Renesas Synergy™ Platform

DMAC HAL Module Guide

R11AN0109EU0101
Rev.1.01
Aug 22, 2017

Introduction

This module guide enables 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) and should be valuable resources for creating more complex designs.

The Direct Memory Access Controller or DMAC HAL module is a high-level API for data-transfer applications and is implemented on `r_dmac`. The DMAC HAL module uses the DMAC peripheral in the Synergy MCU. A user-defined callback can be created to inform the CPU when transfer events occur.

Contents

1. DMAC HAL Module Features

The DMAC HAL module moves data from a user-specified source to a user-specified destination when an interrupt or event occurs. The DMAC HAL module supports the following:

- DMAC module on a Synergy MCU
- Interrupts, if desired
- Multiple transfer modes
  - Single Transfer
  - Repeat Transfer
  - Block Transfer
  - Address increment or fixed modes.
- Multiple channels, with the number depending on the MCU used.

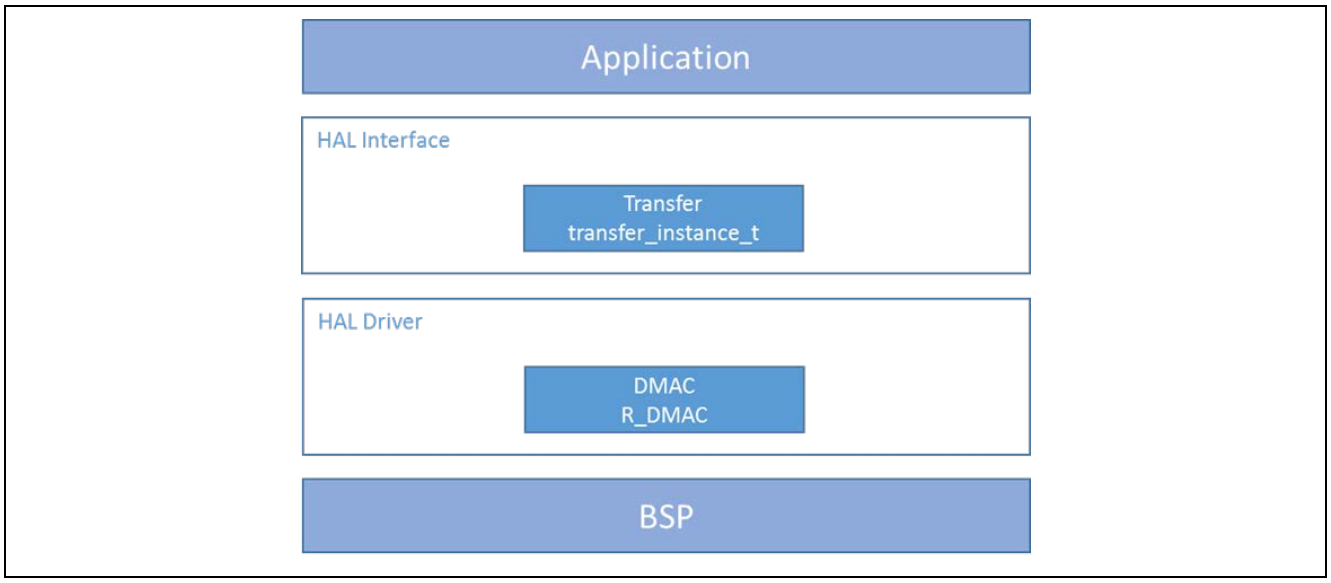


Figure 1 DMAC HAL Module Block Diagram

2. DMAC HAL Module APIs Overview

The DMAC HAL module defines APIs for opening, closing, starting, and stopping timers. The Data Transfer Controller (DTC) and the DMAC use the same transfer interface. Sharing an interface makes it easier to change between DTC and DMA implementations. The API calls are the same independent of the lower-level implementations. A complete list of the available APIs, an example API call, and a short description of each function can be found in the following API summary table. A table of status return values follows the API summary table.

Table 1 DMAC HAL Module API Summary

Function Name	Example API Call and Description
.open	`g_transfer0.api->open(g_transfer0.p_ctrl, g_transfer0.p_cfg)` Open device channel. Initialize driver and hardware on first call.
.close	`g_transfer0.api->close(g_transfer0.p_ctrl)` Close device channel. Turns off hardware if last channel is open.
.reset	`g_transfer0.api->reset(g_transfer0.p_ctrl, &source, &destination, number_of_transfers)` Reset channel settings.
.start	`g_transfer0.api->start(g_transfer0.p_ctrl, mode)` Start data transfer.
.stop	`g_transfer0.api->stop(g_transfer0.p_ctrl)` Stop data transfer.
.enable	`g_transfer0.api->enable(g_transfer0.p_ctrl)` Enable channel.

Function Name	Example API Call and Description
.disable	`g_transfer0.api->disable(g_transfer0.p_ctrl)` Disable channel.
.versionGet	`g_transfer0.api->versionGet(&version)` Retrieve the API version with the version pointer.
.infoGet	`g_transfer0.api->infoGet(g_transfer0.p_ctrl, &info)` Get transfer channel info.
.blockReset	`g_transfer0.api->blockReset(g_transfer0.p_ctrl, &source, &destination, length, size, number_of_transfers)` Reset Block Transfer parameters.

Note: Review the *SSP User's Manual* API References for the associated module for detailed descriptions of operation and definitions for the function data structures, typedefs, defines, API data, API structures and function variables.

Table 2 Status Return Values

Name	Description
SSP_SUCCESS	API Call Successful.
SSP_ERR_ASSERTION	Parameter has invalid value.
SSP_ERR_NOT_OPEN	Channel is not open.
SSP_ERR_UNSUPPORTED	Operation not configured correctly.
SSP_ERR_IN_USE	The channel specified has already been opened. No configurations were changed. Call the associated Close function or use associated Control commands to reconfigure the channel.
SSP_ERR_IRQ_BSP_DISABLED	IRQ not enabled in BSP.
SSP_ERR_NOT_ENABLED	Operation failed.
SSP_ERR_NOT_OPEN	The channel is not opened.

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. DMAC HAL Module Operational Overview

The DMAC HAL module moves data from a user-specified source to a user-specified destination when an interrupt or event occurs. The DMAC HAL module uses DMAC peripheral registers, so the number of transfers in the system is limited to the number of DMAC channels on the device. The activation source does not have to be enabled to use the DMAC. When the DMAC transfer completes, a DMAC interrupt is called. If the activation source interrupt is enabled, it fires at the same time the transfer is triggered. If the DMAC interrupt is enabled, it fires after all transfers are complete. For example, if a normal mode transfer with a length of 16 is triggered by a timer, the timer interrupt fires at the same time each transfer occurs and the DMAC interrupt fires after the 16th transfer completes. The DMAC does not support chained transfers.

3.1 DMAC HAL Module Operational Notes

Normal Mode

In Normal mode, a single transfer is triggered each time an activation source event occurs. A single transfer is 1 byte, 2 bytes, or 4 bytes, depending on the setting selected in the size parameter. Each time a transfer occurs, the transfer length decrements by 1. When the transfer length reaches 0, the transfer is complete.

Repeat Mode

In Repeat mode, a single transfer is triggered each time an activation source event occurs. A single transfer is 1 byte, 2 bytes, or 4 bytes, depending on the setting selected in the size parameter. Each time a transfer occurs, the transfer length decrements by 1. When transfer length reaches 0, the transfer length reloads with its initial value and the transfer restarts. If the repeat area is set to the source, the source register also reloads with its initial value when the transfer restarts. Alternatively, if the repeat area is set to the destination, the destination register reloads with its initial value when the transfer restarts.

Block Mode

In Block mode, the entire transfer length transfers each time an activation source event occurs. For example, if a transfer is configured in the Block mode with a timer as the activation source, a 2-byte size and 12-byte length, 24 bytes

transfer each time the activation source event occurs. Each time a transfer occurs, the transfer length decrements by 1. When the length reaches 0, the transfer length is reloaded with its initial value and the transfer restarts. If the repeat area is set to the source, the source register is also reloaded with its initial value when the transfer restarts. Alternatively, if the repeat area is set to the destination, the destination register reloads with its initial value when the transfer restarts.

Address Mode

After each transfer of size 1 byte, 2 bytes, or 4 bytes, the source pointer and destination pointer adjust by `src_addr_mode` and `dest_addr_mode`, respectively.

For example, if `src_addr_mode` is set to `TRANSFER_ADDR_MODE_INCREMENTED`, and size is set to `TRANSFER_SIZE_4_BYTES`, the `p_dest` pointer is incremented by 4 (the transfer size) after each transfer.

The pointer does not change if set to `TRANSFER_ADDR_MODE_FIXED.`

3.2 DMAC HAL Module Limitations

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

4. Including the DMAC HAL Module in an Application

This section describes how to include the DMAC 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 Starting Development section in the SSP User's Manual to learn how to manage each of these important steps in creating SSP-based applications.

To add the DMAC Driver to an application, simply add it to a thread using the stacks selection sequence provided in the following table. (The default name for the DMAC Driver is g_transfer0. This name can be changed in the associated Properties window.)

Table 3 GPT Selection Sequence

Resource	ISDE Tab	Stacks Selection Sequence
g_ transfer0 Transfer Driver on r_dmac	Threads	New Stack > Driver > Transfer> Transfer Driver on r_dmac

When the DMAC Driver on `r_dmac` is added to the thread stack as shown in the following figure, the configurator automatically adds any needed lower level drivers. Any drivers that need additional configuration information are highlighted in Red. Modules with a Gray band are individual modules that stand alone.

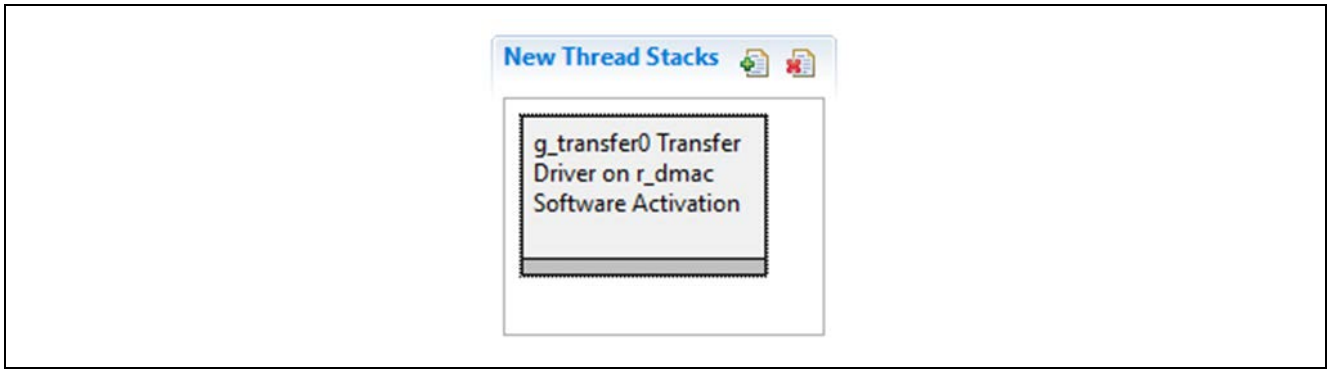


Figure 2 DMAC HAL Module Stack

5. Configuring the DMAC HAL Module

The DMAC 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. Also, only those properties that can be changed without causing conflicts are available for modification. Other properties are 'locked' and are not available for changes, and 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 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. Note that the interrupt priorities listed in the Properties window in the ISDE 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 following tables, but is easily visible within 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 settings table. This helps to 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 DMAC HAL Module on r_dmac

ISDE Property	Value	Description
Parameter Checking	Enabled, Disabled, BSP (Default: BSP)	Selects if code for parameter checking is to be included in the build
Name	g_transfer0	Module name
Channel	0	
Mode	Block	Mode selection
Transfer Size	1 Byte	Transfer size selection
Destination Address Mode	Fixed	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Software Activation	Activation source selection
Auto Enable	FALSE	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
Interrupt Priority	Priority 0 (highest), 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 (lowest, not valid if using Thread X), Disabled (Default: Disabled)	

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

5.1 DMAC HAL Module Clock Configuration

The DMAC peripheral module uses the ICLK as its clock source. The ICLK frequency is set by using the SSP configurator Clock tab, prior to a build, or by using the CGC Interface at run-time.

5.2 DMAC HAL Module Pin Configuration

The DMAC HAL module is not associated with any pins.

6. Using the DMAC HAL Module in an Application

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

1. Initialize the DMAC HAL module using the `open` API.
2. Enable the DMAC HAL module using the `enable` API (if not auto enabled).
3. Manage transfers using other APIs as needed.
4. Close the DMAC HAL module when needed.

The following operational flow diagram shows common steps in using the DMAC HAL module:

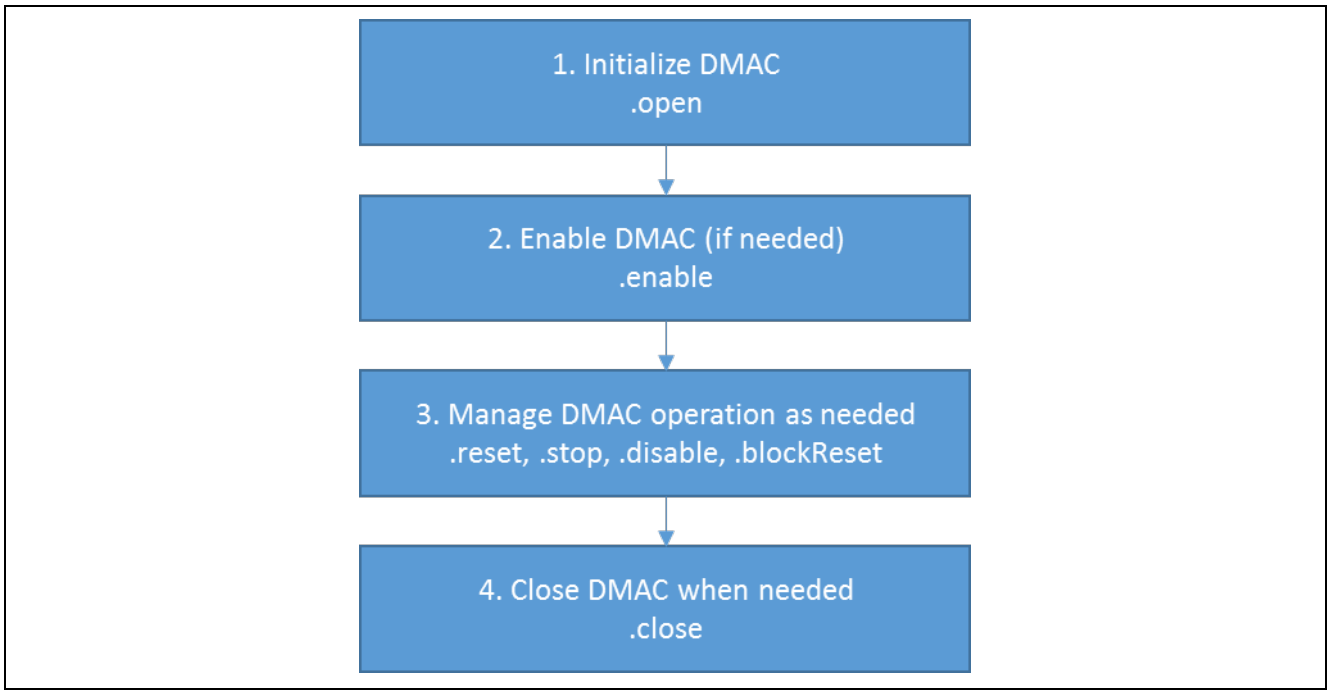


Figure 3 Flow Diagram of a Typical DMAC HAL Module Application

7. The DMAC HAL Module Application Project

The application project associated with this module guide illustrates the steps in an example application. You may want to import and open the application project within the ISDE and view the configuration settings for the DMAC HAL module. You can also read the code (`dmac_hal.c`) used to demonstrate the DMAC HAL APIs in a complete design.

The application project demonstrates the typical use of the DMAC HAL APIs and initializes the DMAC HAL module. The application project uses three transfer interfaces, created on DMAC channels 0, 1 and 3, as well as a mix of external IRQ pins and timers to generate data and trigger sources. Transfer Driver 0 is configured to transfer data from an array to a device I/O port. The chosen I/O port is PORT6, where the LEDs are connected. The transfer of data can be seen visually on the LEDs.

The activation source for Transfer Driver 0 is AGT0, one of the Asynchronous General Purpose Timers (AGT). The AGT is configured to generate a 250 ms interrupt. Transfer Driver 1 is configured to transfer data from a device peripheral register (GPT Count Register 0) to a software buffer register (`g_dest_data[]`). The activation source for Transfer Driver 1 is IRQ11, one of the external IRQ pins that generates when user pushbutton S4 is pressed.

Transfer Driver 2 is configured to transfer data to and from the same addresses as Transfer Driver 0. However, where Transfer Driver 0 was triggered by a peripheral, Transfer Driver 2 is triggered by software activation. The call to software activation is triggered when the user pushbutton, S5, is pressed. Pushbutton S5 generates IRQ10. The IRQ10 interrupt request (IRQ) calls the software activation function in the IRQ11 user callback function. If semi-hosting is enabled, the contents of `g_dest_data[]`output to the terminal. It is then possible to see the results of Transfer Driver 1.

Transfer Driver 0 is configured for Normal mode operation and a transfer count of 60. In this mode, the DMAC transfers data for the specified number of transfers and, unless reactivated, does not perform additional transfers. Transfer Driver 1 and Transfer Driver 2 are configured for Repeat mode operation and a transfer count of 8. In Repeat

mode, when the DMAC has completed the specified number of transfers (8 in this case), the DMAC automatically reconfigures itself and repeats the transfer operation. As a result, the DMAC is continuous.

Table 5 Software and Hardware Resources Used by the Application Project

Resource	Revision	Description
e² studio	5.3.1 or later	Integrated Solution Development Environment
SSP	1.2.0 or later	Synergy Software Platform
IAR EW for Synergy	7.71.2 or later	IAR Embedded Workbench® for Renesas Synergy™
SSC	5.3.1 or later	Synergy Standalone Configurator
SK-S7G2	v3.0 to v3.1	Starter Kit

The following figure shows a simple application project flow diagram.

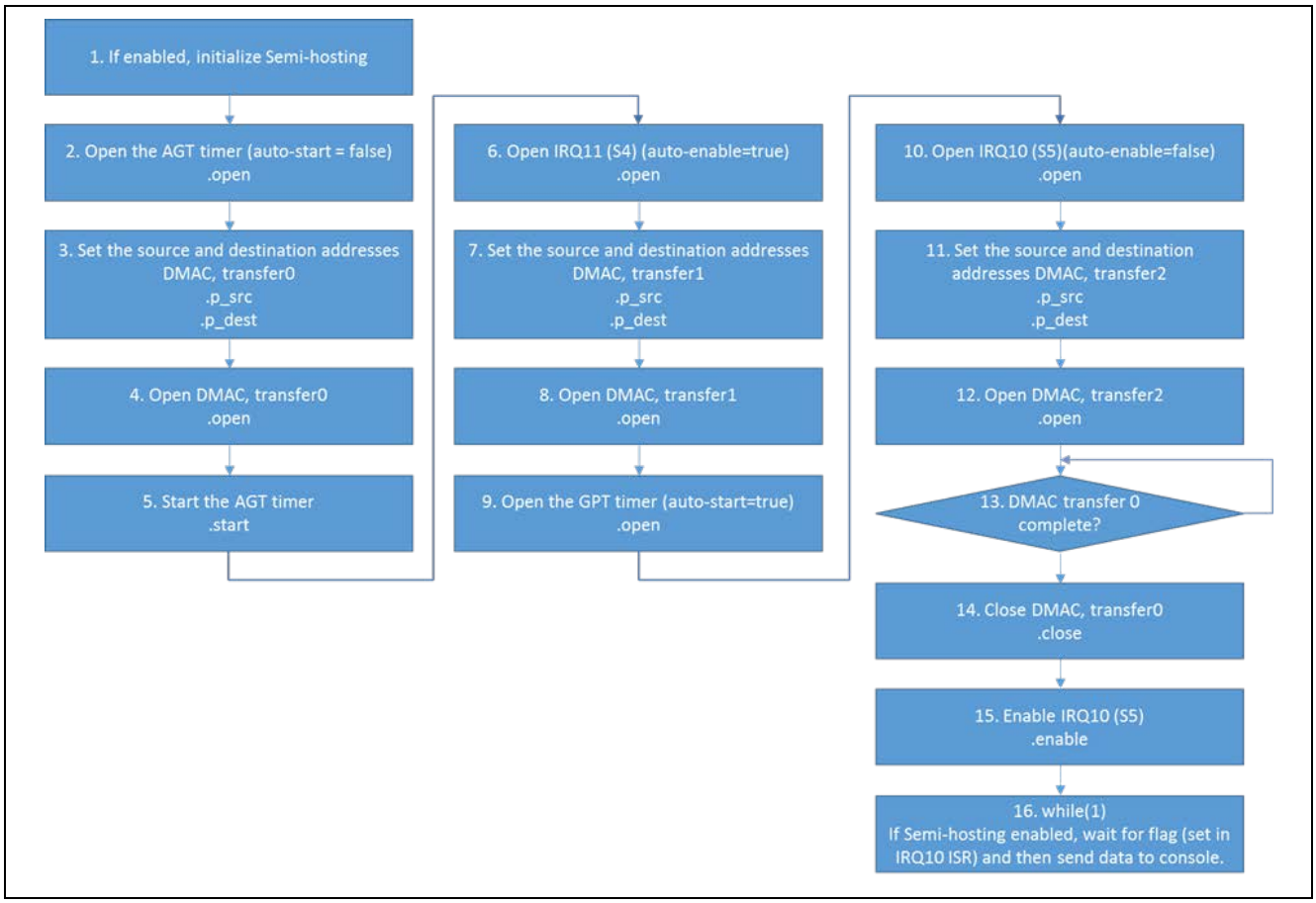


Figure 4 DMAC HAL Module Application Project Flow Diagram

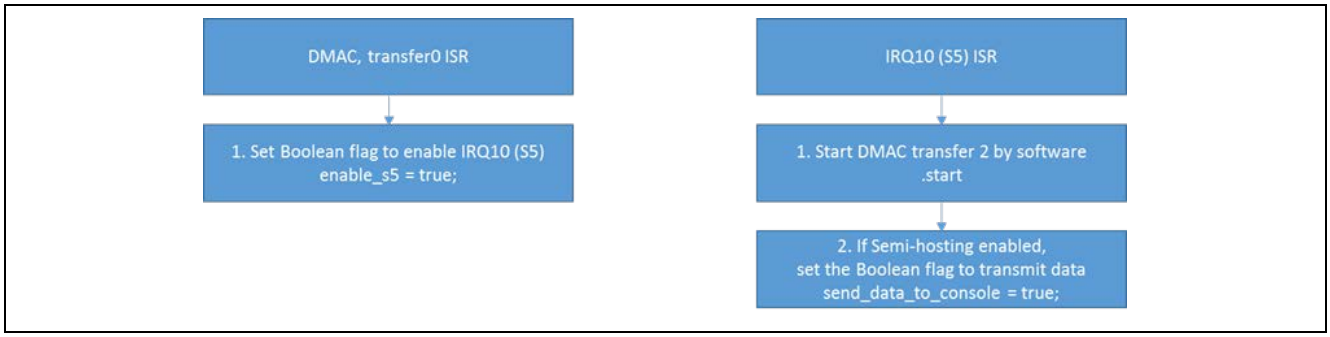


Figure 5 DMAC HAL Module Application Project ISR Flow Diagrams

The `dmac_hal.c` file is located in the project once it has been imported into the ISDE. You can open this file within the ISDE and use the descriptions provided to help identify the key uses of APIs.

The first section of `dmac_hal.c` details the header files. The first header file, `hal_data.h`, references the DMAC transfer and the AGT, as well as the General Purpose Time (GPT) and interrupt request (IRQ) instance structures. The second header file, `dmac_hal.h`, is where semi-hosting can be enabled or disabled by adding or removing the pre-processor command `#define SEMI_HOSTING`. The third header file is provided so that an editor can resolve references to device register locations. These register locations are used as the source and destination addresses for the DMAC transfers.

Global variable definitions that the project uses are described as follows, including a source and destination array, and some Boolean flags. The next section describes the entry function for the main program control section. If the function is enabled, and is using GCC, then semi-hosting is enabled, and this is done automatically using the IAR Embedded Workbench® for Renesas Synergy™. Next, the AGT, which is configured to generate a 1 second interrupt, is opened. As the configuration parameter, Auto Start is set to false. The AGT does not start counting when opened. The AGT interrupt triggers DMAC transfer 0.

Next, the DMAC transfer 0 is set up. The setup needs careful consideration. In the configurator, it is not possible to set a destination pointer and a source pointer, even though the fields are provided. The default is to leave these pointers set to NULL. However, if these pointers are set to NULL and the Auto Enable parameter is set to true, when the transfer is opened, the return code is not SSP_SUCCESS.

To resolve the pointer settings, use one of the following methods:

1. When the Destination Pointer and the Source Pointer are set to NULL, set the Auto Enable parameter to false. Open the transfer using the following API commands:

```
g_transfer0.p_api->open(g_transfer0.p_ctrl, g_transfer0.p_cfg);
```

Then, use the reset API to set the source and destination addresses:

```
g_transfer0.p_api->reset(g_transfer0.p_ctrl, SRC, DEST, COUNT);
```

2. An alternative method is to set the source and destination configuration parameters prior to using `open` API. For example:

```
g_transfer0.p_cfg->p_info->p_src = (void *) &g_source_data;
g_transfer0.p_cfg->p_info->p_dest = (void *) &R_IOPORT6->PCNTR1;


ssp_err = g_transfer0.p_api->open(g_transfer0.p_ctrl, g_transfer0.p_cfg);
if (SSP_SUCCESS != ssp_err)
{
    while (1)
    {
    }
}
```

This method allows for the Auto Enable parameter to be set to true.

In addition, the DMAC transfer 0 is set to transfer 32-bit values. In Normal mode, the source address is incremented, the destination address is fixed, the COUNT value is 60, and the DMAC interrupt is enabled with the callback defined as `cb_dmac0`.

With the AGT and DMAC transfer 0 configured, the next step is to start the AGT. The AGT generates a 1 second interrupt that triggers the DMAC. The DMAC transfers data from `g_source_data` to **R_IOPORT6->PCNTR1**. The source array contains a 32-bit value that sets and clears bits [2], [1], and [0] of the I/O Port Control Register 1. The Port Control Register 1 controls the I/O mode of the port pin; if set as an output, it sets the output value. With the LEDs connected to the port pins, the DMAC transfer alters the state of the LEDs.

Next, IRQ11 is opened. The IRQ11 interrupt occurs when user pushbutton S4 is pressed. Triggering the IRQ11 interrupt initiates DMAC transfer 1. Next, the DMAC transfer 1 is set up in the same way as DMAC transfer 0; the source address and destination address are set prior to `.open` function being called.

DMAC transfer 1 differs from DMAC transfer 0 in several ways, but the most significant difference is its configuration mode; DMAC transfer 1 is set to Repeat mode. DMAC transfer 0 transfers for its specified number of counts, such as 60 for example, and then stops. When DMAC transfer 1 operation transfers its specified number of counts, such as 8 for example, the DMAC automatically reconfigures itself and restarts. The DMAC is therefore continuous. DMAC transfer 1 is configured to transfer 32-bit data from a GPT counter register to the destination array `g_dest_data[]`. For a meaningful data transfer, the GPT timer has to be running. The GPT is opened after DMAC transfer 1 is configured. The GPT Auto Start parameter is set to true.

Next, IRQ10 is opened. IRQ10 is generated when user pushbutton S5 is pressed. IRQ10 interrupt triggers DMAC transfer 2, but by calling the Software Activation API. Next, the DMAC transfer 2 is setup in the same way as DMAC transfer 1: the source address and destination address are set prior to `.open` being called.

DMAC transfer 2 is again configured for Repeat mode, but differs from the other two transfers in that it is triggered by a software call, rather than through an interrupt activation. DMAC transfer 2 is configured to transfer 32-bit data from the source array **g_source_data to R_IOPORT6->PCNTR1**. As a result, pressing the user pushbutton S5 changes the state of the LEDs.

DMAC transfer 2 only occurs when DMAC transfer 0 has finished.

The transfer operation is controlled by IRQ10 enabling only when the DMAC transfer 0 end interrupt occurs and sets a Boolean flag indicating this state.

As DMAC transfer 0 has completed, the transfer is closed.

The last section of the application function has a `while(1)` loop. This loop waits for the Boolean flag to be set. When the flag is set, the application writes the contents of `g_dest_data[]` to the console. The contents of `g_dest_data[]` are the result of S4 and DMAC transfer 1.

The last section has the user callback functions. The first callback is DMAC transfer 0. This call is made when DMAC transfer 0 has performed 60 transfers, and has set a Boolean flag to notify the application code. The second callback is IRQ10 (through pressing S5) and it triggers DMAC transfer 2 through software.

Note: The application project assumes that you are familiar with using `printf()` and the debug console in the Synergy Software Package. If you are unfamiliar with `printf()` and the debug console, refer to the "How do I Use Printf() with the Debug Console in the Synergy Software Package" knowledge base article, that is available in the References section. Alternatively, the user can see results using the watch variables in Debug mode.

Key properties are configured in this application project to support required operations, as well as the physical properties of the target board and the MCU. The following tables list the property values set for this specific project. You may also open the application project and view these settings in the Properties window as a hands-on exercise.

Table 6 AGT HAL Module Configuration Settings for the Application Project

ISDE Property	Value Set
Name	g_agt0
Channel	0
Mode	Periodic
Period Value	250
Period Unit	Milliseconds
Auto Start	False
Count Source	LOCO

Table 7 DMAC transfer 0 HAL Module Configuration Settings for the Application Project

ISDE Property	Value Set
Name	g_transfer0
Channel	0
Mode	Normal
Transfer Size	4 Bytes
Destination Address Mode	Fixed
Source Address Mode	Incremented
Number of Transfers	60

ISDE Property	Value Set
Activation Source	Event AGT0 INT
Auto Enable	True
Callback	cb_dmac0
Interrupt Priority	Priority 8 (CM4: Valid, CM0+: invalid)

Table 8 IRQ11 HAL Module Configuration Settings for the Application Project

ISDE Property	Value Set
Name	g_external_irq11_s4
Channel	11
Trigger	Falling
Callback	NULL
Interrupt Priority	Priority 8 (CM4: Valid, CM0+: invalid)

Table 9 DMAC transfer 1 HAL Module Configuration Settings for the Application Project

ISDE Property	Value Set
Name	g_transfer1
Channel	1
Mode	Repeat
Transfer Size	4 Bytes
Destination Address Mode	Incremented
Source Address Mode	Fixed
Repeat Area	Destination
Number of Transfers	8
Activation Source	Event ICU IRQ11
Auto Enable	True

Table 10 GPT HAL Module Configuration Settings for the Application Project

ISDE Property	Value Set
Name	g_gpt0
Channel	0
Mode	Periodic
Period Value	0xffff
Period Unit	Raw Counts
Auto Start	True

Table 11 IRQ10 HAL Module Configuration Settings for the Application Project

ISDE Property	Value Set
Name	g_external_irq10_s5
Channel	10
Trigger	Falling
Interrupt enabled after initialization	False
Callback	Cb_irq10_s5
Interrupt Priority	Priority 8 (CM4: Valid, CM0+: invalid)

Table 12 DMAC transfer 2 HAL Module Configuration Settings for the Application Project

ISDE Property	Value Set
Name	g_transfer2
Channel	2
Mode	Repeat
Transfer Size	4 Bytes
Destination Address Mode	Fixed
Source Address Mode	Incremented
Repeat Area	Source
Number of Transfers	8
Activation Source	Software Activation
Auto Enable	True

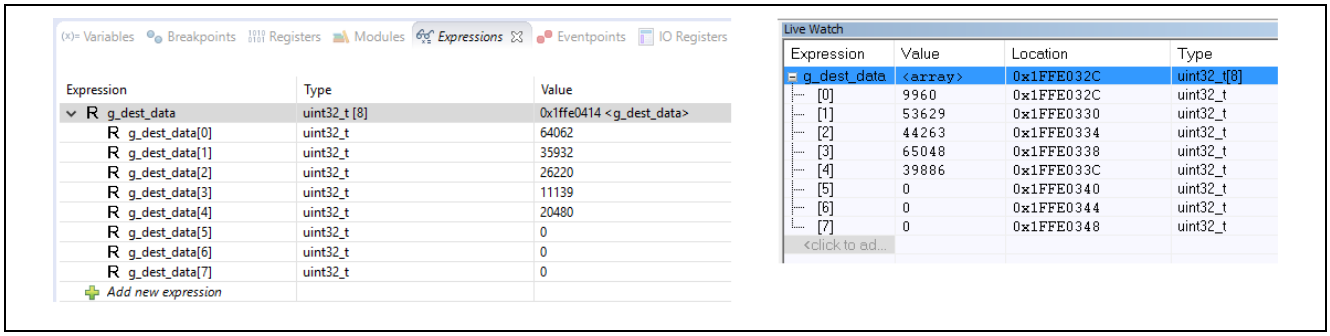
8. Customizing the DMAC HAL Module for a Target Application

The configuration settings provided for this application project are specific to this application example. The DMAC HAL module can be easily modified for your target application. For example, all data transfers are 32-bit transfers in this application example. The DMAC can be configured to transmit 8, 16, or 32-bit data. Also, this application example demonstrates Normal and Repeat modes, whereas the DMAC can also be configured to perform a block transfer on activation.

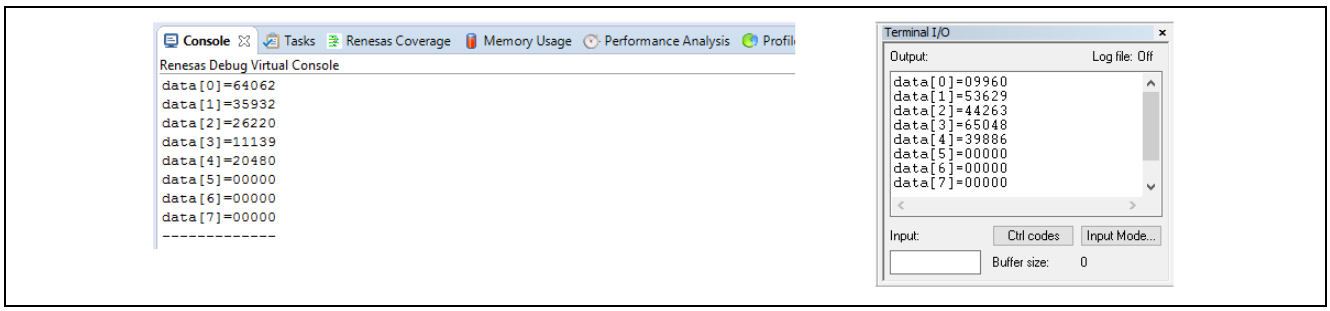
9. Running the DMAC HAL Module Application Project

To run the DMAC HAL module application project and see it execute on a target kit, you can simply import it into your ISDE, compile, and run debug. Refer to *Importing a Renesas Synergy Project* (11an0023eu0116-synergy-ssp-import-guide.pdf) included in this package, for instructions on importing the project into e^2 studio or the IAR EW for Synergy and then building, and running the application.

When running the project, the LEDs toggle for one minute. Pressing S4 transfers the GPT counter value to the array `g_dest_data[]`. As the figure shows, the array can be viewed in the Expressions Window (e^2 studio) on the left or the Watch Window (IAR EW for Synergy) on the right.



When the LEDs have stopped toggling, pressing the S5 pushbutton again changes the state of the LEDs. The array contents write to the console window in e^2 studio (left) or IAR EW for Synergy (right).



10. DMAC 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, avoiding the time required in older development environments to use, or, in some cases, create, lower-level drivers.

11. DMAC HAL Module Next Steps

After you have mastered a simple DMAC HAL module project, you may want to review a more complex example. Other application projects and application notes that demonstrate DMAC HAL use can be found as described in the References section.

12. DMAC 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_dmac` module reference materials and resources are available on the Synergy Knowledge Base: https://en-us.knowledgebase.renesas.com/English_Content/Renesas_Synergy%E2%84%A2_Platform/Renesas_Synergy_Knowledge_Base/r_dmac_Module_Guide_Resources.

Website and Support

Support: https://synergygallery.renesas.com/support

Technical Contact Details:

- America: https://renesas.zendesk.com/anonymous_requests/new
- Europe: https://www.renesas.com/en-eu/support/contact.html
- Japan: https://www.renesas.com/ja-jp/support/contact.html

Revision History

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	May 31, 2017		Initial version
1.01	Aug 22, 2017		Update to Hardware and Software Resources Table

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 disputes 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, drawing, chart, program, algorithm, 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 otherwise misappropriate 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, copy or otherwise misappropriation of Renesas Electronics products.
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; and 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.

   Renesas Electronics products are neither intended nor 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 damages (space and 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 third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. When using the 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 radiation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions or failure or accident arising out of the use of Renesas Electronics products beyond 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. Further, Renesas Electronics products are not subject to radiation resistance design. Please ensure to implement safety measures to guard them against the possibility of bodily injury, injury or damage caused by fire, and social damage in the event of 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 by your own responsibility as warranty for your products/system. Because the evaluation of microcomputer software alone is very difficult and not practical, please evaluate 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. Please investigate applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive carefully and sufficiently and use 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 not use Renesas Electronics products or technologies for (1) any purpose relating to the development, design, manufacture, use, stockpiling, etc., of weapons of mass destruction, such as nuclear weapons, chemical weapons, or biological weapons, or missiles (including unmanned aerial vehicles (UAVs)) for delivering such weapons, (2) any purpose relating to the development, design, manufacture, or use of conventional weapons, or (3) any other purpose of disturbing international peace and security, and you shall not sell, export, lease, transfer, or release Renesas Electronics products or technologies to any third party whether directly or indirectly with knowledge or reason to know that the third party or any other party will engage in the activities described above. When exporting, selling, transferring, etc., Renesas Electronics products or technologies, you shall comply with any applicable export control laws and regulations promulgated and administered by the governments of the countries asserting jurisdiction over the parties or transactions.
10. Please acknowledge and agree that you shall bear all the losses and damages which are incurred from the misuse or violation of the terms and conditions described in this document, including this notice, and hold Renesas Electronics harmless, if such misuse or violation results from your resale or making Renesas Electronics products available any third party.
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 majority-owned subsidiaries.

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

(Rev.3.0-1 November 2016)

SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

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

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

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: +44-1628-585-100, Fax: +44-1628-585-900

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, P. R. China 200333
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 II Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 6.0