Renesas Synergy™ Platform

PDC 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 Parallel Data Capture Unit (PDC) HAL module is a high-level API to capture images from a camera application and is implemented on `r_pdc`. The PDC HAL module uses the PDC peripherals on the Synergy MCU. A user-defined callback can be created to inform the CPU when a capture has been completed.

Contents

1. PDC HAL Module Features

This PDC HAL module controls the PDC on a Synergy microcontroller and has the following key features:

- Supports capture from a connected and configured camera
- Supports a callback that informs the CPU when a capture is complete
- Provides a pointer to the capture buffer
- Provides an indication of the event triggering the callback.

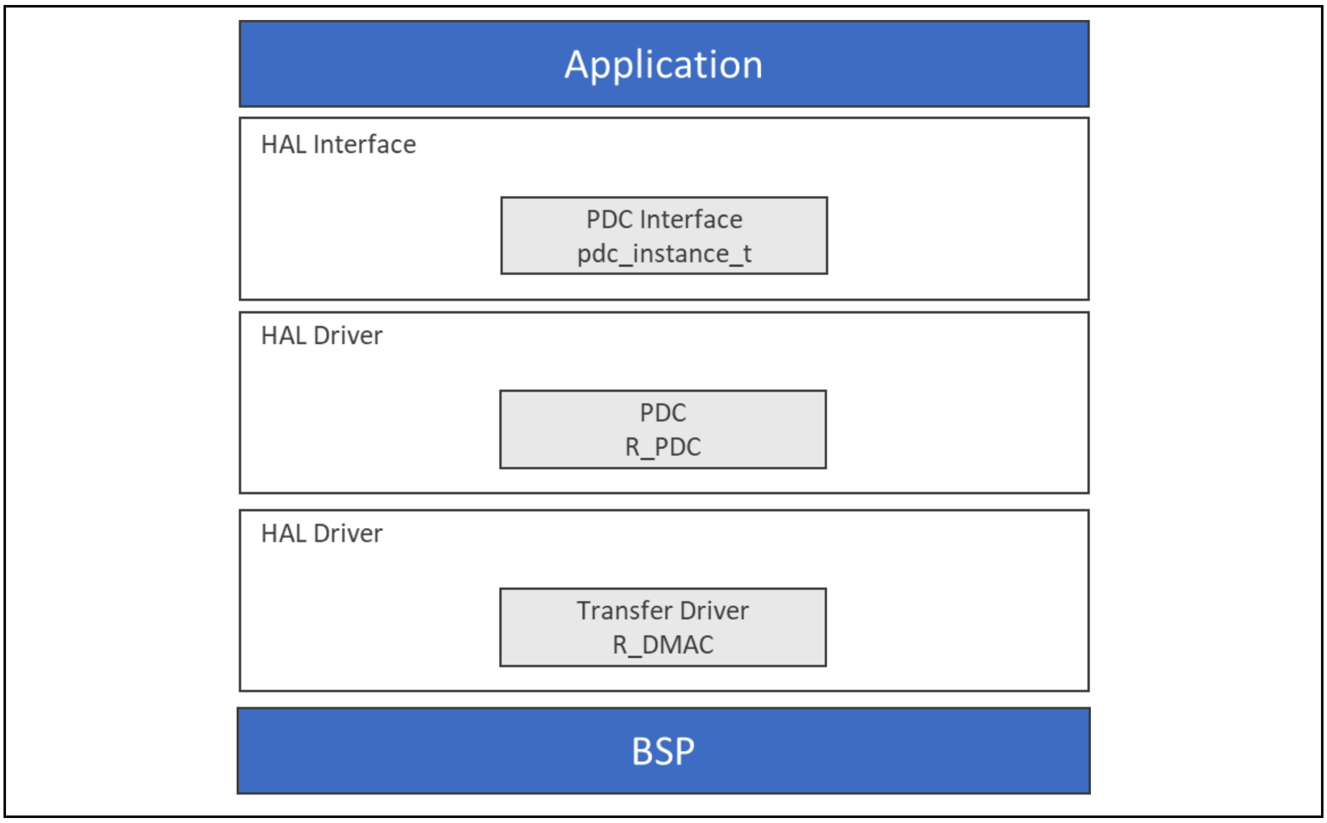

Figure 1. PDC HAL Module Block Diagram

2. PDC HAL Module APIs Overview

The PDC HAL module defines APIs for opening, closing, and starting data capture. 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. PDC HAL Module API Summary

Function Name	Example API Call and Description
.open	`g_pdc.p_api->open(g_pdc.p_ctrl, g_pdc.p_cfg)` Initial configuration.
.close	`g_pdc.p_api->close(g_pdc.p_ctrl))` Closes the driver and allows reconfiguration. May reduce power consumption.
.captureStart	`g_pdc.p_api->captureStart(g_pdc.p_ctrl, p_buffer)` Start a capture.
.stateGet	`g_pdc.p_api->stateGet(g_pdc.p_ctrl, &state_data)` Get the state of VSYNC and HSYNC pins.
.versionGet	`g_pdc.p_api->versionGet(&version)` Return the API version using the version pointer.

Note: Review the *SSP User's Manual* API References for the associated module to learn more about operations 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	The parameter p_ctrl or p_sample is NULL.
SSP_ERR_INVALID_ARGUMENT	Parameter has invalid value.
SSP_ERR_NOT_OPEN	Unit is not open.
SSP_ERR_HW_LOCKED	Unable to reserve BSP hardware lock.
SSP_ERR_TIMEOUT	Reset Operation timed out.

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. PDC HAL Module Operational Overview

The capture operation requires a configured external camera connected to the Synergy microcontroller. Before performing a capture, it is important that the camera is configured and is generating a PIXCLK-clock input into the microcontroller. In some instances, a camera requires a running-clock input before it can be configured.

Use the api call `open`, which configures and starts the PCKO-clock output from the PDC into the camera, before initializing the camera. Once the camera is configured, `captureStart` can be called to capture an image. Configuration of a camera module may require the use of an I^2C or SPI interface.

3.1 PDC HAL Module Operational Notes

In most instances, the data rate from a camera and from the PDC peripheral is too fast to be serviced by the CPU in an interrupt service routine (ISR). Therefore, this module requires an implementation of the transfer driver on the DMAC to perform a high-speed transfer from the PDC peripheral and memory.

In the following situations, you must enable both PDC frame-end and PDC error interrupts to generate:

- When an image is captured (frame end).
- When an error occurs.

Data Buffer Setting

If p_buffer is set to anything other than NULL, one or more data buffers are created to store image data. The size of each buffer is calculated using the following formula:

Buffer size (bytes) = x_capture_pixels x y_capture_pixels x bytes_per_pixel

For large resolution cameras, the captured image could result in a large amount of data. It may be necessary to locate the buffer(s) in external memory (such as, SDRAM). Consideration should be given to bus bandwidth when using external memory.

For example, when using a high frame-rate camera to perform an image capture through the PDC into the SDRAM and using the SDRAM to hold the display buffer for an LCD display with a high refresh rate, may cause a data bottleneck from the PDC to memory that results in an overrun-error condition.

Note: If p_buffer is set to NULL, no memory is allocated to store the captured image data. It is your responsibility to ensure that there is suitable memory of sufficient size available to the PDC. The PDC could capture directly into the display buffer of a connected LCD panel.

3.2 PDC HAL Module Limitations

For the PDC HAL module operational limitations, refer to the latest *SSP Release Notes*.

4. Including the PDC HAL Module in an Application

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

Note: It's assumed that you are familiar with creating a project, adding threads, adding a stack to a thread, and configuring a block within the stack. If you are unfamiliar with any of these items, 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 PDC 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 PDC is `r_pdc0`. This name can be changed in the associated Properties window.)

Table 3. CRC Selection Sequence

Resource	ISDE Tab	Stacks Selection Sequence
r_pdc0 PDCDriver on r_pdc	Threads	New Stack> Driver> Graphics> PDC Driver on r_pdc

When the PDC HAL module on `r_pdc` is added to the thread stack as shown in the following figure, the configurator automatically adds the needed lower-level drivers. Any drivers needing additional configuration information are highlighted in Red. Modules with a Gray band are individual standalone modules.

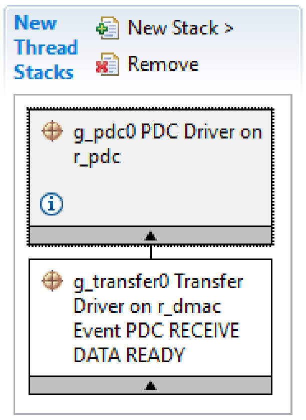
New Thread Stacks
New Stack >
Remove
g_pdc0 PDC Driver on r_pdc
g_transfer0 Transfer Driver on r_dmac Event PDC RECEIVE DATA READY

Figure 2. PDC HAL Module Stack

5. Configuring the PDC HAL Module

The PDC HAL module is 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, configured for lower-level modules to enable successful operation. Only properties that can be changed without causing conflicts, are available for modification. Properties that cannot be changed are 'locked' and identified with a lock icon to indicate the 'locked' property in the Properties window in the ISDE. This approach simplifies the configuration process, making 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 and 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 in the ISDE includes 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 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 table settings. This helps 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 PDC HAL Module on r_pdc

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	Enable or disable the parameter error checking.
Name	g_pdc0	The name of the PDC module instance. Specify arbitrary C symbol.
Name of the data buffer to store image data	g_user_buffer	Specify the name of the data buffer to create or set to NULL, if it is to be created by the user external to the PDC driver.
Section where data buffer is allocated	sdram	Specify the RAM section for the image data buffer. Typically, BSS (internal RAM) or SDRAM.
Number of bytes per pixel	2	Specify the number of bytes per pixel of the captured image data.
Number of image data buffers	1	Specify the number of buffers to create.
Clock Divider	CLK/2, CLK/4, CLK/6, CLK/8, CLK10, CLK12, CLK14, CLK16 (Default: CLK/2)	Specify the clock divider of the clock input to the PDC peripheral.
Endian of image data	Little, Big (Default: Little)	Specify the endian of the captured image data.
HYSNC signal polarity	High, Low (Default: High)	Specify the active polarity of the HSYNC signal.
VSYNC signal polarity	High, Low (Default: High)	Specify the active polarity of the VSYNC signal.
Number of pixels to capture horizontally	640	Number of horizontal pixels to capture.
Number of lines to capture vertically	480	Number of vertical lines to capture.
Horizontal pixel to start capture from	0	Horizontal pixel to start capturing image data from. Allows an image smaller than the native resolution of a camera to be captured.
Line to start capture from	0	Vertical line to start capturing image data from. Allows an image smaller than the native resolution of a camera to be captured.
Callback	g_pdc_user_callback	A user callback function can be registered in open. If this callback function is provided, it is called from the interrupt service routine (ISR) each time a frame is captured and ready to be processed. 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.

ISDE Property	Value	Description
Frame End Interrupt Priority	Priority 0 (highest), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) (Default: Disabled)	The driver needs a valid interrupt priority setting. It will not function if disabled.
PDC 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)	The driver needs a valid interrupt priority setting. It will not function if disabled.

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

In some cases, settings other than the default values for a module can be desirable. For example, it might be useful to select a different clock source than the default. The configurable properties for the lower-level stack modules are given in the following sections for completeness and as a reference.

Typically, only a small number of settings must be modified from the default for lower-level drivers as indicated with the red text in the thread stack block. Notice that some of the configuration properties must be set to a certain value for proper framework operation and are locked to prevent user modification. The following table identifies all the settings within the properties section for the module.

Table 5. Configuration Settings for the DMAC HAL Module

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	Parameter selection.
Name	g_transfer0	Driver name.
Mode	Block	Mode selection.
Transfer Size	4 Bytes	Transfer size selection.
Destination Address Mode	Incremented	Destination address mode selection.
Source Address Mode	Fixed	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	8	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	1	Number of blocks selection.
Activation Source (Must enable IRQ)	Event PDC RECEIVE DATA READY	Activation source selection.
Auto Enable	FALSE	Auto enable selection.
Callback (Only valid with Software start)	NULL	Callback selection.

ISDE Property	Value	Description
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)	Interrupt priority selection

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

5.1 PDC HAL Module Clock Configuration

The PDC uses PCLKB as its clock source. The only restriction when configuring this clock is that the PIXCLCK should be less than 0.6 x PCLKB, so the PCLKB frequency should be set accordingly.

5.2 PDC HAL Module Pin Configuration

The PDC 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 depicts the method to select pins within the SSP configuration window. The subsequent table gives an example selection for PDC pins.

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

Table 6. Pin Selection Sequence for the PDC HAL Module

Resource	ISDE Tab	Pin selection Sequence
PDC	Pins	Select **Peripherals > Graphics: PDC > PDC0**

Note: The selection sequence assumes PDC0 is the desired hardware target for the driver.

Table 7. Pin Configuration Settings for the PDC HAL Module

Property	Value	Description
Pin Group Selection	Mixed, _A Only (Default: Mixed)	Pin group selection
Operation Mode	Disabled, Custom, Enabled (Default: Disabled)	Select Enabled as the Operation Mode for PDC
HSYNC	None, P704 (Default: None)	HSYNC pin
PCKO	None, P511 (Default: P511)	PCKO pin
PIXCLK	None, P705 (Default: None)	PIXCLK pin
VSYNC	None, P512 (Default: P512)	VSNC pin
PIXD0:7	None, Pnnn (Default: None)	PIX Data0:7 pin

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

6. Using the PDC HAL Module in an Application

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

1. Initialize the PDC HAL module using the `open` API.
2. Configure the camera as needed.
3. Start image capture using the `captureStart` API.
4. Callback is called when image is captured.
5. Read state of HSYNC and VSTNC using `stateGet` API.
6. Process data as needed.
7. Close using the `close` API.

The following diagram shows common steps in a typical operational flow.

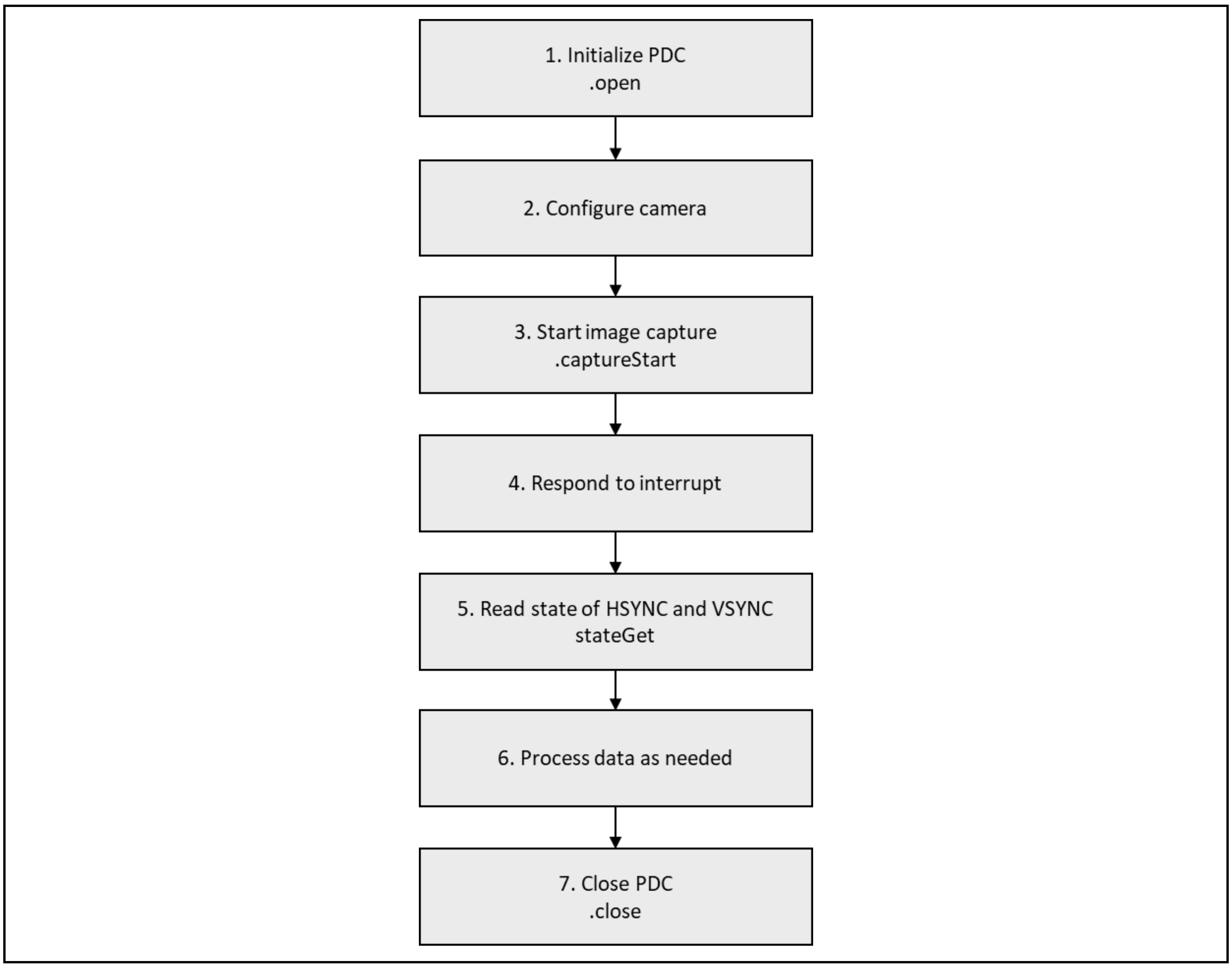

Figure 3. Flow Diagram of a Typical PDC HAL Module Application

7. The PDC HAL Module Application Project

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

The application project demonstrates the typical use of the PDC APIs. In the main thread entry, the PDC is initialized and the camera and display can be set up. The capture of an image from the camera is then started and the result is displayed on the screen. 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
DK-S7G2	v3.0 to v3.1	Developers Kit

The following diagram shows a simple application project flow:

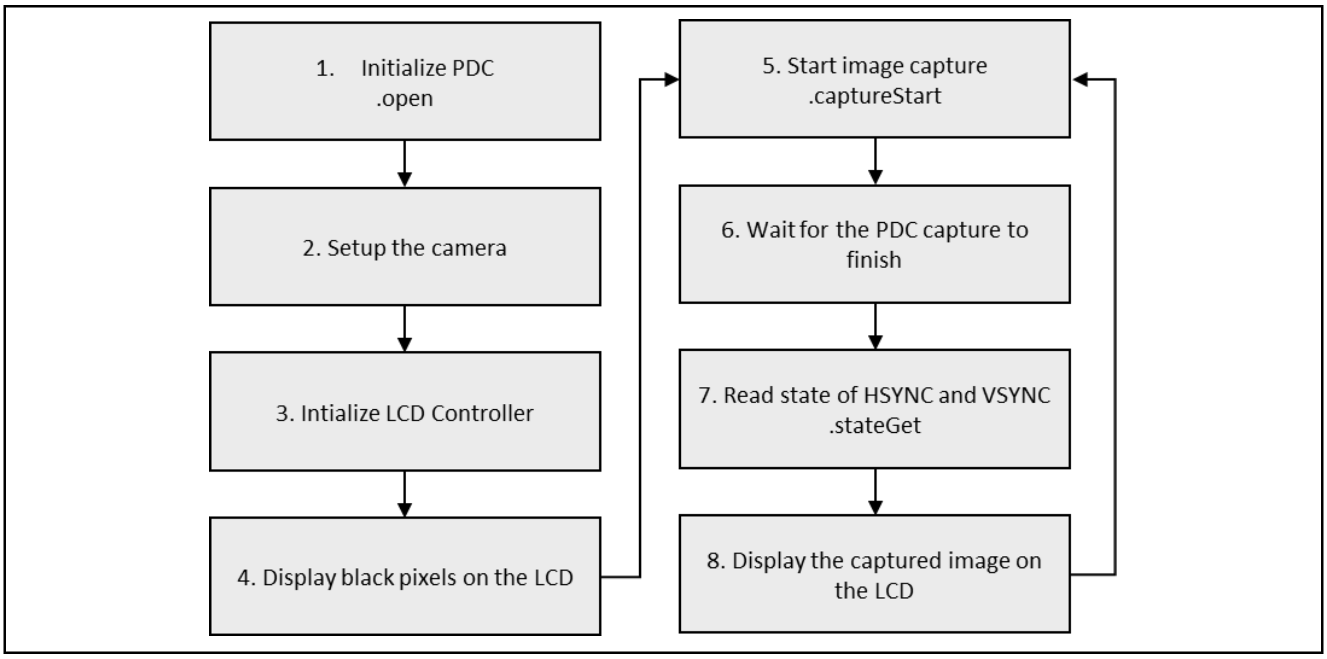
1. Initialize PDC
.open
2. Setup the camera
3. Intialize LCD Controller
4. Display black pixels on the LCD
5. Start image capture
.captureStart
6. Wait for the PDC capture to finish
7. Read state of HSYNC and VSYNC
.stateGet
8. Display the captured image on the LCD

Figure 4. PDC HAL Module Application Project Flow Diagram

The complete application project can be found using the link provided in the Reference section at the end of this document. The `pdc_thread_entry.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 along to help identify key uses of APIs.

The first section of `pdc_thread_enty.c` has the header field which references the PDC instance structure and the camera (OV7670) configuration structure. The next section has the entry function for the main program control. The PDC HAL module is initialized using the `open` API, and the camera is configured using the I^2C protocol. The `open` API can open the display.

The pins for turning on the display and the backlight are set to the high level. The display is ready to start, so after the `start` API call, the first image containing all black pixels displays. Inside the 'forever' loop, the image capture is started using the `captureStart` API. When the transfer is complete, the PDC semaphore counter is decremented and the image can be displayed on the LCD. To show how to use the `stateGet` API, the state of the HSYNC and VSYNC signals are read after the capture. The application project uses double-buffering, so in every iteration the number of the buffer is flipped between 0 and 1. After each step, the returned status is checked (in case of non-zero code the interior infinite while loop is executed.)

The `application.c` file has the PDC user-callback function. It checks whether the transfer is completed and increases the PDC semaphore counter. The I^2C callback function section follows and tries to increase I^2C semaphore counter. If the operation is not successfully completed, the I^2C error flag is set. Next the I^2C event type is checked and if the communication has been aborted, the relevant flag is set as well. After the event type is checked, in next section the I^2C helper function waits for a transmission to finish, including any error checks. The last section has the LCD helper function used to display the contents in a buffer.

A few key properties are configured in this application project to support required operations, as well as the physical properties of the target board and MCU.

The following table lists properties with 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 9. PDC Configuration Settings for the Application Project

ISDE Property	Value Set
Name	g_pdc
Name of the data buffer to store image data	g_user_buffer
Section where data buffer is allocated	sdram
Number of bytes per pixel	2
Number of image data buffers	1
Clock divider	CLK/6
Endian of image data	Big
HSYNC signal polarity	High
VSYNC signal polarity	Low
Number of pixels to capture horizontally	480
Number of lines to capture vertically	272
Horizontal pixel to start capture from	80
Line to start capture from	104
Callback	g_pdc_user_callback
Frame End Interrupt Priority	Priority 8 (CM4: valid, CM0+: invalid)
PDC Interrupt Priority	Priority 8 (CM4: valid, CM0+: invalid)

Note: Apply pin configuration settings from top to bottom. When a peripheral is enabled, some pins configure automatically, so changing their options manually is not needed.

Table 10. Application Project Configuration Settings for the DMAC HAL Module on r_dmac PDC RECEIVE DATA READY

ISDE Property	Value Set
Name	g_pdc_transfer
Channel	0
Mode	Block
Transfer Size	4 Bytes
Destination Address Mode	Incremented
Source Address Mode	Fixed
Repeat Area (Unused in Normal Mode)	Source
Destination Pointer	NULL
Source Pointer	NULL
Number of Transfers	8
Number of Blocks (Valid only in Block Mode)	1
Activation Source	Event PDC RECEIVE DATA READY
Callback	NULL
Interrupt Priority	Priority 8 (CM4: valid, CM0+: invalid)

Note: Apply pin configuration settings from top to bottom. When a peripheral is enabled, some pins configure automatically, so changing their options manually is not needed.

Table 11. Application Project Configuration Settings for the I^{2}C HAL Module on r_sci_i2c

ISDE Property	Value Set
Name	g_i2c7
Channel	7
Rate	Standard
Slave address	0x21
Address Mode	7-Bit
SDA Output Delay (nano seconds)	0

ISDE Property	Value Set
Bit Rate Modulation Enable	Disable
Callback	i2c_7_callback
Receive Interrupt Priority	Priority 8 (CM4: valid, CM0+: invalid)
Transmit Interrupt Priority	Priority 8 (CM4: valid, CM0+: invalid)
Transmit End Interrupt Priority	Priority 8 (CM4: valid, CM0+: invalid)

Note: Apply pin configuration settings from top to bottom. When a peripheral is enabled, some pins configure automatically, so changing their options manually is not needed.

Table 12. Application Project Configuration Settings for the DTC HAL Module on r_dtc Event SCI7 TXI

ISDE Property	Value Set
Parameter Checking	Default (BSP)
Software Start	Disabled
Linker section to keep DTC vector table	.ssp_dtc_vector_table
Name	g_i2c7_tx_transfer
Mode	Normal
Transfer Size	1 Byte
Destination Address Mode	Fixed
Source Address Mode	Incremented
Repeat Area (Unused in Normal Mode)	Source
Interrupt Frequency	After all transfers have completed
Destination Pointer	NULL
Source Pointer	NULL
Number of Transfers	0
Number of Blocks (Valid only in Block Mode)	0
Activation Source (Must enable IRQ)	Event SCI7 TXI
Auto Enable	False
Callback (Only valid with Software reset)	NULL
ELC Software Event Interrupt Priority	Disabled

Note: Apply pin configuration settings from top to bottom. When a peripheral is enabled, some pins configure automatically, so changing their options manually is not needed.

Table 13. Application Project Configuration Settings for the DTC HAL Module on r_dtc Event SCI7 RXI

ISDE Property	Value Set
Parameter Checking	Default (BSP)
Software Start	Disabled
Linker section to keep DTC vector table	.ssp_dtc_vector_table
Name	g_i2c7_rx_transfer
Mode	Normal
Transfer Size	1 Byte
Destination Address Mode	Incremented
Source Address Mode	Fixed
Repeat Area (Unused in Normal Mode)	Destination
Interrupt Frequency	After all transfers have completed
Destination Pointer	NULL
Source Pointer	NULL
Number of Transfers	0
Number of Blocks (Valid only in Block Mode)	0
Activation Source (Must enable IRQ)	Event SCI7 RXI
Auto Enable	False
Callback (Only valid with Software reset)	NULL

ISDE Property	Value Set
ELC Software Event Interrupt Priority	Disabled

Note: Apply pin configuration settings from top to bottom. When a peripheral is enabled, some pins configure automatically, so changing their options manually is not needed.

Table 14. Application Project Configuration Settings for the GLCD HAL Module on r_glcd

ISDE Property	Value Set
Name	g_display
Name of display callback function to be defined by user	NULL
Input - Panel clock source select	Internal clock(GLCDCLK)
Input - Graphics screen1	Used
Input - Graphics screen1 frame buffer name	fb_background
Input - Number of Graphics screen1 frame buffer	2
Input - Section where Graphics screen1 frame buffer allocated	bss
Input - Graphics screen1 input horizontal size	480
Input - Graphics screen1 input vertical size	272
Input - Graphics screen1 horizontal stride (not bytes but pixels)	480
Input - Graphics screen1 input format	16bits RGB565
Input - Graphics screen1 input line descending	Not used
Input - Graphics screen1 input lines repeat	Off
Input - Graphics screen1 input lines repeat times	0
Input - Graphics screen1 layer coordinate X	0
Input - Graphics screen1 layer coordinate Y	0
Input - Graphics screen1 layer background color alpha	255
Input - Graphics screen1 layer background color Red	255
Input - Graphics screen1 layer background color Green	255
Input - Graphics screen1 layer background color Blue	255
Input - Graphics screen1 layer fading control	None
Input - Graphics screen1 layer fade speed	0
Input - Graphics screen2[1]	Not used
Output - Horizontal total cycles	582
Output - Horizontal active video cycles	480
Output - Horizontal back porch cycles	41
Output - Horizontal sync signal cycles	41
Output - Horizontal sync signal polarity	Low active
Output - Vertical total lines	286
Output - Vertical active video lines	272
Output - Vertical back porch lines	3
Output - Vertical sync signal lines	10
Output - Vertical sync signal polarity	Low active
Output - Format	16bits RGB565
Output - Endian	Little endian
Output - Color order	RGB
Output - Data Enable Signal Polarity	High active
Output - Sync edge	Rising edge
Output - Background color alpha channel	255
Output - Background color R channel	0

[1] All *Input – Graphics screen2* related settings can be set to their default values.

ISDE Property	Value Set
Output - Background color G channel	0
Output - Background color B channel	0
CLUT	Used
CLUT - CLUT buffer size	256
TCON - Hsync pin select	LCD_TCON1
TCON - Vsync pin select	LCD_TCON2
TCON - DataEnable pin select	LCD_TCON0
TCON - Panel clock division ratio	1/24
Color correction - Brightness	Off
Color correction - Contrast	Off
Color correction - Gamma correction(Red)	Off
Color correction - Gamma correction(Green)	Off
Color correction - Gamma correction(Blue)[2]	Off
Dithering[3]	Off
Misc - Correction Process Order	Brightness and Contrast then Gamma
Line Detect Interrupt Priority	Disabled
Underflow 1 Interrupt Priority	Disabled
Underflow 2 Interrupt Priority	Disabled

Note: Apply pin configuration settings from top to bottom. When a peripheral is enabled, some pins configure automatically, so changing their options manually is not needed.

Table 15. Pin Configuration for the Application Project

Pin Selection Sequence	Pin Configuration Property	Setting
Peripherals > System:DEBUG > DEBUG0	Operation Mode	JTAG
Peripherals > System:BUS > BUS0	RD	P600
Peripherals > System:BUS > BUS0	WR1_BC1	P202
Peripherals > System:BUS > BUS0	WAIT	P206
Peripherals > System:CGC > CGC0	Operation Mode	Disabled
Peripherals > Storage:SDHI > SDHI0	Operation Mode	Disabled
Peripherals > Storage:QSPI > QSPI0	Operation Mode	Disabled
Peripherals > Connectivity:ETHERC > ETHERC1.RMII	Operation Mode	Disabled
Peripherals > Connectivity:USBFS > USBFS0	Operation Mode	Disabled
Peripherals > Connectivity:USBHS > USBHS0	Operation Mode	Disabled
Peripherals > Connectivity:SCI > SCI8	Operation Mode	Disabled
Peripherals > Connectivity:SCI > SCI7	Pin Group Selection	_B only
Peripherals > Connectivity:SCI > SCI7	Operation Mode	Simple I^{2}C
Peripherals > Graphics:PDC > PDC0	Operation Mode	Enabled
Peripherals > Graphics:GLCDC > GLCDC0	Pin Group Selection	_B only
Peripherals > Graphics:GLCDC > GLCDC0	Operation Mode	RGB 565
Peripherals > Graphics:GLCDC > GLCDC0	LCD_TCON0	P315
Ports > P7 > P710	Symbolic Name	LCD_DISPLAY_ON
Ports > P7 > P710	Mode	Output mode (Initial Low)
Ports > P7 > P712	Symbolic Name	LCD_LIGHT_ON

[2] All numeric *Color correction* related settings can be set to their default values

[3] All *Dithering* related settings can be set to their default values

Pin Selection Sequence	Pin Configuration Property	Setting
Ports > P7 > P712	Mode	Output mode (Initial Low)
Ports > PB > PB06	Symbolic Name	CAMERA_RESET
Ports > PB > PB06	Mode	Output mode (Initial High)
Ports > PB > PB07	Symbolic Name	CAMERA_PWDWN
Ports > PB > PB07	Mode	Output mode (Initial Low)

Note: Apply pin configuration settings from top to bottom. When a peripheral is enabled, some pins configure automatically, so changing their options manually is not needed.

8. Customizing the PDC 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 PDC clock by updating PCLKB in the Clocks tab or the clock divider in the PDC HAL module on stack in the Threads tab. The configuration settings regarding endian of image data or image resolution can also be changed in the PDC HAL module on stack.

9. Running the PDC HAL Module Application Project

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

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

Note: The following steps are in sufficient detail for someone experienced with the basic flow through the Synergy development process. If these steps are not familiar, refer to the first few chapters of the SSP User's Manual, listed in the References section at the end of this document.
For details on how to set up dip switches and connect the cameral module, see the PDC Web Camera Getting Started Guide (r12an0046eu).
If you see the linker error *Error[Li005]: no definition for "__StackLimit" [referenced from tx_thread_schedule.o(libtx.a)]* when building IAR EW project, exclude the libtx.a file in \synergy\ssp\src\framework\el\tx\cm4_gcc folder.

To create and run the PDC application project, use the following steps:

1. Create a new Renesas Synergy project for the S7G2 DK called **PDC_HAL_MG_AP**.
2. Select the **Threads** tab.
3. Add a new thread.
   Symbol: pdc_thread
   Name: PDC Thread
4. Add a new semaphore in the PDC Thread.
   Symbol: g_pdc_semaphore
   Name: PDC Semaphore
5. Add a new semaphore in the PDC Thread.
   Symbol: g_i2c7_semaphore
   Name: I^{2}C Channel 7 Semaphore
6. Add the PDC HAL module to the PDC Thread.
7. Add the I^{2}C HAL module to the PDC Thread.
8. Add the display driver to the PDC Thread.
9. Rename the DTC HAL Event **SCI7 TXI block g_i2c7_tx_transfer**.
10. Configure block according to the preceding tables.
11. Click the **Generate Project Content** button.
    Add the code from the supplied project files pdc_thread_entry.c, ov7670.h, ov7670.c, ov7670_registers.h or copy over the generated pdc_thread_entry.c, ov7670.h, ov7670.c, ov7670_registers.h files.
12. Connect to the host PC through a micro USB cable to J17 on DK-S7G2. Connect the 5V power adapter.

13. Start to debug the application using the following steps.
   View the output on the LCD screen or a variable named g_display_fb_background can be watched in the Debug process.
   A. When the application is running, select the **Memory** tab in the Debug perspective.
   B. Click the **Add Memory Monitor** button.
   C. Enter g_display_fb_background[0] or g_display_fb_background[1].
   D. Click the **New Rendering** tab, select **Raw Image**.
   E. Confirm by pressing the **Add Rendering(s)** button.
   F. Click the **Raw Image Format** button.
   G. Enter the dimensions 480 for width and 272 for height and for encoding add RGB 16bpp (5:6:5).
   H. Confirm the start position is set to Top.
   When the image format is set, the image should be visible soon. For a real-time refresh, click the **Relevant** button (two arrows and a play symbol icon).
   Note: To re-adjust the focus of the OV7670 camera, you can rotate the lens. This is typically required since the camera is very sensitive to the lens setting.

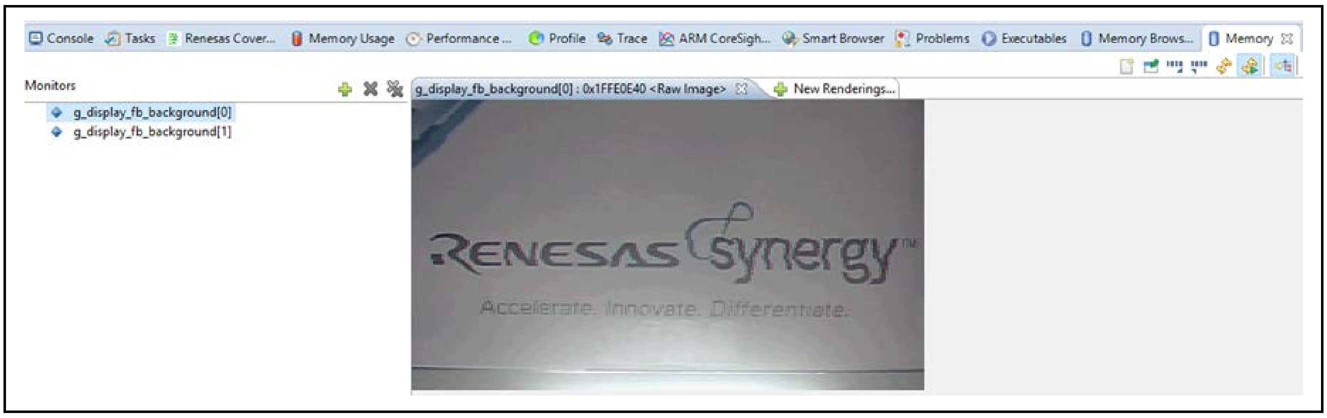

Figure 5. Example Output from PDC Application Project

10. PDC HAL Module Conclusion

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

11. PDC HAL Module Next Steps

After you have mastered a simple PDC HAL module project, you may want to review a more complex example. You may need to write the captured image into external memory or transfer it through USB or Ethernet interfaces. In that case, please refer to USBX and NetX user manuals. These documents are available through the links shown in the References section at the end of this document.

Other application projects and application notes that demonstrate PDC HAL module use are available as described at the end of the References section at the end of this document.

12. PDC 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_pdc` module reference materials and resources are available on the Synergy Knowledge Base: https://en-support.renesas.com/knowledgeBase/16977498.

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	Jun.01.17	—	Initial version
1.01	Aug.31.17	—	Update to Hardware and Software Resources Table
1.02	Mar.02.19	—	Update to Hardware and Software Resources Table

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

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/.