Renesas Synergy™ Platform

Touch Panel Framework Module Guide

R11AN0179EU0103
Rev.1.03
Dec 12, 2018

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 on 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 Touch Panel Framework module provides a high-level API to read messages from a Touch Controller and is implemented using an I^2C port. The module uses the IIC or SCI peripherals on the Synergy MCU Series.

Contents

1. Touch Panel Framework Module Features

The Touch Panel Framework supports the following features:

- Reads data from a touch controller
- Publishes touch messages to any queue subscribed to touch events in the messaging framework
- Provides position data (X and Y coordinates)
- Provides the touch event type (down, up, move, hold, or invalid)
- Supports I^{2}C-based touch panel implementations with external interrupts

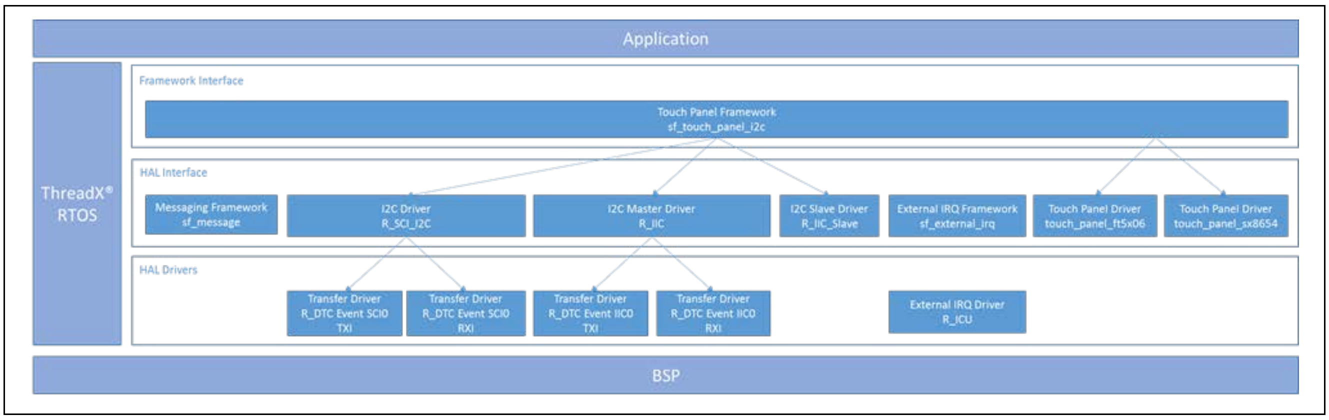


Figure 1 Touch Panel Framework Module Block Diagram

2. Touch Panel Framework Module APIs Overview

The Touch Panel Framework module defines APIs for key functions such as opening, calibrating, starting, stopping or closing. 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 Touch Panel Framework Module API Summary

Function Name	Example API Call and Description
.open	`g_sf_touch_panel_i2c0.p_api->open(g_sf_touch_panel_i2c0.p_ctrl, g_sf_touch_panel_i2c0.p_cfg);` Create required RTOS objects, call lower level module for hardware specific initialization, and create a thread to post touch data to a message queue.
.calibrate	`g_sf_touch_panel_i2c0.p_api->calibrate(g_sf_touch_panel_i2c0.p_ctrl, &expected, &actual, timeout);` Begin calibration routine based on provided expected coordinates. Returns SSP_SUCCESS only if the tolerance is longer than the distance from the expected touch point to the actual touch point (using the formula below): p_calibrate->tolerance_pixels ^ 2 > (p_calibrate->x - x_measured) ^ 2 + (p_calibrate->y - y_measured) ^ 2
.start	`g_sf_touch_panel_i2c0.p_api->start(g_sf_touch_panel_i2c0.p_ctrl);` Start scanning for touch events.
.stop	`g_sf_touch_panel_i2c0.p_api->stop (g_sf_touch_panel_i2c0.p_ctrl);` Stop scanning for touch events.
.reset	`g_sf_touch_panel_i2c0.p_api->reset(g_sf_touch_panel_i2c0.p_ctrl);` Reset touch controller if reset pin is provided, and resets the I2C bus.

Function Name	Example API Call and Description
.close	`g_sf_touch_panel_i2c0.p_api->close(g_sf_touch_panel_i2c0.p_ctrl);` Terminate touch thread and close channel at HAL layer.
.versionGet	`g_sf_touch_panel_i2c0.p_api->versionGet(&version);` Retrieves API version and stores it in 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	API call successful.
SSP_ERR_ASSERTION	A pointer parameter was NULL, or a lower level driver reported this error.
SSP_ERR_INTERNAL	The touch panel thread or event flags could not be created, or a lower level driver reported this error.
SSP_ERR_CALIBRATE_FAILED	Actual touch value was not in expected range.
SSP_ERR_NOT_OPEN	Touch panel is not configured. Call SF_TOUCH_PANEL_I2C_Open.
SSP_ERR_IN_USE	Mutex was not available, or a lower level driver reported this error.

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. Touch Panel Framework Module Operational Overview

The Touch Panel Framework module reads data from a touch controller. It publishes touch messages to any queue subscribed to touch events in the messaging framework. The touch event contains position data (X and Y coordinates) and the touch event type (down, up, move, hold, or invalid). The Touch Panel Framework supports I^{2}C-based touch panel implementations with external interrupts. The Touch Panel Framework creates a thread internally to read the touch controller.

The typical flow when using the Touch Framework is:

1. Pend on a touch message.
2. Parse the touch message.

3.1 Touch Panel Framework Module Important Operational Notes and Limitations

3.1.1 Touch Panel Framework Module Operational Notes

Initial Configuration

The e^2 studio ISDE automatically generates the **Touch Event Class** and the **New Data Event** in the **Messaging** tab of the Synergy Project Configurator when you add the Touch Panel Framework to your project.

Use the following steps to configure Touch Event message subscriber(s) in your project:

1. Select the **Touch Event Class** on the **Event Classed** pane located on the e^2 studio Configurator Messaging tab.
2. **Check** the check box of **subscriber thread** on the **Touch Subscribers** pane in the **Messaging** tab.
   If no thread is shown in the Touch Subscribers pane, add the thread: Click the **New** icon (on the top right of the pane).

Create a Custom Touch Panel Chip Driver

1. To create your custom **Touch Chip Driver**, refer to the existing **touch driver code** in **SSP** at **synergy/ssp_supplemental/ touch_drivers.**
   Note the directory is visible only when any of existing touch chip driver is selected in the Renesas Synergy

configurator.

2. Go to the Synergy configurator and select **New** > **Framework** > **Input** > **Touch Panel Framework on sf_touch_panel_i2c**> **Add Touch Driver**.
3. Select any existing touch driver for reference. Generate the project content using the following instructions and pseudo code to connect it to the Touch Panel Framework:
   - Implement **Touch Chip Driver Instance** to enable the T**ouch Panel Framework** to attach the driver.
   - Implement the **@ref payloadGet** function to obtain the touch events and touch coordinates from a Touch Panel Controller device through the I^2C interface.
   - Implement a reset function to **@ref reset** a Touch Panel Controller device by an associated GPIO pin and I^2C interface.

Configure the Custom Touch Panel Chip Driver

To configure a **Custom Touch Chip Driver** to a project, follow the steps below:

1. Create a Synergy C project.
2. Update existing touch driver XML for the custom touch driver using the following steps:
   A. Go to the **module_descriptions** folder under the project root folder.
   B. Edit the touch driver XML: **Renesas##HAL Drivers##touch panel##touch_panel_i2c_ft5x06####<version>** OR **Renesas##HAL Drivers##touch panel##touch_panel_i2c_sx8654####<version>**.
   C. Change name of the instance structure in the value field with the name of the **custom driver instance structure**.
   D. Add the new instance declaration after the **Property** tab
   E. Rename all touch chip numbers in the XML with the custom touch chip number.
   F. Save and close the XML file.
3. Open the project configurator. Add the touch panel framework by selecting **New** > **Framework** > **Input** > **Touch Panel Framework on sf_touch_panel_i2c**.
   If the configurator was already open, close and open it again.
4. Configure all components.
5. Add the custom touch panel chip driver to the **Add Touch Driver** box.
   The custom touch driver opens and is visible if the XML is modified correctly (from substeps A through F).
6. Generate the **Project Content** and add a new directory structure (for example, touch_panel_i2c_xxxxxx) under **synergy/ssp_supplemental/touch_drivers**.
   The directory structure should look like:
   synergy/ ssp_supplemental/ touch_drivers/ touch_panel_i2c_xxxxxx
   Replace the **xxxxxx** portion with the touch controller part number.
7. Add the custom driver code to this directory.
   This code was created in Create Custom Touch Panel Chip Driver. Copy and paste this code to **ssp_supplemental/ touch_drivers/touch_panel_i2c_xxxxxx**.
8. Exclude any existing driver from the build, if applicable.
   Right-click the .c file > **Exclude from build…** > **Select all** > **OK**.
9. Build the code.

3.1.2 Touch Panel Framework Module Limitations

- User callback is not available in the I^2C-bus communication procedure.
- The reset API is only available after the following conditions are satisfied: (1) The stop API is called. (2) A touch event happened and the Touch Panel Framework posts a touch event message.
- The add-on directory for the Touch Panel driver was changed from **renesas_sybd** (in SSP **v1.2.0-b.1**) to **ssp_supplemental** (SSP **v1.2.0**).
  If SSP **v1.2.0-b.1** and SSP **v1.2.0** are installed in the development environment, both touch panel modules will be selected in the **Component** tab.
  In your project, exclude the **renesas_sybd** folder from the build. **Right click** on the **renesas_sybd** folder -> **Exclude from build->Select all->OK**.
- Refer to the most recent *SSP Release Notes* for any additional operational limitations for this module.

4. Including the Touch Panel Framework Module in an Application

This section describes how to include the Touch Panel Framework module in an application using the SSP Configurator.

Note: This section assumes 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 *SSP Getting Started Guide* listed in the References section at the end of this document to learn how to manage each of these important steps in creating SSP-based applications.

To add the Touch Panel Framework to an application, simply add it to a thread using the stacks selection sequence given in the following table. (The default name for the Touch Panel Framework is **sf_touch_panel_i2c**. This name can be changed in the associated **Properties** window.)

Table 3 Touch Panel Framework Module Selection Sequence

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_touch_panel_i2c0 Touch Panel Framework on sf_touch_panel_i2c	Threads	New Stack> Framework> Input> Touch Panel Framework on sf_touch_panel_i2c

When the Touch Panel Framework on sf_touch_panel_i2c is added to the thread stack, the configurator automatically adds any needed lower-level modules. Any drivers that need additional configuration information will have box text highlighted in Red. Modules with a Gray band are individual, stand-alone modules. Modules with a Blue band are shared or common. They need only be added once and can be used by multiple stacks. Modules with a Pink band can require the selection of lower level drivers; these drivers are either optional or recommended (indicated in the block by text.) If the addition of lower level drivers is required, the module description will include **Add** in the text. Clicking on any Pink-banded modules brings up the **New** icon and displays possible choices.

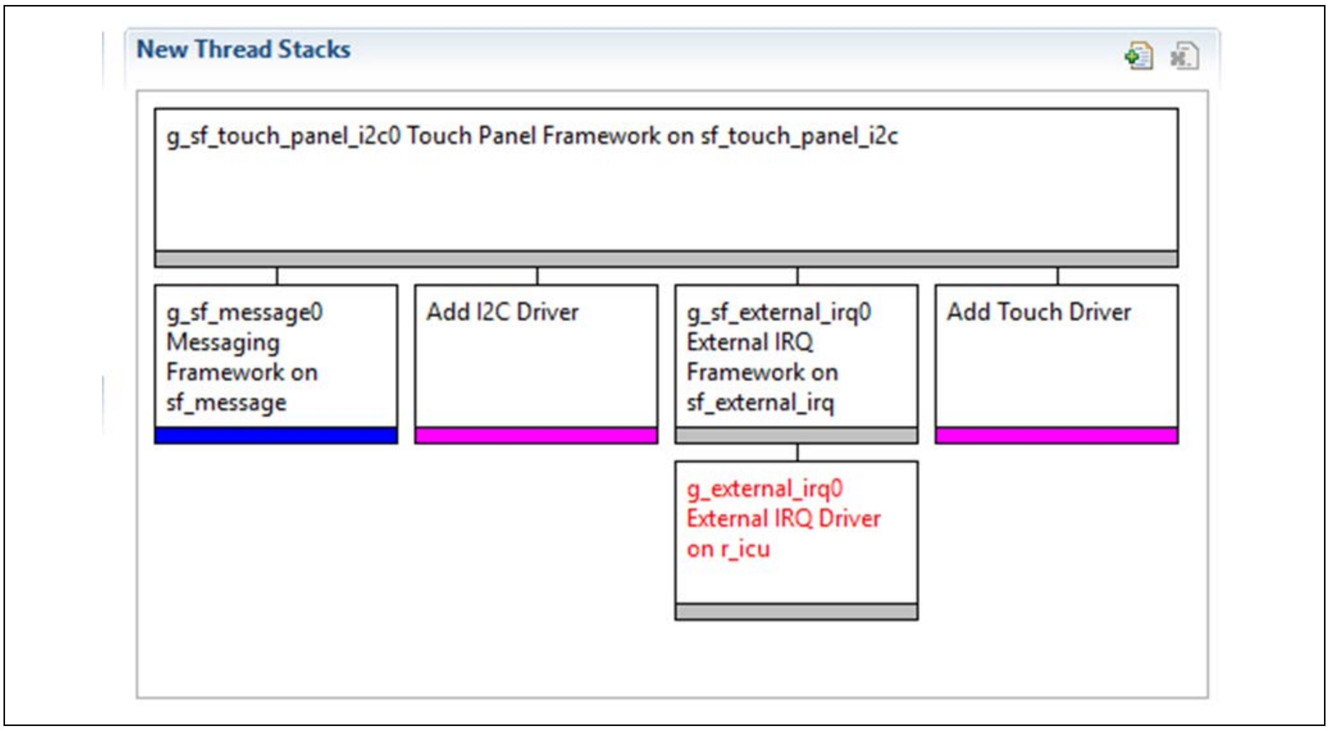

New Thread Stacks
g_sf_touch_panel_i2c0 Touch Panel Framework on sf_touch_panel_i2c
g_sf_message0 Messaging Framework on sf_message
Add I2C Driver
g_sf_external_irq0 External IRQ Framework on sf_external_irq
Add Touch Driver
g_external_irq0 External IRQ Driver on r_icu

Figure 2 Touch Panel Framework Module Stack

5. Configuring the Touch Panel Framework Module

The Touch Panel Framework module must be configured by the developer for the desired operation. The SSP Configuration window will automatically identify (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 not available for changes. Locked properties are identified with a lock icon in the ISDE Properties window. This approach simplifies the process to be 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. Select the indicated module and then view the Properties window. The interrupt settings are often toward the bottom of the properties list. Scroll down until they become available. Note that the interrupt priorities listed in the Properties window in the ISDE will include an indication as to the validity of the setting based on the MCU targeted (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 Touch Panel Framework and explore the property settings in parallel with looking over the following configuration table settings. This will help orient you and can be a useful ‘hands-on’ approach to learning the ins and outs of developing with SSP.

Table 4 Configuration Settings for the Touch Panel Framework Module on sf_touch_panel

Parameter	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking.
Name	g_sf_touch_panel_i2c0	Module name.
Thread Priority	Default: 3	Thread priority selection.
Hsize Pixels	Default: 800	Hsize pixels selection.
Vsize Pixels	Default: 480	Vsize pixels selection.
Update Hz	Default: 10	Update hz selection.
Reset Pin	IOPORT_PORT_10_PIN_02	Reset pin selection.
Touch Event Class Instance Number	0	Touch event class instance number selection.
Touch Coordinate Rotation Angle (Clockwise)	0, 90 (select this if 'Screen Rotation Angle' in GUIX Port is '270'), 180, 270 (select this if 'Screen Rotation Angle' in GUIX Port is '90') Default: 0	Touch coordinate rotation angle selection.

Note: The example values and defaults are for a project using the S7G2 Synergy MCU Group. 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 screen sizes. The configurable properties for the lower-level stack modules are given in the following sections for completeness and as a reference.

Note: Most of the property settings for lower-level modules are intuitive and usually can be determined by inspection of the associated properties window from the SSP Configurator.

5.1 Configuration Settings for the Touch Panel Framework Lower-Level Modules

Typically, only a small number of settings must be modified from the default for lower-level drivers. These setting are indicated via 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 will be locked to prevent user modification. The Touch Panel Framework consists of the following lower-level drivers:

- Messaging Framework on sf_message
- I^{2}C Driver: I^{2}C Master Driver on RIIC or I^{2}C Master Driver on r_sci_i2c

5.2 External IRQ Framework on sf_external_irq

- Touch Panel Driver

The following table identifies all the settings for the lower-level drivers within the properties section for the module.

Table 5 Configuration for the Messaging Framework on sf_message

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enables or disables the parameter checking.
Message Queue Depth (Total number of messages to be enqueued in a Message Queue)	16	Specify the size of Thread X Message Queue in bytes for Message Subscribers. This value is applied to all the Message Queues.
Name	g_sf_message0	The name of Messaging Framework module control block instance.
Work memory size in bytes	2048	Specify the work memory size in bytes. Choosing a small number results a small number of buffers which can be allocated at the same time (You can confirm the total buffer number on: sf_message_ctrl_t::number_of_buffers). If the value is smaller than the peak number of messages to be posted at the same time, the Framework occurs a buffer allocation failure affecting system performance.
Pointer to subscriber list array	p_subscriber_lists	Specify the name of pointer to the Subscriber List array.
name of the block pool internally used in the messaging framework	sf_msg_blk_pool	The name of memory block memory the Framework creates in the control block. This parameter might be useful for debugging purpose but NULL can be specified for saving memory.

Table 6 Configuration for the DTC HAL Module on r_dtc Event SCI0 TXI

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Enabled	Software start selection.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table.
Name	g_transfer0	Module name
Mode	Normal	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
Interrupt Frequency	After all transfers have completed	Interrupt frequency 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)	Event SCI0 TXI	Activation source selection
Auto Enable	FALSE	Auto enable selection

ISDE Property	Value	Description
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event 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	ELC software event interrupt priority selection.

Table 7 Configuration for the DTC HAL Module on r_dtc Event SCI0 RXI

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Disabled	Software start selection.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table.
Name	g_transfer1	Module name
Mode	Normal	Mode selection
Transfer Size	1 Byte	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	Destination	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency 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)	Event SCI0 RXI	Activation source selection
Auto Enable	FALSE	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event 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	ELC software event interrupt priority selection.

Table 8 Configuration for the I^{2}C Master Driver on r_riic

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable parameter error checking.
Name	g_i2c0	Module name.
Channel	0, 1, or 2	Specify the IIC channel to be used with this configuration.
Rate	Standard, Fast-mode, Fast-mode Plus Default: Standard	Standard, Fast, and Fast-plus. (See IIC Rate Calculation.)
Slave Address	0x00	Set the address of the slave device the I^{2}C master will be communicating with.
Address Mode	7-Bit, 10-Bit Default: 7-Bit	Only 7-bit addresses are currently supported.
Callback	NULL	A user callback function can be registered in i2c_api_master_t: open. If this callback function is provided, it will be called from the interrupt service routine (ISR) for each of the conditions defined in i2c_event_t. **Warning**: Since the callback is called from an ISR, do not use blocking calls or lengthy processing. Spending excessive time in an ISR can affect the responsiveness of the system.
Receive 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: Priority 2	Receive interrupt priority selection.
Transmit 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: Priority 2	Transmit interrupt priority selection.
Transmit 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: Priority 2	Transmit end interrupt priority selection.

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

Table 9 Configuration for the DTC HAL Module on r_dtc IIC0 TXI

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Enabled	Software start selection.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table.
Name	g_transfer0	Module name
Mode	Normal	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
Interrupt Frequency	After all transfers have completed	Interrupt frequency 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)	Event IIC0 TXI	Activation source selection
Auto Enable	FALSE	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event 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	ELC software event interrupt priority selection.

Table 10 Configuration for the DTC HAL Module on r_dtc IIC0 RXI

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Disabled	Software start selection.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table.
Name	g_transfer1	Module name
Mode	Normal	Mode selection
Transfer Size	1 Byte	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	Destination	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency 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)	Event IIC0 RXI	Activation source selection
Auto Enable	FALSE	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event 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	ELC software event interrupt priority selection.

Table 11 Configuration for the External IRQ Framework on sf_external_irq

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Controls whether to include code for API parameter checking.
Name	g_sf_external_irq0	Framework name.
Event	None, Semaphore Put Default: Semaphore Put	Event selection.

Table 12 Configuration for the External IRQ HAL Module on r_icu

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Parameter checking setting enables or disables the addition of parameter checking code.
Name	g_external_irq0	Module name.
Channel	0	Specifies the hardware IRQ channel used.
Trigger	Falling, Rising, Both Edges, Low Level Default: Rising	Configures edge or level triggering.
Digital Filtering	Enabled, Disabled Default: Disabled	Digital filter enable/disable.
Digital Filtering Sample Clock (Only valid when Digital Filtering is Enabled)	PCLK/1, PLCK/8, PLCK/32, PCLK/64 Default: PCKL/64	Sets noise filter sampling period.
Interrupt enabled after initialization	True, False Default: True	Determines if the interrupt is enabled immediately after initialization.
Callback	NULL	A user callback function can be registered in external_irq_api_t::open. If this callback function is provided, it is called from the interrupt service routine (ISR) each time the IRQn triggers. **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	An Interrupt priority can be registered in external_irq_cfg_t::irq_ipl.

Table 13 Configuration for the Touch Panel Driver on touch_panel_ft5x06

ISDE Property	Value	Description
Name	g_sf_touch_panel_i2c_chip_ft5x06	Module name.

Table 14 Configuration for the Touch Panel Driver on touch_panel_sx8654

ISDE Property	Value	Description
Name	g_sf_touch_panel_i2c_chip_sx8654	Module name.

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

5.3 Clock Configuration for the Touch Panel Framework

The I^{2}C interface implementation example described here uses the SCI peripheral. Other implementations might have different selections, but can be inferred from the following example.

The SCI peripheral module uses PCLKB as its clock source. The PCLKB frequency is set by using the SSP configurator clock tab, prior to a build, or by using the CGC Interface at run-time. During configuration, the I^{2}C transfer rate is calculated and set internally by the driver, based on the user selected PCLB rate and the user selected transfer rate. If the PCLKB is configured in such a manner that the user selected rate cannot be achieved, an error will be returned when initializing the driver.

5.4 Pin Configuration for the Touch Panel Framework

The I^{2}C interface implementation example described here uses the SCI peripheral. Other implementations might have different selections, but can be inferred from the following example.

Synergy Kit-specific settings for pins are shown in the following section. The SCI 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 lists the method used for selecting the pins within the SSP configuration window and the subsequent table has an example selection for the I^{2}C pins. The settings in this example are for a project using the Synergy S7G2 and the SK-S7G2 Kit. Other Synergy Kits and other Synergy MCUs may have different available pin configuration settings

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

Table 15 Pin Selection Sequence for I^{2}C Master Driver on SCI

Resource	ISDE Tab	Pin selection Sequence
SCI0	Pins	Select **Peripherals > Connectivity:SCI > SCI0**

Note: The selection sequence assumes SCI0 is the desired hardware target for the driver.

Table 16 Pin Configuration Settings for I^{2}C Master Driver on SCI

Pin Configuration Property	Value	Description
Operation Mode	Disabled, Asynchronous UART, Synchronous UART, Simple I^{2}C, Simple SPI, SmartCard (Default: Disabled) **Simple I^{2}C**	Select Simple I^{2}C as the Operation Mode for SPI on SCI
RXD1_SCL1_MISO1	None, P212, P708 (Default: None)	SCL Pin
TXD1_SDA1_MOSI1	None, P213, P709 (Default: None)	SDA Pin

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

5.5 Touch Panel Framework Module Synergy Kit Specific Settings

In addition to the above selections some kits specific settings may be needed, based on the touch chip used. These settings are summarized below as examples of how you may need to configure these for your own custom touch controller device.

Configure the following External IRQ settings:

Table 17 External IRQ Kit-specific settings

Target	Select External IRQ settings
DK-S7G2 (Touch Chip SX8654)	• Channel: 7 • Trigger: Falling
SK-S7G2 (Touch Chip SX8654)	• Channel: 9 • Trigger: Falling
PE-HMI1-S7G2 (Touch Chip FT5x06)	• Channel: 12 • Trigger: Falling

Note: Common setting for all boards:

— Digital Filter settings: Any

— Interrupt enabled after initialization: True

— Callback: NULL

Table 18 I^{2}C Driver Kit-specific settings

Target	Select I^{2}C Driver settings
DK-S7G2	**r_sci_i2c** • Channel: 7 • Rate: Fast-Mode • Slave Address: 0x48 • Address Mode: 7-bit
SK-S7G2	**r_riic**: • Channel: 2 • Rate: Fast-Mode • Slave Address: 0x48 • Address Mode: 7-bit
PE-HMI1-S7G2	**r_riic** • Channel: 1 • Rate: Fast-Mode • Slave Address: 0x38 • Address Mode: 7-bit

Note: Common setting for all boards:

— Callback: NULL

Table 19 Touch Panel Framework Module Kit-specific settings

Target	Configure
DK-S7G2	Touch Chip: **g_sf_touch_panel_i2c_chip_sx8654** • H size Pixels: 480 • V size Pixels: 272 • Reset Pin: IOPORT_PORT_07_PIN_11
SK-S7G2	Touch Chip: **g_sf_touch_panel_i2c_chip_sx8654** (see note) • H size Pixels: 240 • V size Pixels: 320 • Reset Pin: IOPORT_PORT_06_PIN_09
PE-HMI1-S7G2	Touch Chip: **g_sf_touch_panel_i2c_chip_ft5x06** (see note) • H size Pixels: 800 • V size Pixels: 480 • Reset Pin: IOPORT_PORT_10_PIN_02

Notes:

- Common setting for all boards: Thread Priority: Any

The Touch Chip Drivers for SX8654 and FT5x06 are built in the SSP. You can select these drivers when configuring the module.

6. Using the Touch Panel Framework Module in an Application

Once the module has been configured and the files generated, the touch panel framework is ready to be used in an application. The Touch Panel Framework opens automatically from the Synergy-generated file once the application starts.

Suggested use of the Touch Panel Framework module:

1. Start the touch panel framework using start API of touch panel framework.
2. Pend on a touch message using the pend API
3. Parse the touch message with application code
4. Operate on the received data as needed by the application. Go to step 2.

The following diagram is a typical operational flow that illustrates these common steps:

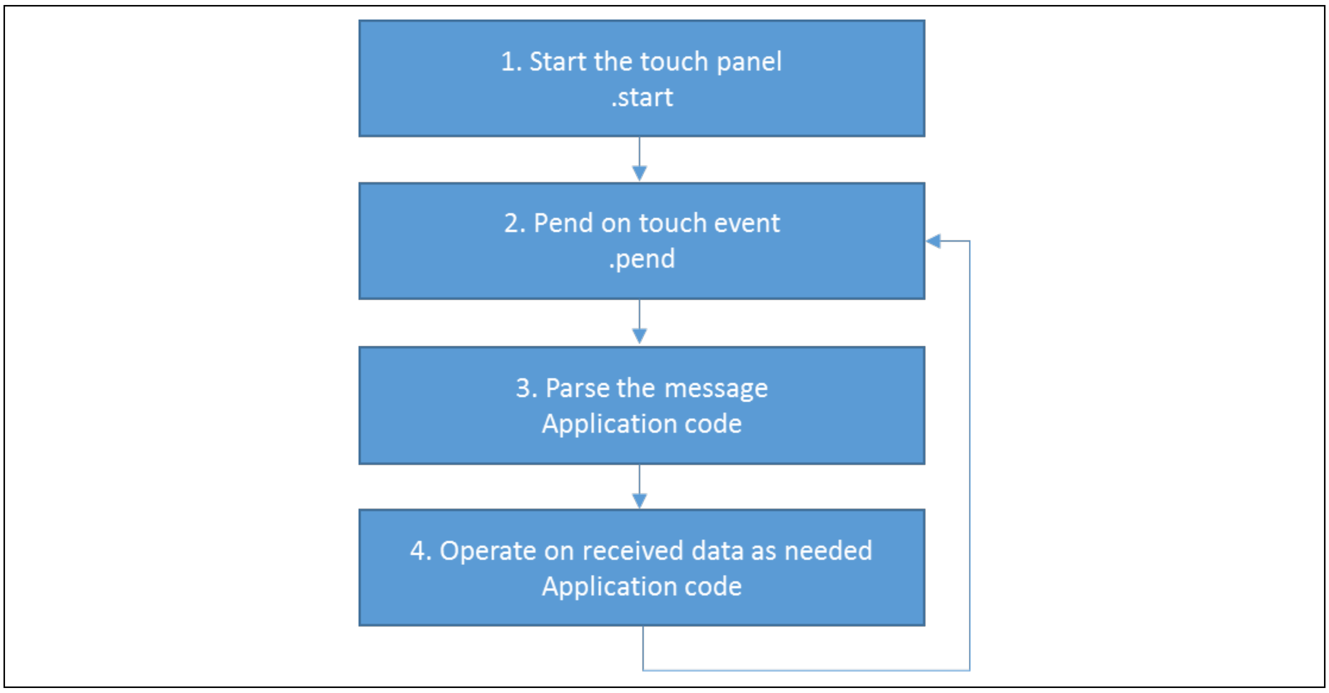


Figure 3 Flow Diagram of a Typical Touch Panel Framework Module Application

7. The Touch Panel Framework Module Application Project

The application project associated with this module guide demonstrates the aforementioned steps in an example application. You may want to import and open the application project within ISDE and view the configuration settings for the Touch Panel framework module. You can also read over the code, in the `Touch_Panel_Framework_Module_Guide_AP`, which is used to illustrate the Touch Panel framework module APIs in a complete design.

The application project demonstrates the typical use of the Touch Panel framework module APIs. The application project main thread entry enables the Touch Panel framework module. Thereafter, it pends for a touch event message, and parses it upon reception.

The following table identifies the target versions for the associated software and hardware used by the application project.

Table 20 Software and Hardware Resources Used by the Application Project

Resource	Revision	Description
e^2 studio ISDE	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 diagram lists the application project flow in simple steps:

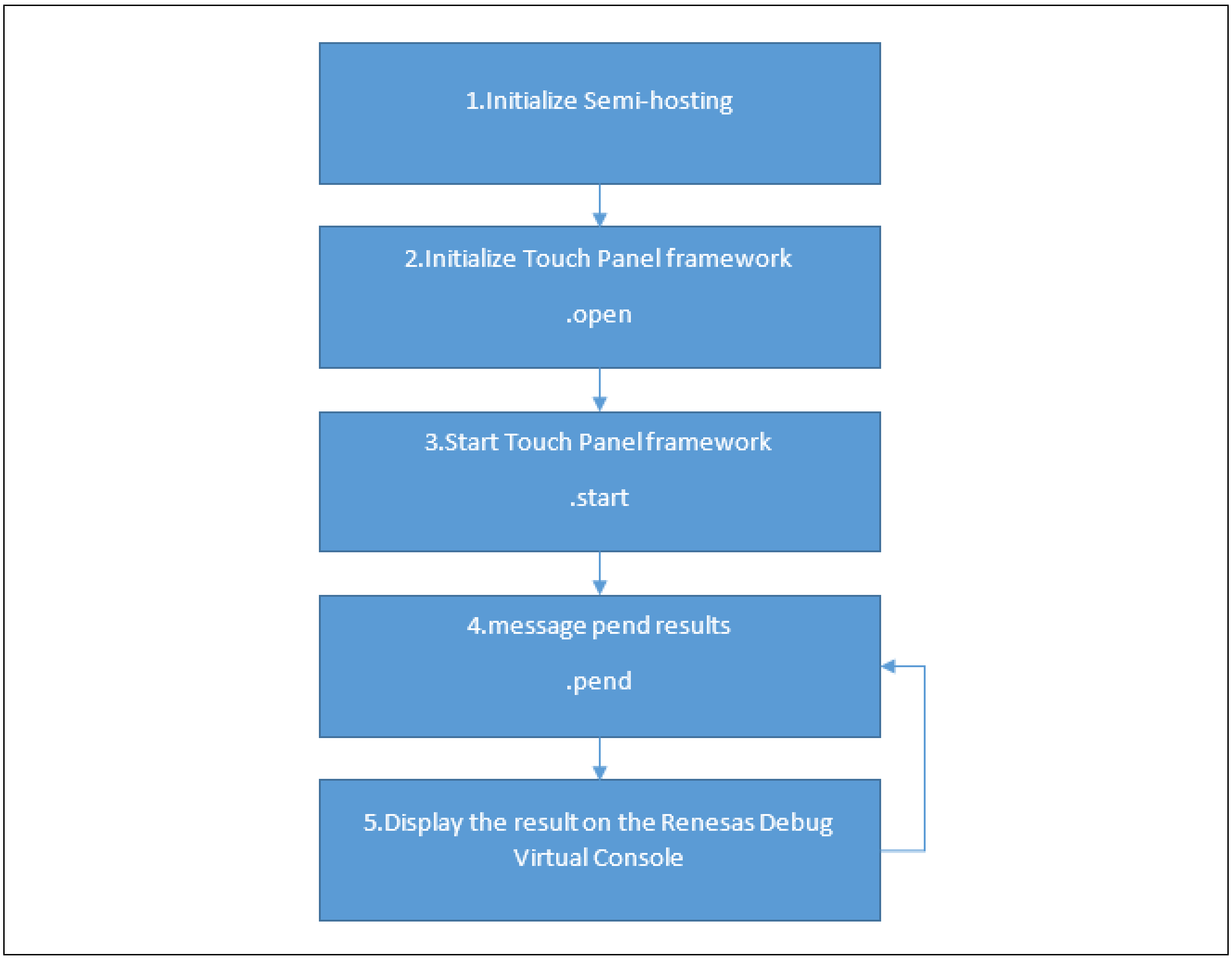

1.Initialize Semi-hosting
2.Initialize Touch Panel framework
.open
3.Start Touch Panel framework
.start
4.message pend results
.pend
5.Display the result on the Renesas Debug Virtual Console

Figure 4 Touch Panel Framework Module Application Project Flow Diagram

The `sf_touch_panel_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 along with the description provided to help identify key uses of APIs.

The first section of `sf_touch_panel_mg.c` has the header files that reference the Touch Panel instance structure, and a code section that allows semi-hosting to display results using printf().

The entry function for the main program control section is g_sf_touch_panel_i2c_framework_enable. Within the function, the Touch Panel Framework is enabled, and it enters the g_sf_touch_panel_event function, which pends for a touch event message. This touch event message is sent from the Touch Panel internal thread using the Messaging Framework. The message is parsed upon reception. A short message is printed using printf to indicate the type of touch event, and the coordinates of the touch. LEDs also illuminate to the touch event:

1. A pen down touch event has been received: LED 1 illuminates.
2. A pen up touch event has been received: NO LEDs illuminate.
3. A pen hold touch event has been received: LEDs 1 and 2 illuminate.
4. A pen move touch event has been received: LEDs 1 and 3 illuminate.

Note: It is assumed that you are familiar with using printf() and the debug console in the Synergy Software Package. If you are unfamiliar, refer to the Knowledge Base article, “How do I Use Printf() with the Debug Console in the Synergy Software Package,” available in the Reference section at the end of this document. Alternatively, you can see the results via 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. On the SK-S7G2 board, the selected I^{2}C driver should be the I^{2}C Master Driver on r_riic. The following table lists properties with the values set for this specific project. You can also open the application project and view these settings in the property window as a hands-on exercise.

Table 21 Touch Panel Framework Module Configuration - Transfer Driver on r_dtc - Settings for the Application Project

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Selects if code for parameter checking is to be included in the build
Software Start	Enabled, Disabled Default: Disabled	Software start selection.
Linker section to keep DTC vector table	.ssp_dtc_vector_table	Linker section to keep DTC vector table.
Name	g_transfer1	Module name
Mode	Normal	Mode selection
Transfer Size	1 Byte	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	Destination	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency 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)	Event IIC0 RXI	Activation source selection
Auto Enable	FALSE	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection

ISDE Property	Value	Description
ELC Software Event 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	ELC software event interrupt priority selection.

Table 22 Touch Panel Framework Module Configuration Settings for the Application Project

ISDE Property	Value Set
Thread Priority	Priority 3
Name	g_sf_touch_panel_i2c
Unit	3
Hsize Pixels	240
Vsize Pixels	320
Update HZ	10
Reset Pin	IOPORT_PORT_06_PIN_09

Table 23 Touch Panel Framework Module Configuration – I^{2}C Master Driver on r_riic - Settings for the Application Project

ISDE Property	Value Set
Name	g_i2c
Channel	2
Rate	Fast-Mode
Slave Address	0x48
Address Mode	7-Bit
Callback	Null

Table 24 Touch Panel Framework Module- Lower Level External IRQ Driver - Configuration Settings for the Application Project

ISDE Property	Value Set
Channel	9
Interrupt enabled after initialization	True

For this application project, the LCD has also been enabled. The following tables show the properties with the values set for this specific project.

Table 25 Pin Selection Sequence LCD_WR signal

Resource	ISDE Tab	Pin selection Sequence
LCD_WR Signal P115	Pins	Select Ports > P1 > P115 Mode: Output mode (Initial Low) Chip Input/Output: GPIO
RESET# for Touch Panel P609	Pins	Select Ports > P6 > P609 Mode: Output mode (Initial Low) Chip Input/Output: GPIO
LCD_RESET Signal P610	Pins	Select Ports > P6 > P610 Mode: Output mode (Initial Low) Chip Input/Output: GPIO
LCD_CS Signal P611	Pins	Select Ports > P6 > P611 Mode: Output mode (Initial Low) Chip Input/Output: GPIO

Table 26 Pin Configuration Settings for Driver Capability GLCDC_Controller_Pin_Option_B

Pin Configuration Property	Value	Description
Pin Group Selection	Mixed, _A only, _B only (Default: Mixed)	Select Pin Group Mapping. Select _B only.
Operation Mode	Disabled, Custom, RGB 666, RGB 565 (Default: Custom)	Select Custom.
LCD_CLK	None, P900, P101 (Default: P900)	Select P900.
LCD_DATA00	None, *P106, P804 (Default: P804)	Select P804.
LCD_DATA01	None, *P107, P803 (Default: P803)	Select P803.
LCD_DATA02	None, *P600, P802 (Default: P802)	Select P802.
LCD_DATA03	None, P606, *P601 (Default: P606)	Select P606.
LCD_DATA04	None, P607, *P602 (Default: P607)	Select P607.
LCD_DATA05	None, *P610, PA00 (Default: PA00)	Select PA00.
LCD_DATA06	None, *P609, PA01 (Default: PA01)	Select PA01.
LCD_DATA07	None, *P608, PA10 (Default: PA10)	Select PA10.
LCD_DATA08	None, *P115, PA09 (Default: PA09)	Select PA09.
LCD_DATA09	None, *P114, PA08 (Default: PA08)	Select PA08.
LCD_DATA10	None, *P113, P615 (Default: P615)	Select P615.
LCD_DATA11	None, P905, *P112 (Default: P905)	Select P905.
LCD_DATA12	None, P906, *P111 (Default: P906)	Select P906.
LCD_DATA13	None, P907, *P301 (Default: P907)	Select P907.
LCD_DATA14	None, P908, *P302 (Default: P908)	Select P908.
LCD_DATA15	None, P901, *P303 (Default: P901)	Select P901.
LCD_TCON0	None, P315, *P102 (Default: P315)	Select P315.
LCD_TCON1	None, P314, *P103 (Default: P314)	Select P314.
LCD_TCON2	None, P313, *P104 (Default: P313)	Select P313.

8. Customizing the Touch Panel Framework Module for a Target Application

Some configuration settings are normally changed by the developer from those shown in the application project. For example, if the user would like to interface a custom Touch IC to the MCU, refer to the instructions in section 3 to create a custom driver. You can then change the pin configuration in the Synergy Configurator to suit the desired connection.

9. Running the Touch Panel Framework Module Application Example

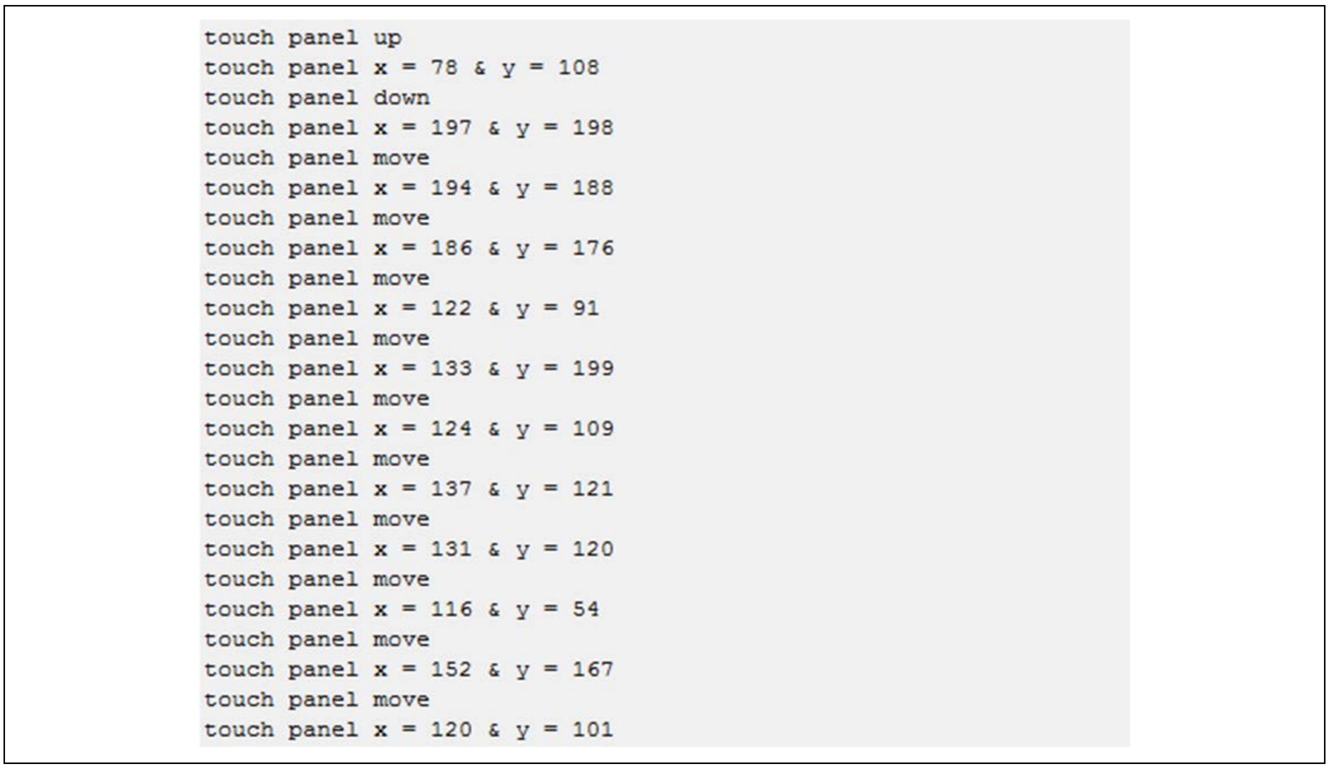
To run the application project for the Touch Panel Framework Module and watch it execute on a target kit, you can simply import it into your ISDE, compile, and run debug. In the package, refer to the *Renesas Synergy Project Import Guide* (r11an0023eu0121-synergy-ssp-import-guide.pdf) for instructions on importing the project into e^2 studio, or IAR EW for Synergy, and then building and running the application.

To implement the Touch Panel Framework module application in a new project, use the following steps to define, configure, auto-generate files, add code, compile, and debug the target kit. Following these steps is a hands-on approach that can help make the development process with SSP more practical.

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 first few chapters of the *SSP User's Manual* to learn how to accomplish these steps.

To create and run the Touch Panel application project, follow these steps:

1. Create a new Renesas Synergy project for the S7G2-SK called Touch_Panel_Framework_Module_Guide.
2. Select the SK-S7G2 board, select the BSP option and create the project.
3. Select the **Threads** tab.
4. Add a new thread with the symbol touch_panel_thread.
5. Add the Touch Panel Framework from **New Stack** > **Framework** > **Input** > **Touch Panel Framework**.
   Configure it with the appropriate properties as described in Chapter 7.
6. Click on the **Generate Project Content** button.
7. Add the code from the supplied project file `touch_panel_thread_entry.c`.
8. Connect to the host PC via a micro USB cable to J19 on SK-S7G2.
9. Start to debug the application.
10. Touch the LCD panel.
    The output can be viewed in the Renesas Debug Console is the touch panel event (example: down, up, hold, move) and coordinate for X and Y. You may also watch the LEDs illuminate.



```
touch panel up
touch panel x = 78 & y = 108
touch panel down
touch panel x = 197 & y = 198
touch panel move
touch panel x = 194 & y = 188
touch panel move
touch panel x = 186 & y = 176
touch panel move
touch panel x = 122 & y = 91
touch panel move
touch panel x = 133 & y = 199
touch panel move
touch panel x = 124 & y = 109
touch panel move
touch panel x = 137 & y = 121
touch panel move
touch panel x = 131 & y = 120
touch panel move
touch panel x = 116 & y = 54
touch panel move
touch panel x = 152 & y = 167
touch panel move
touch panel x = 120 & y = 101
```

Figure 5 Example Renesas Debug Virtual Console for Touch Panel Application Project

10. Touch Panel Framework 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 incorrect selection of low-level modules. 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. Touch Panel Framework Module Next Steps

After you have mastered a simple Touch Panel module project, you may want to review a more complex example. Other application projects and application notes that demonstrate Touch Panel functions use can be found in the reference section at the end of this document.

12. Touch Panel Framework Module Reference Information

The *SSP User Manual* is available in HTML in the SSP distribution package, and as a pdf from the Synergy Gallery.

To find the most up-to-date resource materials and their locations, visit the Synergy Knowledge Base, and search for the module name. For example, if you are looking for the resources for the sf_touch_panel_i2c module, visit "https://en-support.renesas.com/knowledgeBase" and enter "sf_touch_panel_i2c module guide resources" in the search bar. The search will present you with a list of results, and the top one will be the Resource Page for that Module Guide. The following link will take you directly to the search results for the example given: https://en-support.renesas.com/search/sf_touch_panel_i2c.

Website and Support

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

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

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	May 24, 2017	-	Version 1.0
1.01	Sep 1, 2017	-	Update to Hardware and Software Resources Table
1.02	Sep 13, 2017	-	The sample code that accompanies this application note was incorrect. It has been replaced. No changes have been made to the document itself.
1.03	Dec 12, 2018	-	Updated to SSP v1.5.0

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