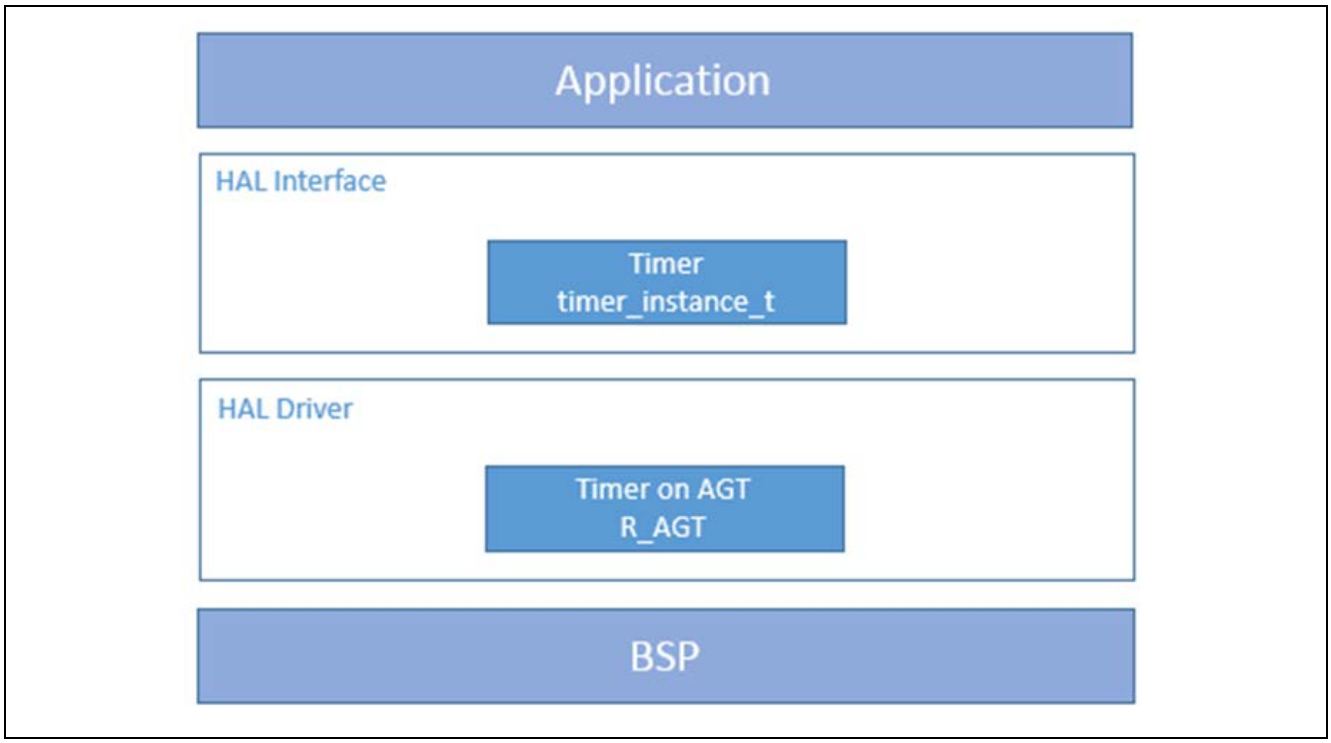

Figure 1 AGT HAL Module Block Diagram

2. AGT HAL Module APIs Overview

The AGT HAL module defines APIs for opening, closing, starting and stopping timers. 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 follows the API summary table.

Table 1 AGT HAL Module API Summary

Function Name	Example API Call and Description
.open	`g_timer0.p_api->open(g_timer0.p_ctrl, g_timer0.p_cfg)`Initial configuration.
.stop	`g_timer0.p_api->stop(g_timer0.p_ctrl)` Stop the counter.
.start	`g_timer0.p_api->start(g_timer0.p_ctrl)` Start the counter.
.reset	`g_timer0.p_api->reset(g_timer0.p_ctrl)` Reset the counter initial value.
.counterGet	`g_timer0.p_api->counterGet(g_timer0.p_ctrl, &value)` Get current counter value and store it in the provided pointer, value.
.periodSet	`g_timer0.p_api->periodSet(g_timer0.p_ctrl, period, unit)` Set the time until the timer expires.

Function Name	Example API Call and Description
.dutyCycleSet	`g_timer0.p_api->dutyCycleSet(g_timer0.p_ctrl, period, unit, pin)` Sets the time until the duty cycle expires on the defined pin.
.infoGet	`g_timer0.p_api->infoGet(g_timer0.p_ctrl, &info)` Get the time until the timer expires in clock counts and store it in provided pointer, info.
.close	`g_timer0.p_api->close(g_timer0.p_ctrl)` Allows driver to be reconfigured and may reduce power consumption.
.versionGet	`g_timer0.p_api->versionGet(&version)` Retrieve the API version with the version pointer.

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	Operation is successful
SSP_ERR_ASSERTION	Parameter is NULL or configuration setting is not allowed
SSP_ERR_IN_USE	The channel specified is already open
SSP_ERR_IRQ_BSP_DISABLED	A required interrupt is not enabled in the BSP
SSP_ERROR_NOT_OPEN	The channel is not open
SSP_ERR_INVALID_ARGUMENT	Invalid argument provided
SSP_ERR_INVALID_HW_CONDITION	Invalid hardware setting detected
SSP_ERR_INVALID_PTR	A pointer parameter was NULL, but needed a non-NULL value

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. AGT HAL Module Operational Overview

The AGT HAL module configures a timer to a user-specified period. When the period elapses, the CPU can be interrupted, a port pin can be toggled, a transfer of data using the DMAC or DTC can be initiated or another peripheral can be triggered to begin operation.

The following flowchart shows toggling a port pin or generating a CPU interrupt after a specified period. This flowchart is appropriate for both AGT and GPT counters.

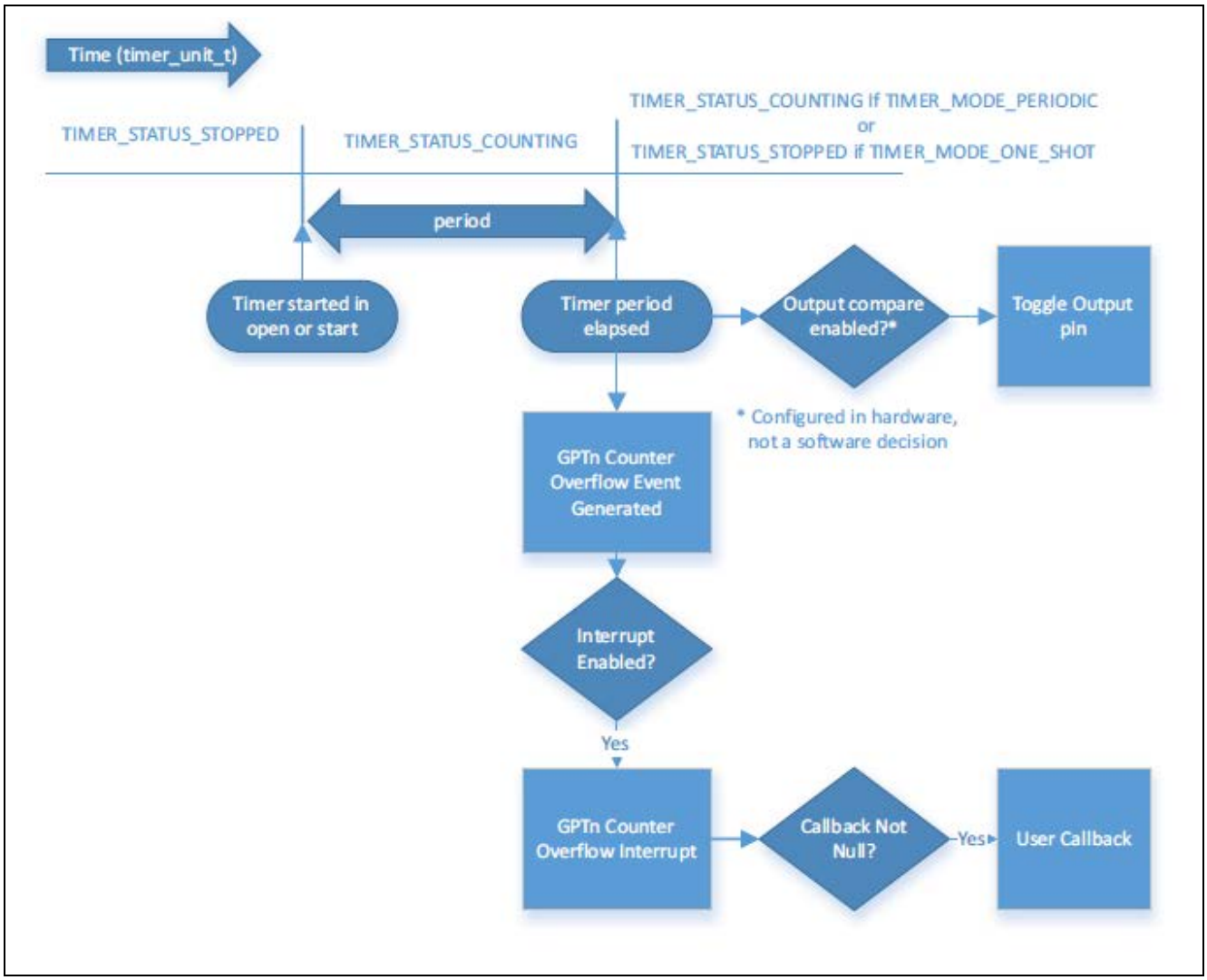

Figure 2 GPT Timer-Periodic or One-Shot Mode

Two different timer modules, the GPT and the AGT, are supported in the SSP. The following sections provide information on both modules so that the developer can compare and contrast the capabilities of each module for a particular application. For additional information on the GPT, refer to the *GPT User's Guide*.

The GPT module is recommended for most generic timer applications, but either module can be used for a basic timer functionality. The use cases in which one timer module would be preferred over the other are described as follows.

Selecting the GPT Timer Module

The GPT module uses a high-resolution 32-bit counter that can only be clocked by PCLKA. There are more GPT channels than AGT channels on Synergy devices, so using the GPT is less likely to cause a resource conflict.

Selecting the AGT Timer Module

The AGT module uses a 16-bit counter that can be clocked by PCLKB, LOCO, or Fsub. If clocked by LOCO or Fsub, the AGT interrupt can be used to wake the MCU from sleep modes. There are two channels, channel 0 and channel 1. Channel 1 can be clocked by channel 0 underflow, effectively creating a 32-bit cascaded timer.

3.1 AGT HAL Module Operational Notes

The maximum time period depends on the timer type and the input clock frequency.

- On a GPT with 32-bit resolution with PCLKA running at 120 MHz, the maximum period is approximately 36,650 seconds, which is just over 10 hours
- On a GPT with 16-bit resolution with PCLKA running at 32 MHz, the maximum period is approximately 2.09 seconds
- On an AGT with 16-bit resolution with PCLKB running at 60 MHz, the maximum period is approximately 8.7 ms
- On an AGT with 16-bit resolution with Fsub running at 32 kHz, the maximum period is approximately 2.0 seconds.

Note: The maximum time period is calculated based on the clock frequency supplied, the divisor for the clock frequency, and the resolution of the timer. The divisor is calculated internally in the driver according to the time period suppled in the configurator.

For example, for a GPT with a 32-bit resolution and a 120 MHz clock, the divisor calculated is 1024 in the driver. Using these values in the formula T = (1 / (clock frequency/Divisor)) results in the time period for one timer tick. To get the total delay, use the formula Delay = (Resolution of the timer used)*T. This results is the time period of nearly 10 hours for the first case of GPT. The same can be applied to the AGT.

In the following situations, in the BSP enable the AGT counter underflow interrupt for the selected channel used:

1. To get a software interrupt when the timer period has elapsed.
2. To use one-shot mode.

When the AGT*n* AGTI interrupt is enabled in the BSP, the corresponding ISR is defined in the timer driver. The ISR calls a user-callback function if one was registered in open.

Note: Interrupts may be skipped when used with the DTC peripheral with the IRQ set to TRANSFER_IRQ_END.

AGT Output Timer Signal

If the timer output is configured, (AGTO Output Enabled set to true) the output pin starts at a high level if the output inverted is configured to True and a low level if it is configured to False. The output pin toggles every time the period elapses, beginning with the first time the period elapses after the timer is started.

In one-shot mode, the output is also configured to toggle when the timer starts counting. This generates a pulse - the timer toggles from the stop level when counting begins and toggles back to the stop level when counting ends.

Timer Period Calculation

The timer period is defined as the time until the timer expires. When output compare is used, the output pin toggles once per period, so the traditional period(from rising edge to rising edge of the output pin) is twice the period(timer period) specified in the software.

Runtime period calculation based on the current clock settings is available from `open` and `periodSet`.

If the specified timer period is different than the raw counts, the period is calculated using the current timer clock frequency (see Configuring the GPT Clocks or Configuring the AGT Clocks). The current timer clock frequency is determined using `systemClockFreqGet`. This frequency is used in the appropriate formula from the following table, as **clk_freq_hz**.

Table 3 Timer period calculation

Timer Units	Description
TIMER_UNIT_PERIOD_NSEC	Counts = (period * clk_freq_hz)/ 1,000,000,000
TIMER_UNIT_PERIOD_USEC	Counts = (period * clk_freq_hz)/ 1,000,000
TIMER_UNIT_PERIOD_MSEC	Counts = (period * clk_freq_hz)/ 1,000
TIMER_UNIT_PERIOD_SEC	Counts = (period * clk_freq_hz)
TIMER_UNIT_FREQUENCY_HZ	Counts = (clk_freq_hz)/ period
TIMER_UNIT_FREQUENCY_KHZ	Counts = (clk_freq_hz)/ (1,000 * period)

If the requested period is larger than the counter size (32-bit or 16-bit), the driver selects the smallest divisor that allows the result to fit in the counter size. If the counter value is larger than the counter size with the largest divisor (1,024), an error code (SSP_ERR_INVALID_ARGUMENT) is returned.

Triggering DMAC/DTC with GPT

To trigger a transfer of data using the DMAC or DTC peripheral when the timer period elapses, configure the DMAC/DTC transfer with `activation_source` set to ELC_EVENT_GPTn_COUNTER_OVERFLOW (where n is the GPT channel number). See the DMAC or DTC guides for further information.

Note: If you use the timer in one-shot mode with the DTC, the entire transfer completes before the interrupt stops the timer if the IRQ is set to TRANSFER_IRQ_END. To generate only one transfer after the timer period elapses, set the IRQ to TRANSFER_IRQ_EACH or use the DMAC for the transfer.

Triggering ELC Events with GPT

The GPT timer can trigger the start of other peripherals. The ELC guide provides a list of all available peripherals.

Triggering DMAC/DTC with AGT

To trigger a transfer of data using the DMAC or DTC peripheral when the timer period elapses, configure the DMAC/DTC transfer with `activation_source` set to ELC_EVENT_AGTn_AGTI (where n is the AGT channel number). See the Transfer interface for further information.

Note: If you use the timer in one-shot mode with the DTC, the entire transfer completes before the interrupt stops the timer if the irq is set to TRANSFER_IRQ_END. To generate only one transfer after the timer period elapses, set the irq to TRANSFER_IRQ_EACH, or use the DMAC for the transfer.

Triggering ELC Events with AGT

The AGT timer can trigger the start of other peripherals. The ELC module guide provides a list of all available peripherals listed in `elc_peripheral_t`. See the events and peripheral definitions for further information.

Cascaded AGT Timers creating a 32-bit timer

AGT channel 1 can be clocked by the AGT Channel 0 underflow, creating a cascaded 32-bit timer.

3.2 AGT HAL Module Limitations

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

4. Including the AGT HAL Module in an Application

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

Note: This section assumes 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, 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.

To add the AGT 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 AGT Driver is `g_timer0`. This name can be changed in the associated Properties window.)

Table 4 AGT Driver Selection Sequence

Resource	ISDE Tab	Stacks Selection Sequence
r_timer0 Timer Driver on r_agt	Threads	New Stack> Driver> Timers> Timer Driver on r_agt

In the following figure, when the AGT HAL module on `r_agt` is added to the thread stack, the configurator automatically adds any needed lower-level drivers. Any drivers that need additional configuration information are box text highlighted in red.

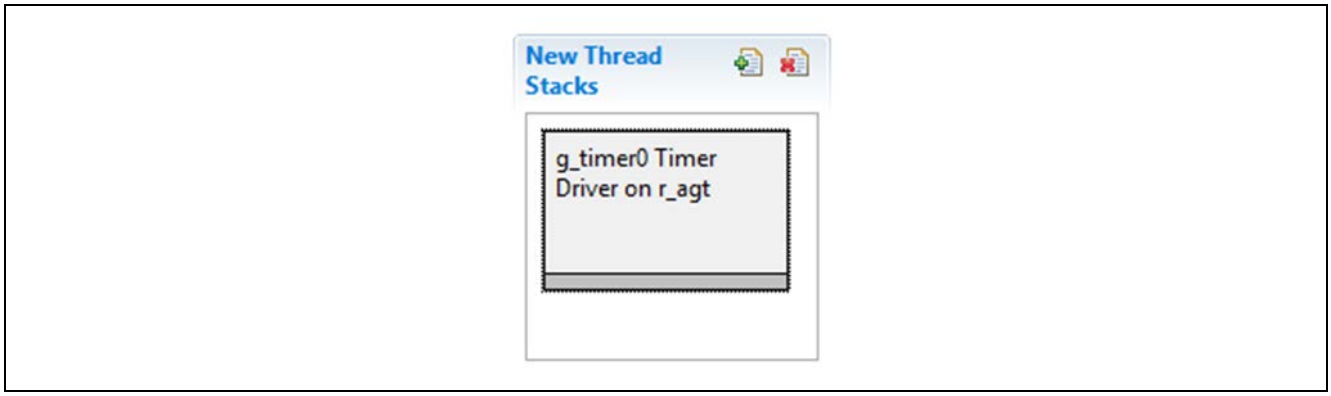

Figure 3 AGT HAL Module Stack

5. Configuring the AGT HAL Module

The AGT 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 are not available for changes. These properties are identified with a lock icon for the 'locked' property in the window. This approach simplifies the configuration process and makes it much less error-prone than previous 'manual' approaches. 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 and then view the Properties window. The interrupt settings are often toward the bottom of the properties list, so scroll down until they become available. Also note that the interrupt priorities listed in the Properties window include an indication as to the validity of the setting based on the targeted MCU (CM4 or CM0+). This level of detail is not included in the following configuration properties tables, but is easily visible with the ISDE when configuring 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 following configuration table settings. This helps to 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 AGT HAL Module on r_agt

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	Enables or disables parameter checking.
Name	g_timer0	Module name.
Channel	0	Physical hardware channel.
Mode	Periodic, One shot (Default: Periodic)	**Warning:** One-shot functionality is not available in the GPT hardware, so it is implemented in software by stopping the timer in the ISR called when the period expires. For this reason, ISRs must be enabled for one-shot mode even if the callback is unused.
Period Value	10	See Timer Period Calculation

ISDE Property	Value	Description
Period Unit	Raw Counts, Nanoseconds, Microseconds, Milliseconds, Seconds, Hertz, Kilohertz (Default: Microseconds)	See Timer Period Calculation
Auto Start	True, False (Default: True)	Set to true to start the timer after configuring or false to leave the timer stopped until timer_api_t::start is called.
Count Source	PCLKB, PCLKB/8, PCLKB/2, LOCO, AGT0 Underflow, AGT0 fSub (Default: PCLKB)	The clock source for the AGT counter.
AGTO Output Enabled	True, False (Default: False)	Set to true to output the timer signal on a port pin configured for AGT (AGTO pin). Set to false for no output of the timer signal.
AGTIO Output Enabled	True, False (Default: False)	Set to true to output the timer signal on a port pin configured for AGT (AGTIO pin). Set to false for no output of the timer signal.
Output Inverted	True, False (Default: False)	Set to false to start the output signal low. Set to true to start the output signal high.
Enable Comparator A output pin	True, False (Default: False)	Enable for comparator A
Enable Comparator B output pin	True, False (Default: False)	Enable for comparator B
Callback	NULL	A user callback function can be registered in timer_api_t::open. If this callback function is provided, it is called from the interrupt service routine (ISR) each time the timer period elapses. **Warning:** Since the callback is called from an ISR, care should be taken not to use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.
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) (Default: Disabled)	Timer interrupt priority. 0 is the highest priority.

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

In some cases, settings other than the defaults can be desirable. For example, it might be useful to select different period or duty-cycle values. The configurable properties for the lower-level stack modules are provided in the following sections for completeness and as a reference.

5.1 AGT HAL Module Clock Configuration

The AGT timer is clocked based on the PCLKB, LOCO, Fsub, or AGT Underflow frequency. The AGT clock is selectable in the Properties window in e^2 studio. You can set clock frequencies using the clock configurator in e^2 studio Configuring Clocks or by using the CGC interface at run-time.

5.2 AGT HAL Module Pin Configuration

The AGT peripheral module uses pins on the MCU to communicate to external devices. I/O pins must be selected and configured as required by the external device. The following table illustrates the method for selecting the pins within the SSP configuration window and the subsequent table illustrates an example selection for the associated pins.

Note: The operation mode selection determines what peripheral signals are available and what MCU pins are required.

Table 6 Pin Selection Sequence for AGT HAL Driver

Resource	ISDE Tab	Pin selection Sequence
AGT	Pins	Select Peripherals > Timer: AGT>AGTn

Note: The selection sequence assumes that AGT0 is the desired hardware target for the driver.

Table 7 Pin Configuration Settings for AGT HAL Driver

Pin Configuration Property	Value	Description
Operation Mode	Disabled, Custom, Timer Output, Compare Match, Count Measurement, Gated Count (Default: Disabled)	Select timer operation mode
AGTIO	None (Default: None)	AGTIO Pin
AGTO	None, P102 (Default: None)	AGTO Pin
AGTOA	None (Default: None)	AGTOA Pin
AGTOB	None (Default: None)	AGTOB Pin
AGTEE	None, P101 (Default: None)	AGTEE 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 AGT HAL Module in an Application

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

1. Initialize the AGT HAL module using the `open` API.
2. Start the AGT HAL module by calling the `start` API.
3. Read the counter value by calling the `counterGet` API.
4. Set the period value by using the `periodSet` API.
5. Set the duty cycle (duty_cycle_counts) by using `dutyCycleSet` API.
6. Get the timer information using `infoGet` API.
7. Respond to the AGT HAL module callback as needed.
8. Resets the counter value using the `reset` API.
9. Stop the AGT channel using `stop` API.
10. Use the `close` API to power down the peripheral.

Note: The timer-period and duty-cycle parameters can be reconfigured based on the application needs.

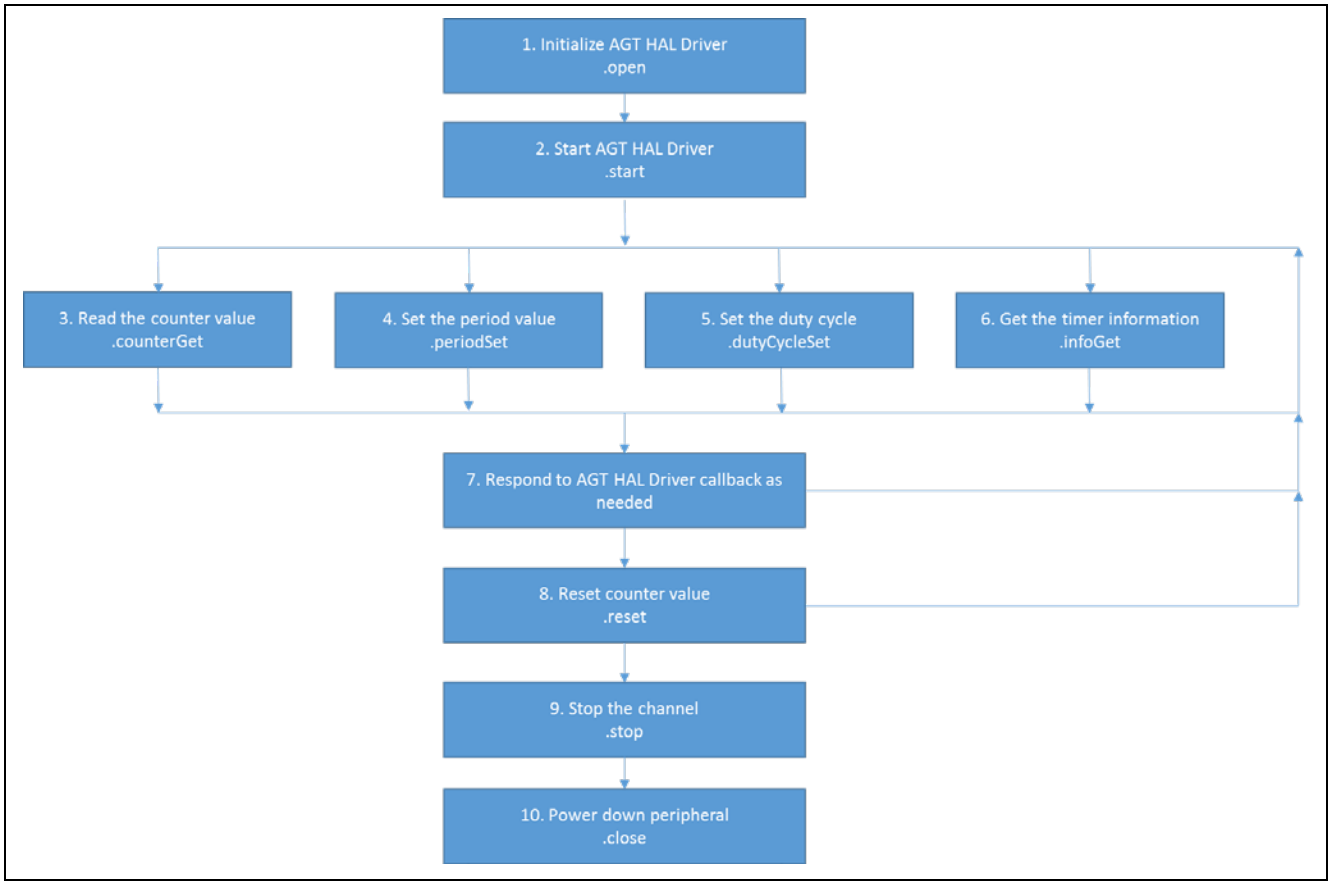

Figure 4 Typical AGT HAL Module Application Flow Chart

7. The AGT HAL Module Application Project

The application project associated with this module guide demonstrates the aforementioned steps in an application example. 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 AGT HAL module. You can also read over the code (in `AGT_HAL_MG_AP.c`) which is used to illustrate the AGT HAL module APIs in a complete design.

The application project demonstrates the typical use of the timer drivers APIs. Two AGT timers are used. The application project main thread entry initializes the timers and starts and reconfigures the period and duty cycle parameters. When either of the timer expires, their respective callbacks are called to toggle an LED. Functions to get information about the timers print the result on the Debug Console using the common semi-hosting function. The following table identifies the target versions for the associated software and hardware used by the application project.

Table 8 Software and Hardware Resources Used by the Application Project

Resource	Revision	Description
e^2 studio	6.2.1 or later	Integrated Solution Development Environment
SSP	1.5.0 or later	Synergy Software Platform
IAR EW for Synergy	8.23.1 or later	IAR Embedded Workbench® for Renesas Synergy™
SSC	6.2.1 or later	Synergy Standalone Configurator
SK-S7G2	v3.0 to v3.1	Starter Kit

The following flow diagram shows key steps in the application project.

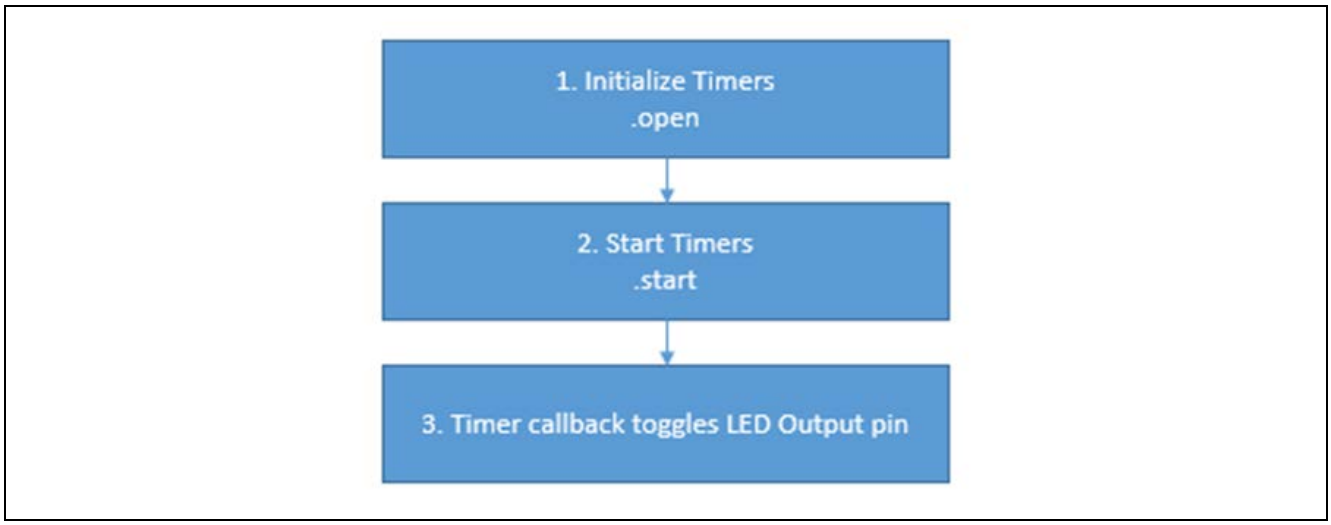

Figure 5 AGT HAL Module Application Project Flow Diagram

Two timers are used in this project:

1. g_timer_agt_0 is set up as a 16-bit periodic AGT.
2. g_timer_agt_1 is set up as a 32-bit periodic AGT, clocked by g_timer_agt_0.

The following tables show the configuration properties for each of the timers being used. Other settings use default values.

Table 9 Configuration Properties for g_timer_agt_0

ISDE Property	Value Set
Name	g_timer_agt_0
Channel	0
Mode	Periodic
Period Value	100
Period Unit	Microseconds
Auto Start	False
AGT0 Output Enabled	True
Interrupt Priority	Priority 5
Count Source	PCLKB

Table 10 Configuration Properties for g_timer_agt_1 (Source is AGT0 Underflow)

ISDE Property	Value Set
Name	g_timer_agt_1
Channel	1
Mode	Periodic
Period Value	1
Period Unit	Seconds
Auto Start	False
Callback	agt_32b_callback
AGT0 Output Enabled	True
Interrupt Priority	Priority 5
Count Source	AGT0 Underflow

In the Pin tabs, configure the following pins. These pins can be scoped to view the timer output:

1. Peripherals > Timer:AGT > AGT0 > Operation Mode: Timer Output
   Peripherals > Timer:AGT > AGT0 > AGTO > P102. Then expand on P102, to navigate to the P102 configuration page. This should be set to AGT0_AGTO.
2. Peripherals > Timer:AGT > AGT1 > Operation Mode: Timer Output
   Peripherals > Timer:AGT > AGT1 > AGTO > P205. Then expand on P205, to navigate to the P205 configuration page. This should be set to AGT1_AGTO.

Note: You may need to disable SPI0 and CTSU0 to free up AGTO on AGT0 and AGT1. The complete application project can be found using the link provided in the References section at the end of this document. The `AGT_HAL_MG.c` file is located in the project once it has been imported into the ISDE. You can open this file within the ISDE and follow the description provided to help identify key uses of APIs.

`AGT_HAL_MG.c` contains the prototypes of the functions which are used to configure the timers. `Hal_entry.c` calls the `init_timers()` function in `AGT_HAL_MG.c` that opens, starts, and configures the timers.
From `init_timers()`, the timers are opened with the `open_timers()` function. This function calls the `open` API for each timer. The `start_timers()` function calls the `start` API for each timer. Next, `init_timers()` does the following:

1. Changes the timer period of `g_timer_agt_1` to 2 seconds. The on-board LED1 thus toggles every 2 seconds. The `set_timer_period` function defined in `AGT_HAL_MG.c` can be used to achieve this. The function in turn calls the SSP API `periodSet`.
2. The counter values of all the timers are obtained and printed on the debug console. The `get_timer_value` function does this by calling the `counterGet` API.
3. Information from all the timers is obtained and printed on the debug console. The `get_timer_info` function does by calling the `infoGet` API.

The last section of `AGT_HAL_MG.c` is the user callback function. `agt_32b_callback` is called when `g_timer_agt_1` expires, and toggles LED1.

Note: This description assumes you are familiar with using printf() with the Debug Console in 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” given in the References section at the end of this document. Alternatively, the user can see results using the watch variables in the debug mode.

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

8. Customizing the AGT 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 AGT clock by updating the PCLKB in the Clock tab. The user can also change the AGT clock source in the respective AGT module properties tab.

The application project also provides functions to reset and close the timer. These functions can be used according to the user's application.

9. Running the AGT HAL Module Application Project

To run the AGT 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.

To implement the AGT HAL module application in a new project, follow the steps for defining, configuring, auto-generating files, adding code, compiling, and debugging on the target kit. Following these steps is a hands-on approach that can help make the development process with SSP more practical, while just reading over this guide tends to be more theoretical.

Note: The following steps are described in sufficient detail for someone experienced with the basic flow through the Synergy development process. If these steps are not familiar, refer to the *Getting Started with SSP* chapter in the *SSP User's Manual* listed in the References section at the end of this document.

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

1. Create a new Renesas Synergy project for the SK-S7G2 kit called AGT_HAL_MG_AP.
2. Add the AGT drivers to HAL/common and configure them with appropriate parameters as described in chapter 7.
3. Click on the **Generate Project Content** button.
4. Add the code from the supplied project file `AGT_HAL_MG.c`, `AGT_HAL_MG.h`, and `hal_entry.c`. (`hal_entry.c` can also be copied over the generated `hal_entry.c`).
5. Connect to the host PC through a micro USB cable to J19 on SK-S7G2.
6. Start to debug the application.
7. The output can be viewed in the Renesas Debug Virtual Console.

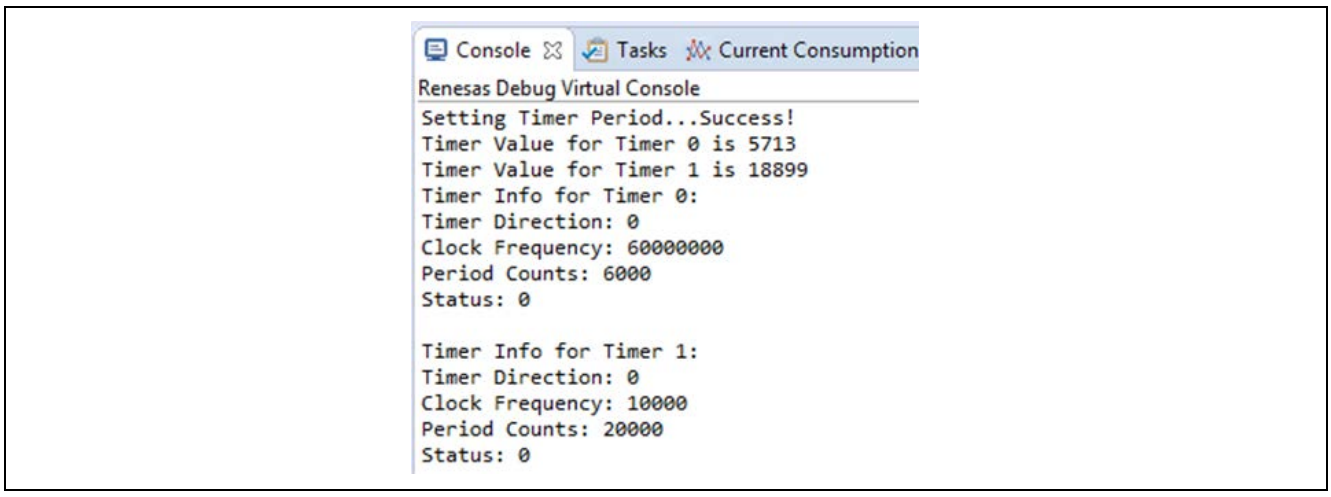

Figure 6 Example Output from Timer Drivers Application Project

The outputs from the AGT0 (P102) and AGT1 (P205) can also be viewed on an oscilloscope.

10. AGT 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) illustrate 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. AGT HAL Module Next Steps

After you have mastered a simple AGT HAL module project, you may want to review a more complex example. You can review the GPT HAL module guide if you require more instances or complex uses of timers.

You may also wish to explore using the AGT in various other applications; for example, the AGT Driver can also be used in a low power mode application to wake an application up.

12. AGT 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_agt` module reference materials and resources are available on the Synergy Knowledge Base: https://en-support.renesas.com/search/r_agt Module Guide Resources.

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 6, 2017		Initial Release
1.01	Dec 12, 2018		Updated for v1.5.0

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.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

RENESAS

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics Corporation
TOYOSU FORESIA, 3-2-24 Toyosu, Koto-ku, Tokyo 135-0061, Japan

Renesas Electronics America Inc.
1001 Murphy Ranch Road, Milpitas, CA 95035, U.S.A.
Tel: +1-408-432-8888, Fax: +1-408-434-5351

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-651-700

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709 Quantum Plaza, No.27 ZhichunLu, Haidian District, Beijing, 100191 P. R. China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, 200333 P. R. China
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HAL 2nd Stage, Indiranagar, Bangalore 560 038, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
17F, KAMCO Yangjae Tower, 262, Gangnam-daero, Gangnam-gu, Seoul, 06265 Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5338

Colophon 7.2