Renesas Synergy™ Platform

Self-Capacitive Touch Software Application Design with Synergy S124 and S3A7 MCUs

Introduction

The goal of this application note is to provide guidelines to create a Self-Capacitive Touch Application using the Synergy™ Software Package (SSP) with Renesas Synergy™ MCUs.

The target users are application and system developers who have already worked with the e^2 studio and/or IAR Embedded Workbench® for Renesas Synergy™ (IAR EW for Synergy). The reader of this application note should have experience importing a Synergy application into e^2 studio, opening a new workspace in IAR EW for Synergy, and configuring, generating, building, downloading, and executing a Synergy application. New Synergy MCU users are recommended to exercise the Blinky Project Tutorial from the *Synergy Software Package (SSP) User's Manual* before proceeding to the applications in this application note. An average reader of this application note takes about three hours to go through the contents and exercises the sample projects.

The software projects described in this application note are built upon the Synergy Self-Capacitive Touch Application Example Kit AE-CAP1. This kit includes five PCB boards.

Renesas S124 and S3A7 Synergy MCUs are used in this kit, the method to construct and implement the Self-Capacitive Touch Application design described in this application note also apply to other Renesas Synergy MCUs.

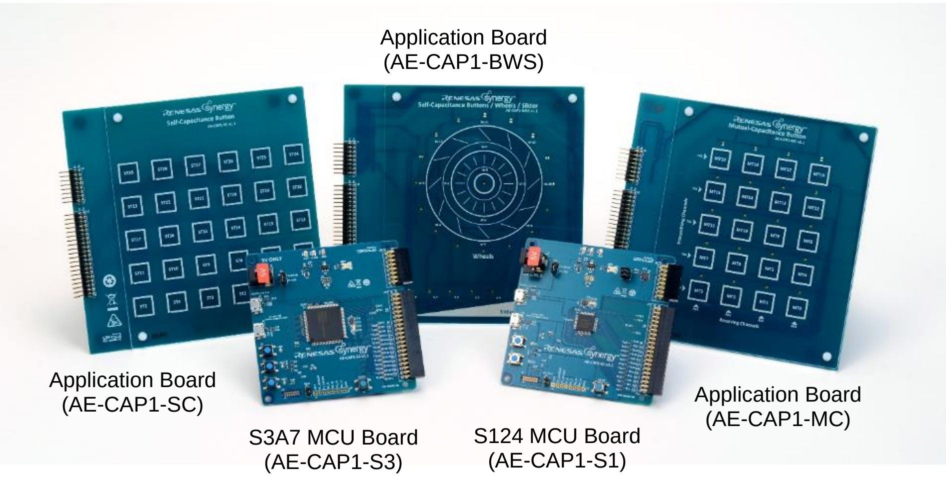

Figure 1. AE-CAP1 Kit

Target Device

Renesas S124 and S3A7 Synergy MCUs

Target Kit

AE-CAP1 v1.1

SSP Version

SSP v1.7.0 or later

ISDE Version

- e^2 studio v7.5.1 or later
- IAR EW for Synergy v8.23.3 or later
- CTW for Synergy First Step Guide 1.05.0033 and later
- Synergy Standalone Configurator (SSC) 7.5.1 or later

Contents

1. Application Overview 4
1.1 Self-Capacitance Touch with Synergy MCU 4
1.2 Synergy Software Package Frameworks 4
1.3 Overview of AE-CAP1 and Capacitive Touch Workbench for Renesas Synergy™ Operation 5
1.3.1 Capacitive Touch Pins used on AE-CAP1-S124 and AE-CAP1-S3A7 6

2. Capacitive Touch Sensing Framework 6
2.1 Capacitive Touch Button Framework 7
2.2 Capacitive Touch Slider/Wheel Framework 7
2.2.1 Wheel and Slider Position Reporting 8
2.3 Capacitive Touch Framework 9
2.4 CTSU HAL Driver 9

3. Application Software Projects 10
3.1 Build and Download the Projects 10
3.1.1 e^2 studio 10
3.1.2 IAR EW for Synergy 11
3.2 Threads in the Projects 11
3.3 Capacitive Touch Framework and Driver Settings 12
3.3.1 S124 Capacitive Touch Settings 12
3.3.1.1 Capacitive Touch Button Framework Settings 12
3.3.1.2 Capacitive Touch Slider Framework Settings 13
3.3.1.3 Capacitive Touch Framework Settings 14
3.3.1.4 CTSU HAL Driver Settings 15
3.3.2 S3A7 MCU Capacitive Touch System Settings 16
3.3.2.1 Capacitive Touch Button Framework Settings 16
3.3.2.2 Capacitive Touch Slider Framework Settings 17
3.3.2.3 CTSU HAL Driver Settings 18
3.4 Capacitive Touch Application User Callbacks 19
3.4.1 Button Callbacks 19
3.4.2 Slider/Wheel User Callbacks 20
3.4.3 Capacitive Touch Application Automatic Error Checking 22
3.5 USB Communication with CTW for Synergy 22

4. Operation of the Application Projects 23
4.1 Project AE_CAP1_S124_BWS_App 23
4.1.1 Operations for Project AE_CAP1_S124_BWS_App 23
4.2 Project AE_CAP1_S124_BWS_UART_Monitor_App 24
4.2.1 Operation for Project AE_CAP1_S124_BWS_UART_Monitoring App 25
4.2.1.1 Standalone Mode 25

4.2.1.2 Communication with CTW for Synergy ... 25
4.3 Project AE_CAP1_S124_ST_UART_Monitoring_App ... 26
4.3.1 Operation for Project AE_CAP1_ST_UART_Monitoring_App ... 26
4.3.1.1 Standalone Mode ... 26
4.3.1.2 Communication with CTW for Synergy ... 27
4.4 Project AE_CAP1_S3_BWS_App ... 27
4.4.1 Operations for Project AE_CAP1_S3A7_BWS_App ... 28
4.4.1.1 Standalone Mode ... 28
4.4.1.2 Communication with CTW for Synergy ... 29
4.5 Project AE_CAP1_S3A7_ST_App ... 29
4.6 Operations for Project AE_CAP1_S3A7_ST_App ... 29
4.6.1.1 Standalone Mode ... 29
4.6.1.2 Communication with CTW for Synergy ... 30

5. References ... 30

Revision History ... 32

1. Application Overview

1.1 Self-Capacitance Touch with Synergy MCU

Self-Capacitance Touch detects the electrostatic capacitance between a single electrode and a human fingertip (Cf). The circuit containing the electrode has a fixed electrostatic capacitance (parasitic capacitance Cp) between the electrode and the ground. Cp is measured in the Capacitive Touch system tuning process. When a fingertip is near the electrode, the sum of Cf and Cp (Ct) is measured. The difference between Ct and Cp results in Cf.

Cf = Ct – Cp

See the *Capacitive Touch Hardware Design and Layout Guidelines* to learn more about the Self-Capacitive Touch hardware design.

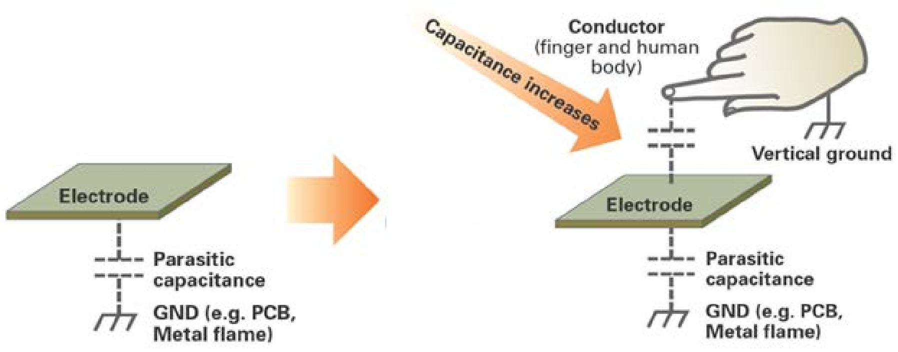

Figure 2. Self Capacitance Sensing

The Capacitive Touch Sensing Unit (CTSU) on S124 and S3A7 Synergy MCUs, as well as other Synergy MCUs such as the RX130 and RX113, use the Renesas Touch Gen 2 IP. See the *S124 and S3A7 Synergy MCUs User's Manual* for details on the internal operation of the Capacitive Touch Sensing Unit (CTSU) peripheral.

1.2 Synergy Software Package Frameworks

Figure 3 describes the major software systems used in this application project. Notice that the blue box contains all components from SSP.

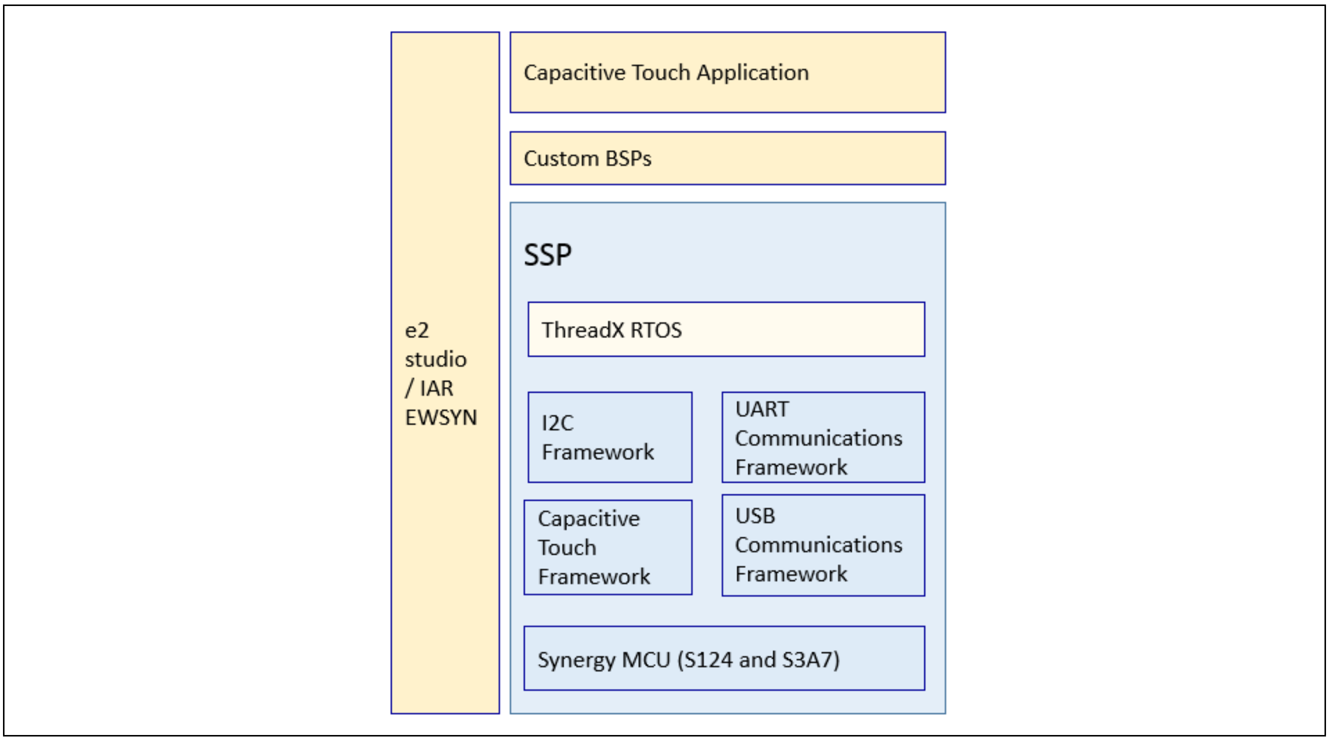

Figure 3. SSP Modules

1.3 Overview of AE-CAP1 and Capacitive Touch Workbench for Renesas Synergy™ Operation

See Figure 4 and Figure 5 shows the high-level operation of the included capacitive touch software projects.

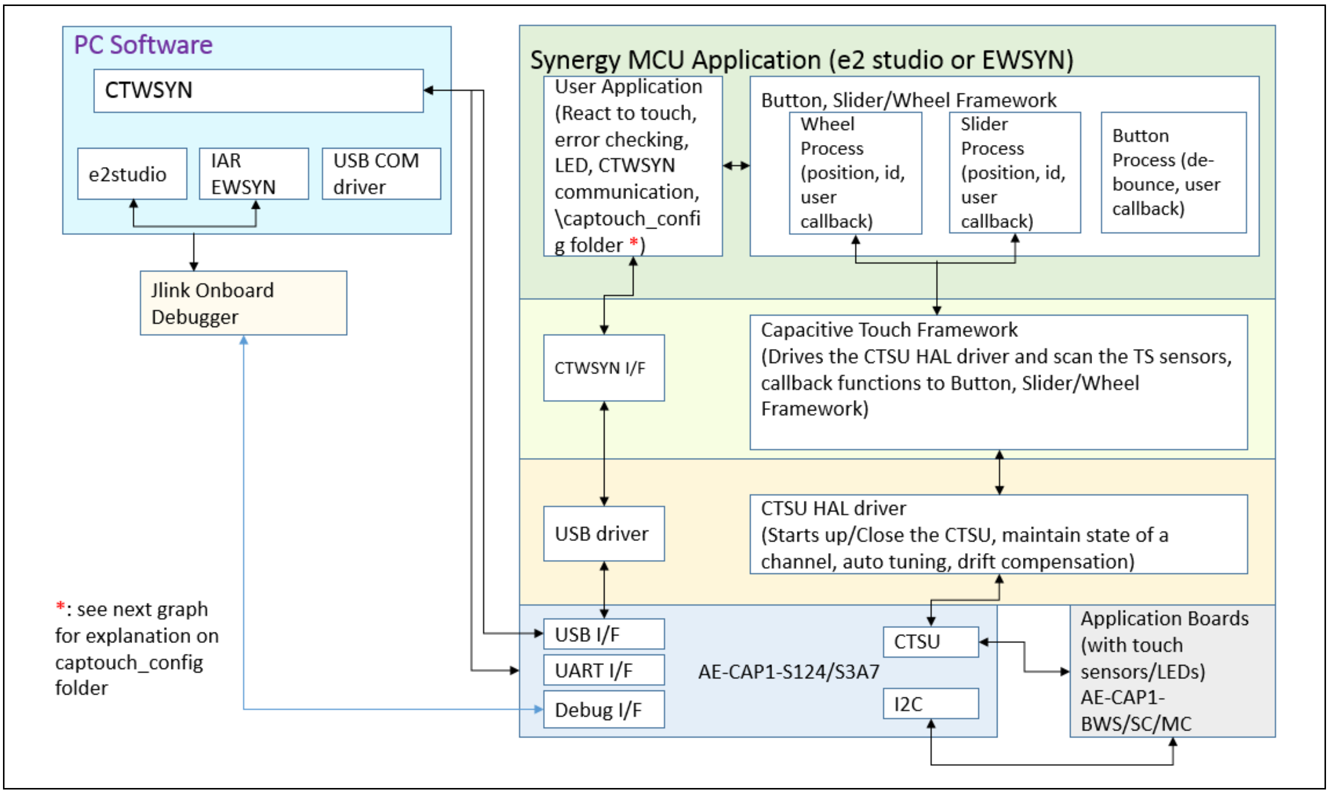

Figure 4. System Block Diagram

Some of the software projects provided functionally communicate with the Capacitive Touch Workbench for Renesas Synergy™ (CTW for Synergy) through UART or USB. The following table lists projects that have this functionality.

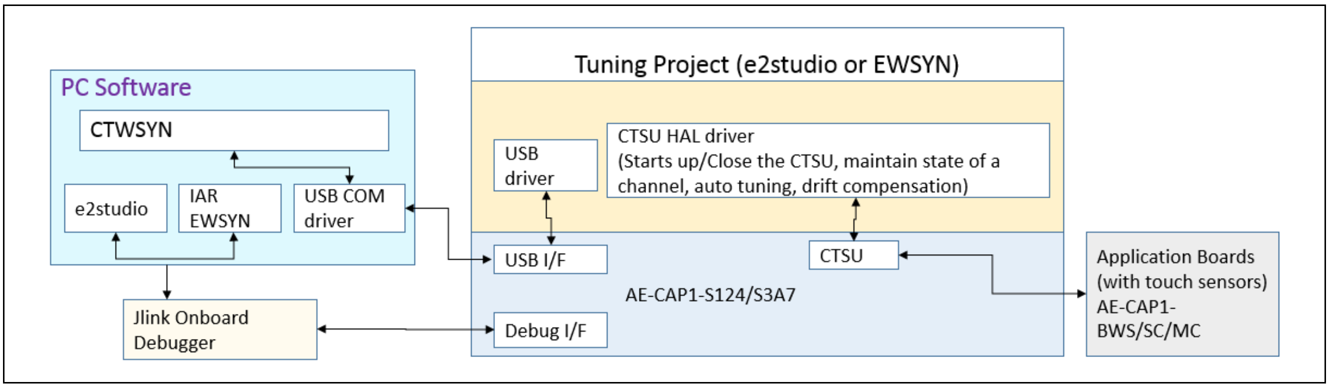

Figure 5. Tuning Process

- An e[2] studio or IAR EW for Synergy tuning project is needed for each capacitive touch application board.
- This tuning project communicates with the CTW for Synergy to generate the `\captouch_config` folder used in the application project.
- The tuning process determines the touch threshold and maximize the Capacitive Touch Application sensitivity. The resulting parameters are stored in the `\captouch_config` folder.
- You need to use identical PCLKB frequency in both the tuning and application project.

See *Tuning the* Capacitive Touch Tuning Solution application project to understand the process of tuning the Synergy Capacitive Touch System. This application note focuses on creating application projects once tuning is successfully performed.

1.3.1 Capacitive Touch Pins used on AE-CAP1-S124 and AE-CAP1-S3A7

For capacitive touch purposes, AE-CAP1 hardware and software projects implemented 28 capacitive touch pins used for the S124 Synergy MCU design and 30 Capacitive Touch pins for S3A7 Synergy MCU. For details on MCU Capacitive Touch Sensing support, see the *Microcontroller Hardware User's Manual*.

Table 1. Pins Used

S124 Pin name	S3A7 Pin name	Function
TSCAP	TSCAP	CTSU power stabilization
TS00-TS02, TS04-TS09, TS12-TS28, TS30-TS31	TS00-TS01, TS04-TS22, TS26-TS27, TS29-TS35	Capacitive touch sensing
TS03, TS10, TS11 are assigned other functionalities.	TS03 is assigned other functionality	See schematic
TS29 not functioning	TS02, TS23, TS24, TS25 not functioning	See hardware user manual

2. Capacitive Touch Sensing Framework

The Renesas Synergy™ Software Package (SSP), part of the Renesas Synergy™ Platform, is a complete integrated software package designed to provide easy to use, scalable, high quality software for embedded systems.

Figure 6 shows the structure of the software interfaces related to Capacitive Touch Sensing.

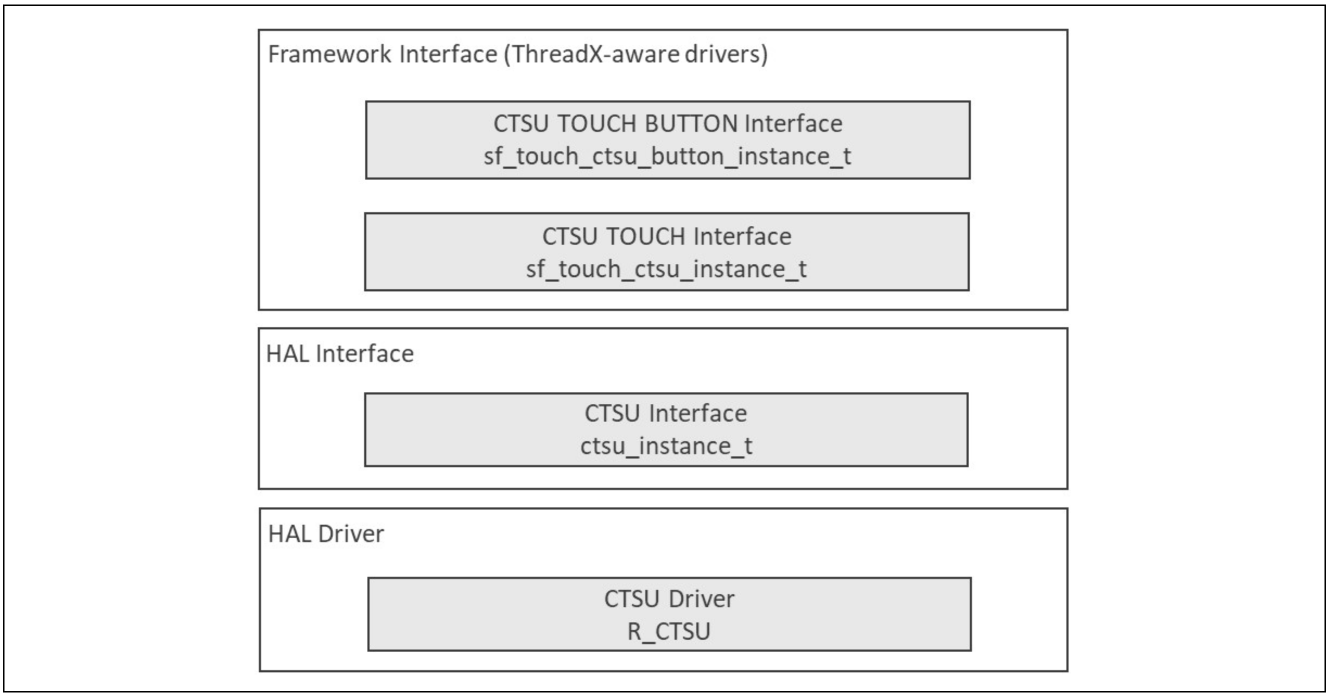

Figure 6. SSP CTSU Framework

Three SSP frameworks are used in the software projects. See the *SSP User's Manual* and the Button and Slider framework module guide (link provided in section 5, References) for capacitive touch API usage.

- Capacitive Touch Button Framework
- Capacitive Touch Slider/Wheel Framework
- Capacitive Touch Framework

2.1 Capacitive Touch Button Framework

- The Capacitive Touch Button Framework allows you to configure many buttons, debounce settings, and a call back function to allow you to take actions on touch events.
- The Capacitive Touch Button Framework is used to interpret the CTSU data for all the buttons that are present in the system. It also initializes the Capacitive Touch Framework layer.
- The Capacitive Touch Button Framework registers a callback with the Capacitive Touch Framework layer which is called each time processed data is available. The Capacitive Touch Button Framework then uses this processed data to perform de-bounce and to determine which of the configured events (Press, Release, and so on) are valid for each button.
- Then the Framework calls your callback for each button in the order that they are present in the button configuration table.
- This framework layer uses the Capacitive Touch Framework layer to implement a button interface. Using this Button Framework, you can configure and use multiple buttons with the configuration structure generated from CTW for Synergy. An action on each button results in a callback with an argument indicating the button ID and event type.

2.2 Capacitive Touch Slider/Wheel Framework

- The Capacitive Touch Slider Framework is a generic API implemented on **sf_touch_ctsu_slider** for Capacitive Touch Slider and Wheel applications using the ThreadX® RTOS.
- The Capacitive Touch Slider Framework requires the Capacitive Touch Framework and the CTSU Driver modules on the Synergy MCU. The Capacitive Touch Slider Framework is used to interpret the CTSU

data for all the slider configurations initialized by the system. It also initializes the Capacitive Touch Framework layer.

- The Capacitive Touch Slider Framework registers a callback with the Capacitive Touch Framework layer, which is called each time processed data is available.
- The Slider Framework uses this data (raw values) to determine if a touch or release occurred and if so, where it occurred. If there is a state change, the Framework calls the callback for each slider/wheel, in the order they are present in the slider/wheel configuration table, with the event and position.
- The Slider Framework executes the callback at the update rate (**sf_touch_ctsu** configuration **update_hz**) between the touch and release events.

2.2.1 Wheel and Slider Position Reporting

The slider position reading from the SSP is linearly distributed on the slider capacitive touch component from 0 to 500. The position reading from the Capacitive Touch Callback function is 0-500 and evenly distributed along the slider. The position of 0 reading is the first sensor layout in the tuning process and the reading increases following the slider orientation chosen in the tuning process.

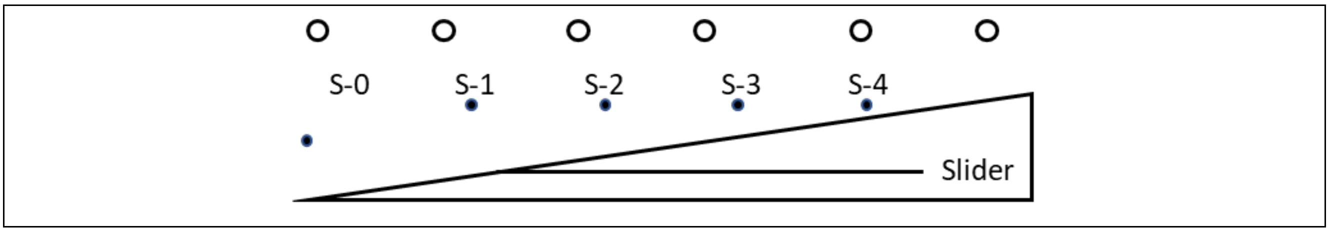

Figure 7. Slider Touch Sensor Layout and Position

For both the S124 and S3A7 MCUs sample projects used in this application project, the slider is defined to be a left to right slider. For the S124 MCU, the lineup for the sensors from left to right is TS5, TS4, TS1, TS0, TS2. For the S3A7 MCU, the lineup for the sensors from left to right is TS27, TS29, TS22, TS26, TS20.

A wheel shape is similar in concept to the slider, but instead of a linear interpretation, the software assumes it as a circle with a layout between 0 and 360 degrees. The position reading from the Capacitive Touch Callback function is 0-360 and evenly distributed along the wheel. The position of 0 reading is the first sensor layout in the tuning process and the reading increase direction follows the direction chosen in tuning process.

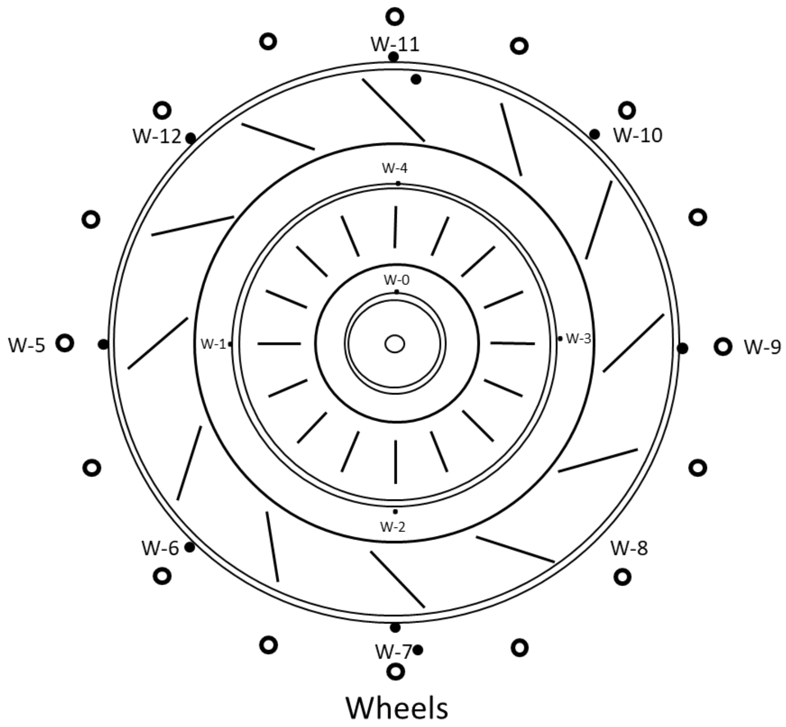

Figure 8. Wheel Touch Sensor Layout and Position

For both the S124 and S3A7 MCU sample projects used in this application project, all wheel definitions are clockwise.

- With S124 outer wheel, the lineup of the sensor is TS22, TS25, TS31, TS28, TS08, TS17, TS18, TS19.
- With S124 inner wheel, the lineup of the sensor is TS24, TS30, TS27, TS20.
- With S3A7 outer wheel, the lineup of the sensors is TS08, TS04, TS01, TS00, TS21, TS35, TS34, TS13.
- With S3A7 inner wheel, the lineup of the sensor is TS10, TS05, TS32, TS12.

2.3 Capacitive Touch Framework

The Capacitive Touch Framework is a ThreadX-aware CTSU interface that is used to drive the CTSU HAL driver. It can be used to run the CTSU hardware and read back the results of the scans.

The Capacitive Touch Framework Interface creates a private thread that drives a hardware scan of a capacitive touch panel and updates the panel at a periodic rate.

The Capacitive Touch Framework reads the scanned results using the HAL layer CTSU driver. When a scan is completed, the callback registered by the application layer is invoked. If multiple upper layers are using this framework (for example: button, slider, wheel), this layer invokes the callbacks for reach of those layers in the order that they initialized in this layer.

2.4 CTSU HAL Driver

The CTSU Driver is used to initialize the CTSU peripheral to detect a change in capacitance on any of the configured (and enabled) channels, perform requisite filtering, and generate a variety of data that can be used by higher level widget layers like buttons, wheel, and sliders.

To support the different types of data required by these layers, the implementation provides a `Read()` function that allows upper level layers to read different types of processed data based on their need.

The driver also provides a callback when each scan is complete and when new processing data is available. These callbacks can be used by upper layers to read the data.

The CTSU Driver allows you to configure the CTSU channels for all the supported operation modes including Mutual and Self-Capacitance.

The driver scans the configured channels, moves the data using the DTC, performs filtering, drift compensation, and auto-tuning, and notifies you via a callback once each iteration is completed.

The driver can only support one configuration at a time, but you can reopen the driver with multiple channel configurations as required by the application.

3. Application Software Projects

Table 2 lists the software projects included in this application project. All the projects support Windows10/Windows7 with e² studio and IAR EW for Synergy, unless otherwise indicated in the table.

Table 2. Application Project Summary

Software Projects	Project Description
AE_CAP1_S3A7_BWS_App	S3A7 application project for the AE-CAP1-BWS 3 buttons, 2 wheels and 1 slider LED indication on the AE-CAP1-BWS board Communication with CTW for Synergy through USB
AE_CAP1_S3A7_ST_App	S3A7 application project for the AE-CAP1-SC 30 buttons LED indication on the AE-CAP1-SC Communication with CTW for Synergy through USB
AE_CAP1_S124_BWS_App	S124 application project for the AE-CAP1-BWS 3 buttons, 2 wheels and 1 slider LED indication on the AE-CAP1-BWS
AE_CAP1_S124_BWS_UART_Monitor_App	S124 application project for the AE-CAP1-BWS 3 buttons, 2 wheels and 1 slider LED indication on the AE-CAP1-BWS Communication with CTW for Synergy through UART
AE_CAP1_S124_ST_UART_Monitor_App	S124 application project for the AE-CAP1-SC 28 buttons LED indication on the AE-CAP1-SC Communication with CTW for Synergy through UART

3.1 Build and Download the Projects

The included `Self-Capacitive Touch Source Code.zip` file contains the complete project. Use the following section to build the projects.

3.1.1 e² studio

1. Unzip AE_CAP1_BSP.zip and place the two BSP files: `Renesas.Synergy_board_s3a7_ae_cap1.1.7.0.pack` and `Renesas.Synergy_Board_s124_ae_cap1.1.7.0.pack` in your e² studio installation folder \<your e2_studio folder<\internal\projectgen\arm\Packs.
2. Refer to the *Renesas Synergy™ Project Import Guide* (r11an0023eu0121-synergy-ssp-import-guide.pdf), included in this package) for instructions on importing the project into e² studio or IAR EW for Synergy and building/running the application. The following file structure should appear after importing.

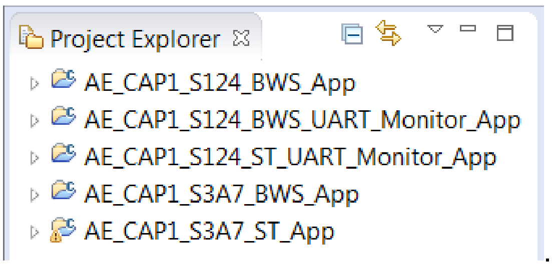

Figure 9. Imported Files in e² studio

3. The sample projects are shipped with proper launch files to help with the debugging process.

3.1.2 IAR EW for Synergy

1. Unzip `AE_CAP1_BSP.zip` and place the two BSP files: `Renesas.Synergy_board_s3a7_ae_cap1.1.7.0.pack` and `Renesas.Synergy_Board_s124_ae_cap1.1.7.0.pack` (in case of SSP 1.7.0) to your IAR EW for Synergy SSC folder `\<your SSC folder>\internal\projectgen\arm\Packs`.
2. The SSP needs to be installed in the SSC before opening the IAR EW Workspace. After opening the IAR EW for Synergy Workspace, go **to Renesas Synergy > Settings** and point IAR EW for Synergy to the location where you installed SSC v7.3.0, as well as the associated SSP license file.

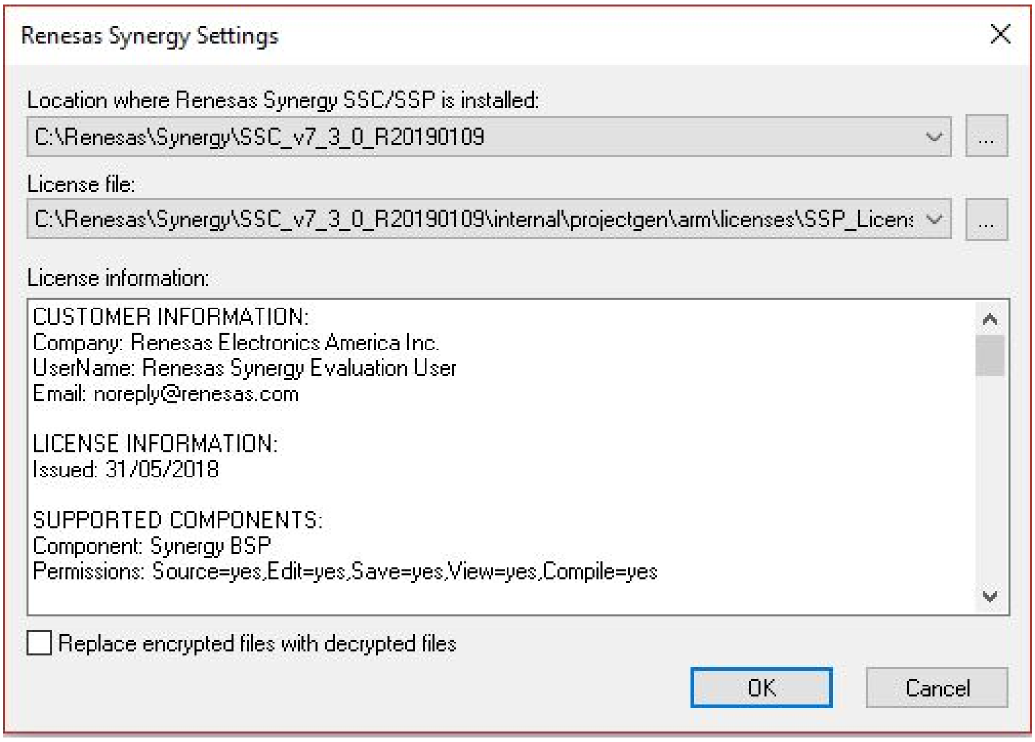

Figure 10. SSC Location

3. Go to **Project->Rebuild All** in the IAR EW for Synergy. Click the green arrow to download and debug the project.

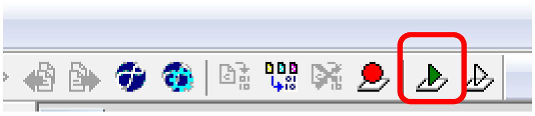

Figure 11. Download and Debug

3.2 Threads in the Projects

All sample projects include a CTSU Thread for the Capacitive Touch Application. A USB communication thread for S3A7 MCU and an UART thread for S124 are used to communicate with the CTW for Synergy.

3.3 Capacitive Touch Framework and Driver Settings

3.3.1 S124 Capacitive Touch Settings

3.3.1.1 Capacitive Touch Button Framework Settings

Figure 12 shows some common parameter settings for the Capacitive Touch Button Framework in the S124 MCU Self-Capacitive Software Projects.

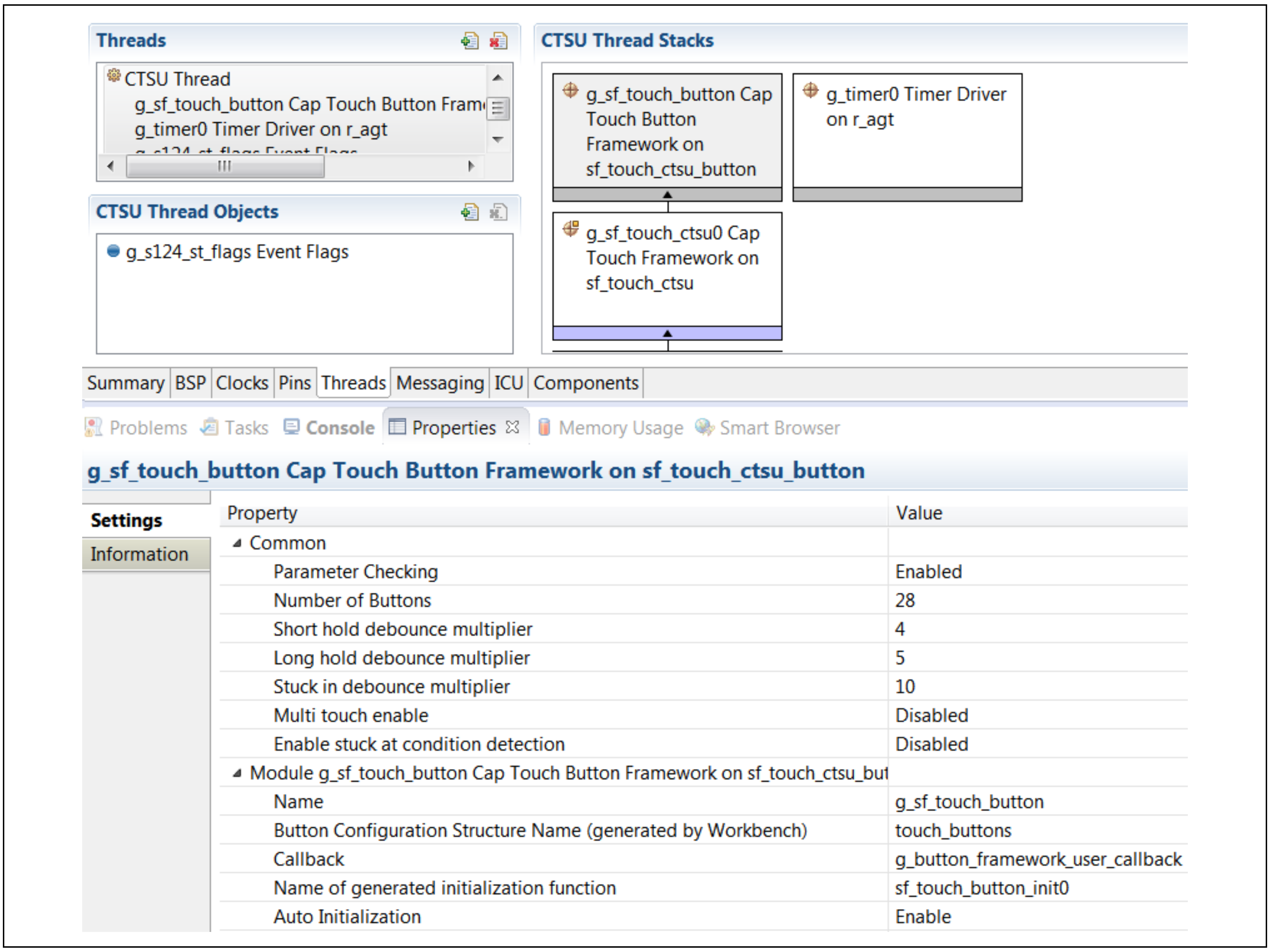

Figure 12. Capacitive Touch Button Framework Settings for S124 MCU in the Self-Capacitive Application

- Setting for property **Number of Buttons** varies based the project:
  - BWS application, **Number of Buttons** is 3.
  - SC application, **Number of Buttons** is 28.
- You will need to implement the callback function `g_button_framework_user_callback` in the application code. See Figure 19 for a reference implementation.

Note: The debounce multiplier is internally multiplied by 7 in the application.

3.3.1.2 Capacitive Touch Slider Framework Settings

Figure 13 shows some common parameter settings for the Capacitive Touch Slider/Wheel Framework in the self-capacitive S124 MCU software projects.

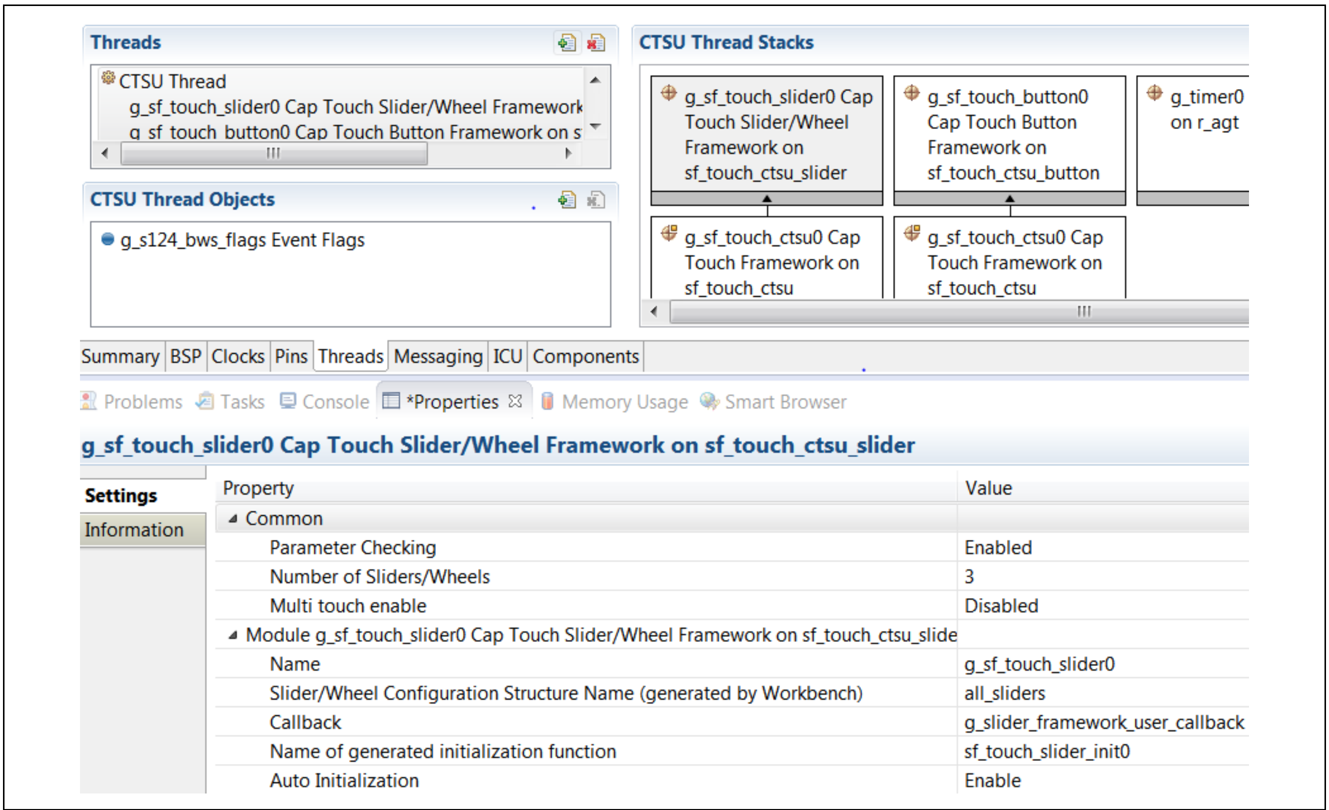

Figure 13. Capacitive Touch Slider/Wheel Framework Settings for the S124 MCU

- Property **Number of Sliders/Wheels** is set to 3 as there are one slider and two wheels on the AE-CAP1-BWS.
- You will need to implement the callback function `g_slider_framework_user_callback` in the application code.

3.3.1.3 Capacitive Touch Framework Settings

Figure 14 shows some common parameter settings for the Capacitive Touch Framework in the S124 MCU software projects.

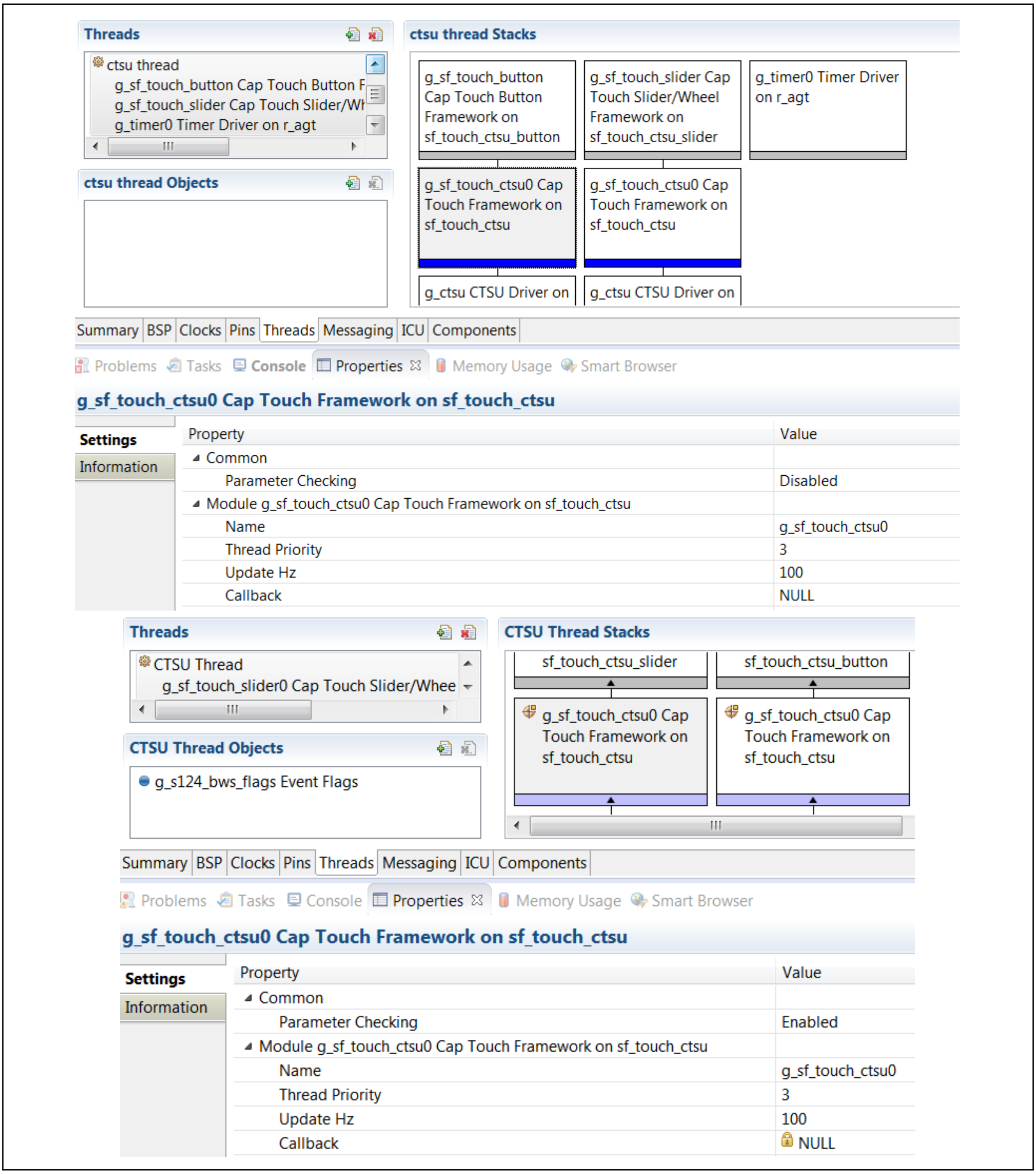

Figure 14. Capacitive Touch Framework Settings for both S124 and S3A7 Synergy MCUs

- You can set proper thread priority in their software applications.
- **Update Hz** needs to be less than the RTOS tick rate that is set to 100 Hz in the application.

3.3.1.4 CTSU HAL Driver Settings

Figure 15 shows some common parameter settings for the CTSU HAL Driver Settings in the S124 MCU software projects.

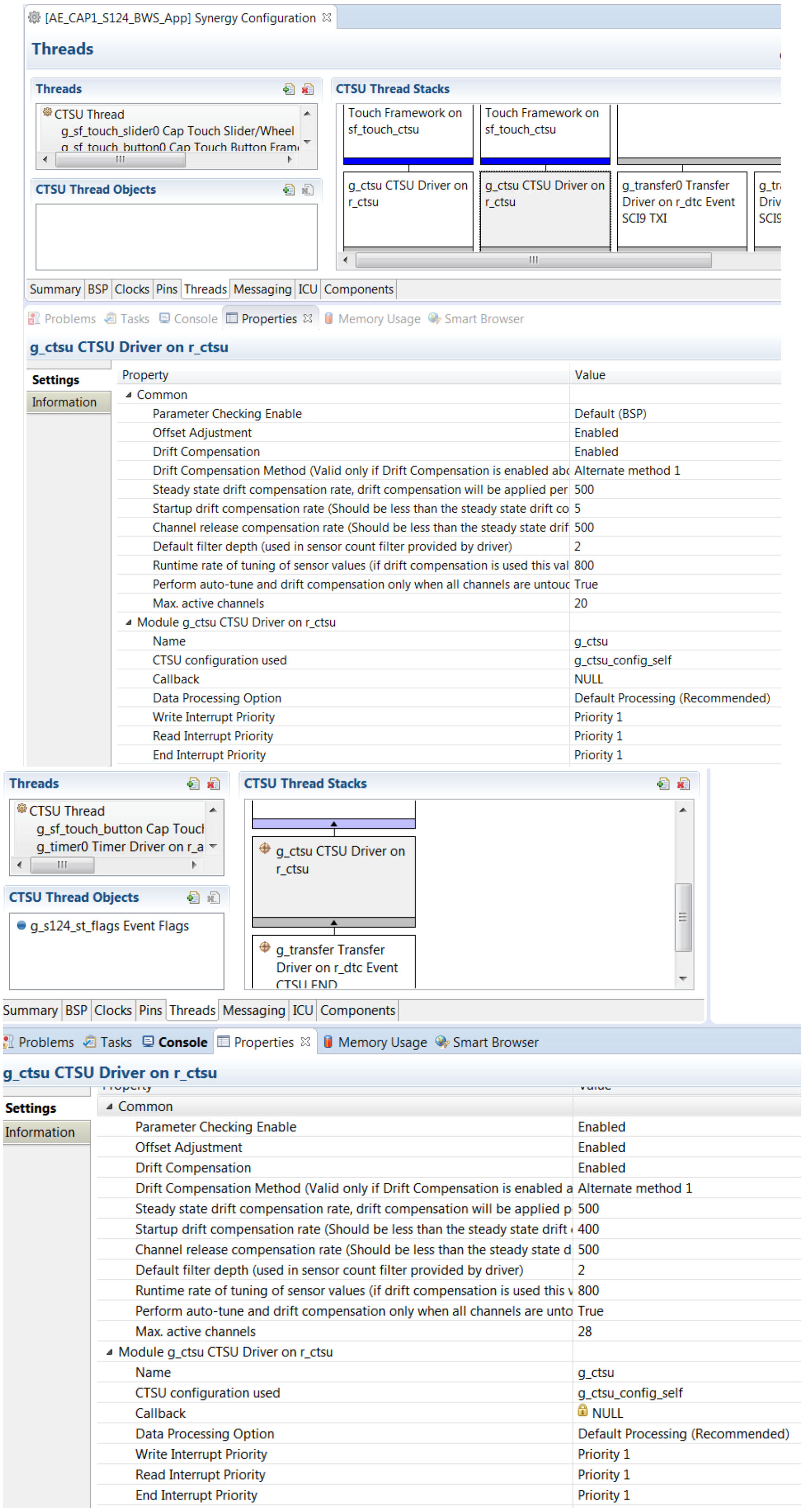

Figure 15. CTSU HAL Driver Settings for the S124 Synergy MCU

- Settings for property **Max. active channels** varies based the project:
  - BWS application, **Max. active channels** is 20
  - SC application, **Max. active channels** is 28
- Refer to the *SSP User's Manual* to understand the meaning for the properties.
- Notice that the CTSU configuration used is set to `g_ctsu_config_self`. This is name of the data structure generated from the tuning process which holds the self-capacitance tuning data.
- Startup drift compensation is changed from the default value of 5 to 400. A setting of 5 with the application project corresponds to a drift compensation rate of 50 ms and a setting of 400 corresponds to a drift compensation rate of 4 seconds.
- You can adjust the **Default Filer Depth** based on the application noise environment. Increase the **Default Filter Depth** increases the noise resistance at a slight cost of processing time.
- You can set the **Write**, **Read**, **End Interrupt Priority** based on their application.

3.3.2 S3A7 MCU Capacitive Touch System Settings

3.3.2.1 Capacitive Touch Button Framework Settings

Figure 16 shows some common parameter settings for the Capacitive Touch Button framework in the S3A7 MCU software projects.

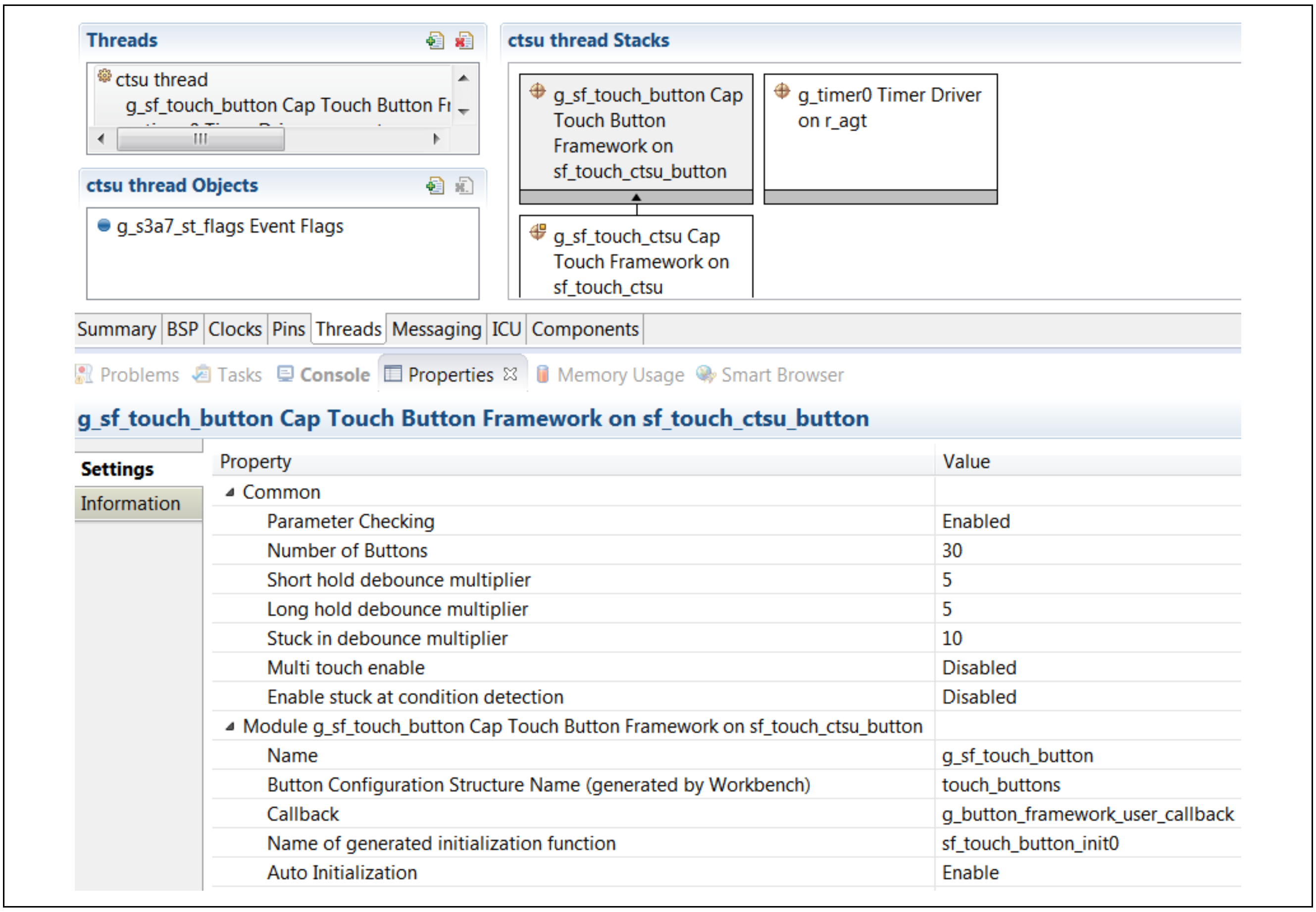

Figure 16. Capacitive Touch Button Framework for S3A7

- Setting for property **Number of Buttons** varies based the project:
  - BWS application, **Number of Buttons** is 3.
  - SC application, **Number of Buttons** is 30.
- You will need to implement the callback function `g_button_framework_user_callback` in the application code.
- Note that the debounce multiplier is internally multiplied by 7 in the application.

3.3.2.2 Capacitive Touch Slider Framework Settings

Figure 17 shows some common parameter settings for the Capacitive Touch Slider/Wheel framework in the S3A7 MCU software projects.

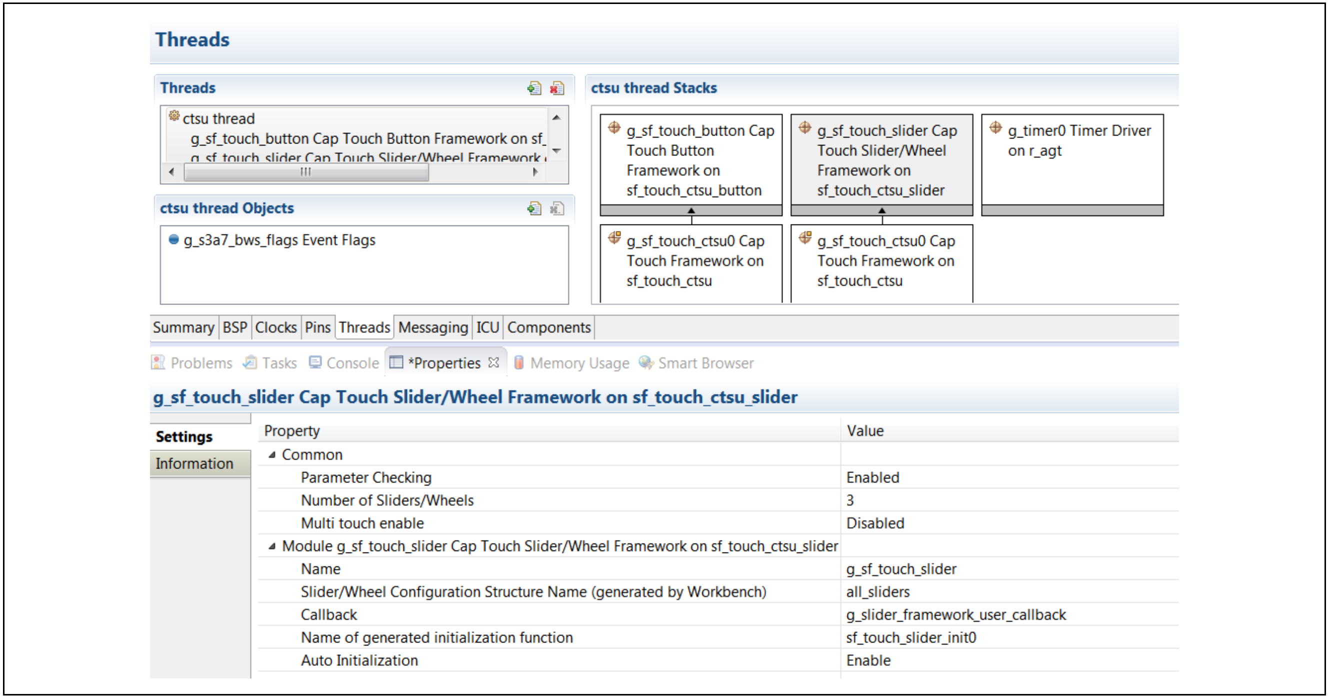

Figure 17. Capacitive Touch Slider/Wheel Framework Settings for S3A7

The Capacitive Touch Framework layer share common settings between the S124 and S3A7 Synergy MCU. See Figure 14 for the relevant settings.

3.3.2.3 CTSU HAL Driver Settings

Figure 18 show some common parameter settings for the CTSU HAL Driver Settings in the S3A7 MCU software projects.

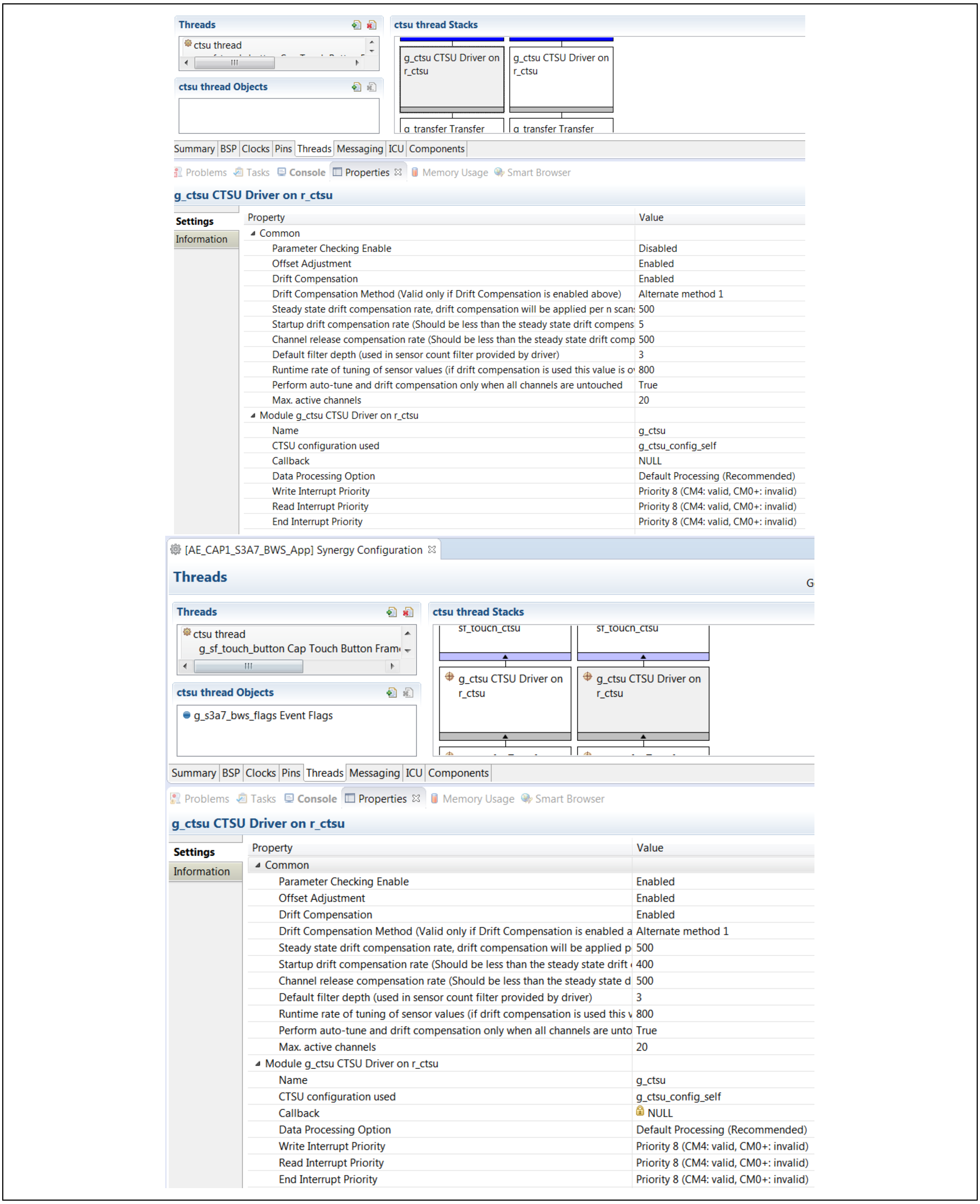

Figure 18. CTSU HAL Driver Settings for S3A7

Settings for property **Max. active channels** varies based the project:

- BWS application, **Max. active channels** is 20
- SC application, **Max. active channels** is 30
  - Refer to the *SSP User's Manual* to understand the meaning for the properties.
  - Notice that the CTSU configuration used is set to `g_ctsu_config_self`. This is the name of the data structure generated from the tuning process that holds the self-capacitance tuning data.
- Startup drift compensation is changed from the default value of 5 to 400. A setting of 5 with the application project corresponds to a drift compensation rate of 50 ms and a setting of 400 corresponds to a drift compensation rate of 4 seconds.
- You can adjust the **Default Filter Depth** based on the application noise environment. Increasing the Default Filter Depth increases the noise resistance at a slight cost of processing time.
- You can set the **Write**, **Read**, **End Interrupt Priority** based on their application.

3.4 Capacitive Touch Application User Callbacks

3.4.1 Button Callbacks

The application project uses two steps to handle the button callback.

Step 1: Set the button callback event:

```
void g_button_framework_user_callback(sf_touch_ctsu_button_callback_args_t * p_args)
{
    button_callback_args = p_args;
    tx_event_flags_set(&g_s3a7_st_flags, BUTTON_EVENT, TX_OR);

}
```

Figure 19. Button Framework User Callbacks – Set Event

Step 2: Pick up the button event in the CTSU thread entry function and process the events in `CB_Self_Button()`.

```
while (true)
{
     status = tx_event_flags_get(&g_s3a7_st_flags, ALL_EVENTS, TX_OR_CLEAR, &events_got, TX_WAIT_FOREVER);
    if (status == TX_SUCCESS)
    {
        switch(events_got)
        {
            case BUTTON_EVENT:
                CB_Self_Button();
                break;
            case TIMER_EVENT:
                handle_timer();
                break;
            default:
                break;
        }
    }
    tx_thread_sleep (10);
}
```

Figure 20. Button Events Processing

Figure 21 shows the button event handling in this application project.

```
/* handle the button callback events */

static void CB_Self_Button (void)
{
    switch(button_callback_args->event)
    {
        case TOUCH_BUTTON_STATE_INITIAL:
            break;
        case TOUCH_BUTTON_STATE_PRESSED:
            light_up_correspondingLED();
            break;
        case TOUCH_BUTTON_STATE_LONG_HOLD:
            break;
        case TOUCH_BUTTON_STATE_SHORT_HOLD:
            break;
        case TOUCH_BUTTON_STATE_STUCK:

            break;
        case TOUCH_BUTTON_STATE_RELEASED:
            offLEDs();
            break;
        case TOUCH_BUTTON_STATE_CLOSING:
            break;
        case TOUCH_BUTTON_STATE_DISABLED:
            break;
        case TOUCH_BUTTON_STATE_MULTI_TOUCH:
            break;
        default:
            break;

    }
}
```

Figure 21. Button Events

3.4.2 Slider/Wheel User Callbacks

The application project uses two steps to handle the slider and wheel event callback.

Step 1: Set the slider wheel event.

```
/* slider framework callback */
void g_slider_framework_user_callback(sf_touch_ctsu_slider_callback_args_t * p_args)
{
    slider_wheel_callback_args = p_args;
    tx_event_flags_set(&g_s3a7_bws_flags, SLIDER_WHEEL_EVENT, TX_OR);
}
```

Figure 22. Slider User Callback – Set Event

Step 2: Pick up the slider wheel event in the CTSU thread entry function and process the events in `handle_slider_wheel_touch()`.

```
        while (true)
        {
            status = tx_event_flags_get(&g_s3a7_bws_flags, ALL_EVENTS, TX_OR_CLEAR, &events_got, TX_WAIT_FOREVER);
            if (status == TX_SUCCESS)
            {
               switch(events_got){
                   case SLIDER_WHEEL_EVENT:
                       handle_slider_wheel_touch();
                       break;
                   case BUTTON_EVENT:
                       CB_Self_Button ();
                       break;
                   case TIMER_EVENT:
                       handle_timer();
                       break;
                   default:
                       break;
               }

            }
            tx_thread_sleep (1);
        }
```

Figure 23. Process the Slider and Wheel Events

Notice that the `g_slider_framework_user_callback()` handles both the Slider callback and the Wheel callback.

See Figure 24 to view the possible events generated from the slider callback. Wheel callback includes the similar events.

```
void CB_Self_Slider_0(sf_touch_ctsu_slider_callback_args_t * p_args);
void CB_Self_Slider_0(sf_touch_ctsu_slider_callback_args_t * p_args)
{
    extern slider_info_t g_sliderInfo[];

    switch(p_args->event)
    {
        case SF_TOUCH_CTSU_SLIDER_STATE_INITIALIZED:

            break;
        case SF_TOUCH_CTSU_SLIDER_STATE_TOUCHED:
        case SF_TOUCH_CTSU_SLIDER_STATE_TOUCHED |
SF_TOUCH_CTSU_SLIDER_STATE_MULTI_TOUCH:
            if (0 != (p_args->event &
(uint32_t)SF_TOUCH_CTSU_SLIDER_STATE_MULTI_TOUCH))
            {

            }
            else
            {
                g_sliderInfo[0].value = (uint16_t)p_args->current_position;
                slider0_led_response(p_args);
            }
            break;
        case SF_TOUCH_CTSU_SLIDER_STATE_HELD:
        case SF_TOUCH_CTSU_SLIDER_STATE_HELD |
SF_TOUCH_CTSU_SLIDER_STATE_MULTI_TOUCH:
            if (0 != (p_args->event &
(uint32_t)SF_TOUCH_CTSU_SLIDER_STATE_MULTI_TOUCH))
            {

            }
            else
            {
                g_sliderInfo[0].value = (uint16_t)p_args->current_position;
                slider0_led_response(p_args);
            }
            break;
        case SF_TOUCH_CTSU_SLIDER_STATE_RELEASED:
            offLEDs ();

            break;
        case SF_TOUCH_CTSU_SLIDER_STATE_CLOSED:

            break;
        case SF_TOUCH_CTSU_SLIDER_STATE_DISABLED:

            break;
        default:
            break;
    }
}
```

For communication with CTW for Synergy

Figure 24. Slider Events Processing

3.4.3 Capacitive Touch Application Automatic Error Checking

```
void g_sf_touch_button_err_callback(void * p_instance, void * p_data)
{
    ssp_err_t ssp_err_g_sf_touch_button;
    SSP_PARAMETER_NOT_USED (p_instance);
    ssp_err_g_sf_touch_button = *((ssp_err_t*)p_data);

    switch(ssp_err_g_sf_touch_button){
        case SSP_ERR_CTSU_OFFSET_ADJUSTMENT_FAILED:
            Blink_Red_Led = 1;
            break;
        case SSP_ERR_CTSU_SC_OVERFLOW:
            break;
        case SSP_ERR_CTSU_RC_OVERFLOW:
            break;
        case SSP_ERR_CTSU_SAFETY_CHECK_FAILED:
            break;
        default:
            break;

    }
}
```

Figure 25. Automatic Error Checking

You can add more error handling based on the other available events.

3.5 USB Communication with CTW for Synergy

The S3A7 MCU software uses USB to communicate with CTW for Synergy from J9. To work with Windows 10, there is no need to install a USB driver. USB Serial Device enumeration happens with sample projects.

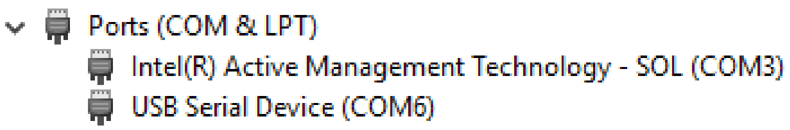

Figure 26. USB Serial Device COM Port on Windows 10

To work with Windows 7, you need to install the USB CDC/ACM device driver. The USB driver is attached with this application project (`Windows_USB_serial_driver.zip`). Unzip it to folder `\Windows_USB_serial_driver`.

1. When you download and run the application, the CDC/ACM device shows up in the Device Manager of your PC under the Universal Serial Bus Controller group as UNKNOWN DEVICE.
2. Right click on this device and select **Update Driver Software**.
3. When prompted for the location of the drivers, browse to the location `\Windows_USB_serial_driver` you created previously.
4. Once the driver is updated a new COM device shows up in the Device Manager.

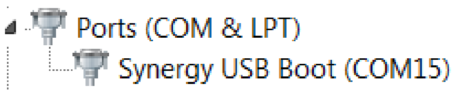

Figure 27. Communications Port on Windows 7

4. Operation of the Application Projects

4.1 Project AE_CAP1_S124_BWS_App

Use the following hardware settings to bring up the AE-CAP1-S124 and AE-CAP1-BWS.

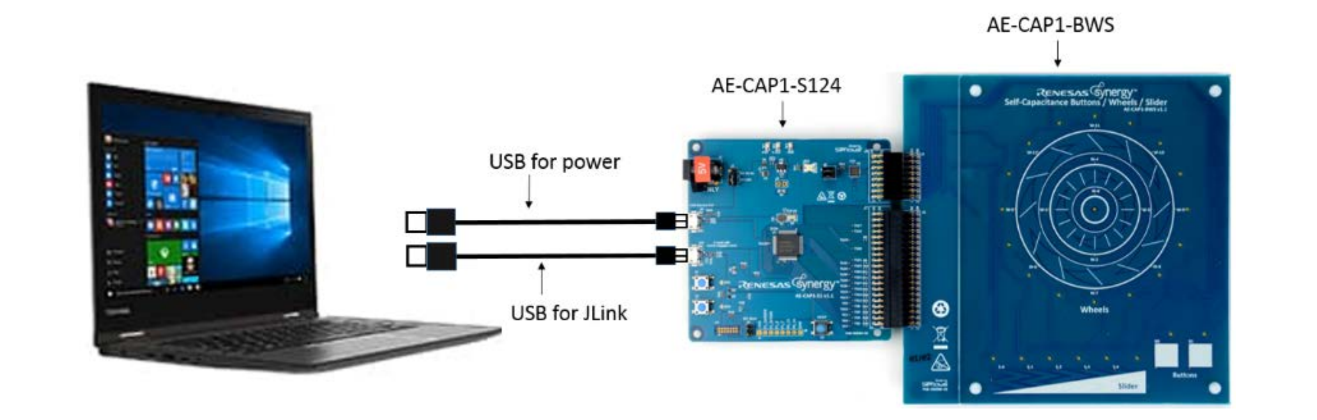

Figure 28. Project AE_CAP1_S124_BWS_App Hardware Setup

Leave jumper J4 open, oriente jumper J2 towards the USB Device and Install jumpers for J5 as shown in Figure 29.

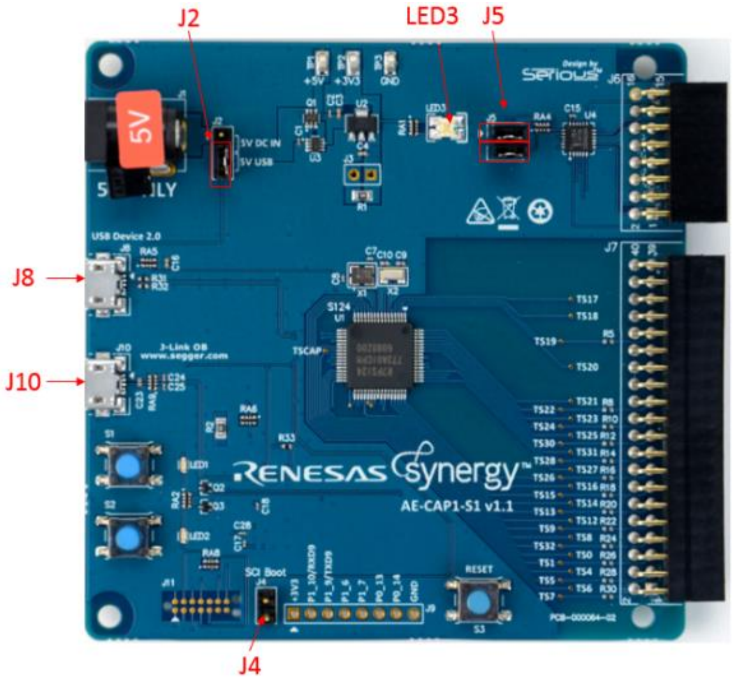

Figure 29. AE-CAP1-S1 Setting for Application Board LED Control

1. Connect AE-CAP1-S1 from J8 to PC using the USB cable included to provide power to the system. LED3 turns on to indicate a good power connection.
2. Connect AE-CAP1-S124 from J10 to PC using the other USB cable included to provide the J-Link connection.
3. Build and download the project.

4.1.1 Operations for Project AE_CAP1_S124_BWS_App

Note: If you are running the system without connecting to the debugger, it is recommended to power on the system and then press the **Reset** button to start the system.

Upon running after downloading, following events happen in the system in sequence:

1. The system performs an auto tuning test on the Buttons/Wheels/Slider board (about 2 seconds).
2. When this auto tuning is finished, LED1 (Red) and LED2 (Green) on the AE-CAP1-S1 board blinks five times. **Please do not touch the board before LED1 and LED2 start to blink.**
3. If auto tuning is successful, LED2 (Green) continues to blink and the board is ready to accept the user touch. Use the following LED signaling scheme on the AE-CAP1-BWS Application board:
   A. Touching any of the button illuminates the LED associated with that button. Try touch buttons B0 and B1 and W-0 and see the corresponding LED light up.
   B. Sliding a finger up and down; the slider area illuminates the LEDs along the top of the slider.
   C. Move your finger along the inner wheel and see the LEDs follow your finger movement.
   D. Move your fingers along the outer wheel and see the LEDs follow your finger movement.
4. If auto tuning fails, LED1 (Red) alone blinks to indicate a hardware issue. If you get a hardware issue:
   A. Press the **Reset** button to restart the system.
   B. Make sure you are using the correct target board AE-CAP1-BWS.
   C. Make sure the boards you are using are placed on a non-conductive surface.

4.2 Project AE_CAP1_S124_BWS_UART_Monitor_App

Use the following hardware settings to bring up the AE-CAP1-S124 and AE-CAP1-BWS.

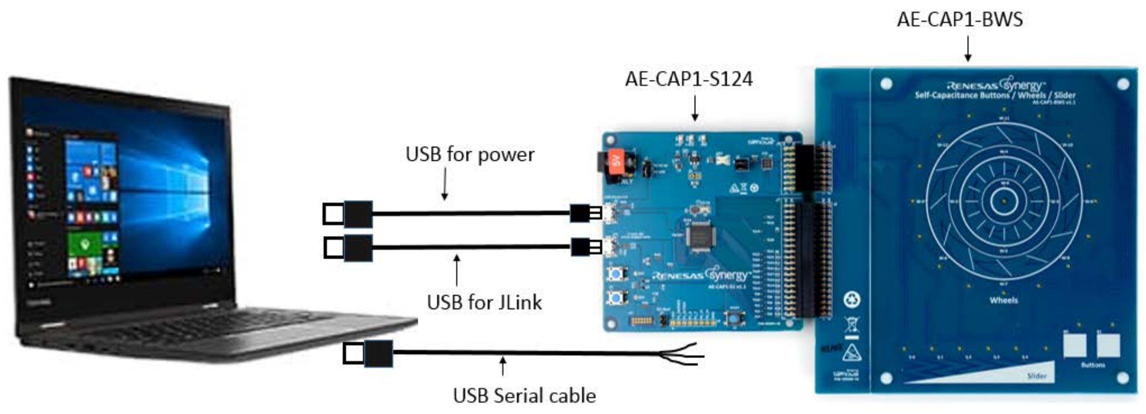

Figure 30. Project AE_CAP1_S124_BWS_UART_Monitor_App Hardware Setup

1. On AE-CAP1-S1, leave jumper J4 open, orient jumper J2 towards the USB Device. **Un-install jumpers from J5 (leave J5 open)**. See Figure 29 for locations of J4, J2, and J5.
2. Connect AE-CAP1-S1 from J8 to PC using the USB cable included to provide power to the system.
3. Connect AE-CAP1-S124 from J10 to PC using the other USB cable included to provide J-Link connection.
4. Connect AE-CAP1-S124 J9 through a Serial-to-USB converter, as shown in Figure 31 to PC. **The Serial-to-USB conversion cable is not included in the kit**.

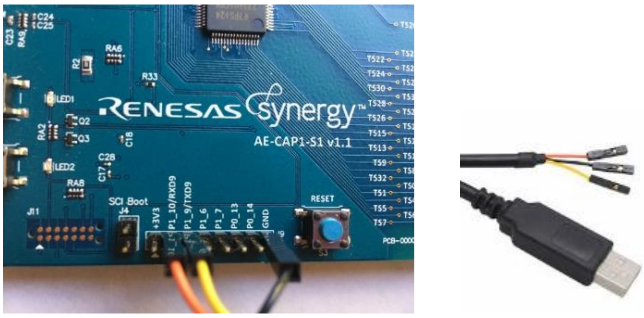

Figure 31. Serial Connection

5. Build and download the project.

4.2.1 Operation for Project AE_CAP1_S124_BWS_UART_Monitoring App

4.2.1.1 Standalone Mode

Note: If you are running the system without connecting to the debugger, it is recommended to power on the system and then press the **Reset** button to start the system.

Upon running, events happen in the system in the following sequence:

1. The system performs an auto tuning test on the Buttons/Wheels/Slider board (about 2 seconds).
2. When this auto tuning is finished, LED1 (Red) and LED2 (Green) on the AE-CAP1-S1 board blinks five times. **Please do not touch board before LED1 and LED2 start to blink.**
3. If auto tuning is successful, LED2 (Green) continues to blink and the board is ready to accept your touch sensing. **Note that the LED on the AE-CAP1-BWS board is not active with the demo code.**
4. If auto tuning fails, LED1 (Red) alone blinks to indicate a hardware issue. If you get a hardware issue, reference the debugging tips listed in item 4 of Section 4.1.1 for trouble shooting.

4.2.1.2 Communication with CTW for Synergy

For communication with CTW for Synergy, set UART baud rate to 115200 to establish the communication.

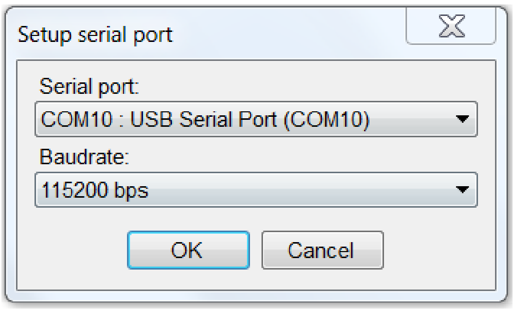

Figure 32. UART Baudrate Setting

See Table 3 to map the Silk Screen button marking to the touch sensor channel assignment.

Table 3. S124 AE-CAP1-BWS Silk Screen Marking and Mapping to Sensor Touch Channels

CTW for Synergy Assignment	AE-CAP1-BWS Mark
TS06	B0
TS07	B1

See the *AE-CAP1 Quick Start Guide* to review how to communicate with the CTW for Synergy.

4.3 Project AE_CAP1_S124_ST_UART_Monitoring_App

Figure 33 shows the hardware settings to bring up the AE-CAP1-S124 and AE-CAP1-SC.

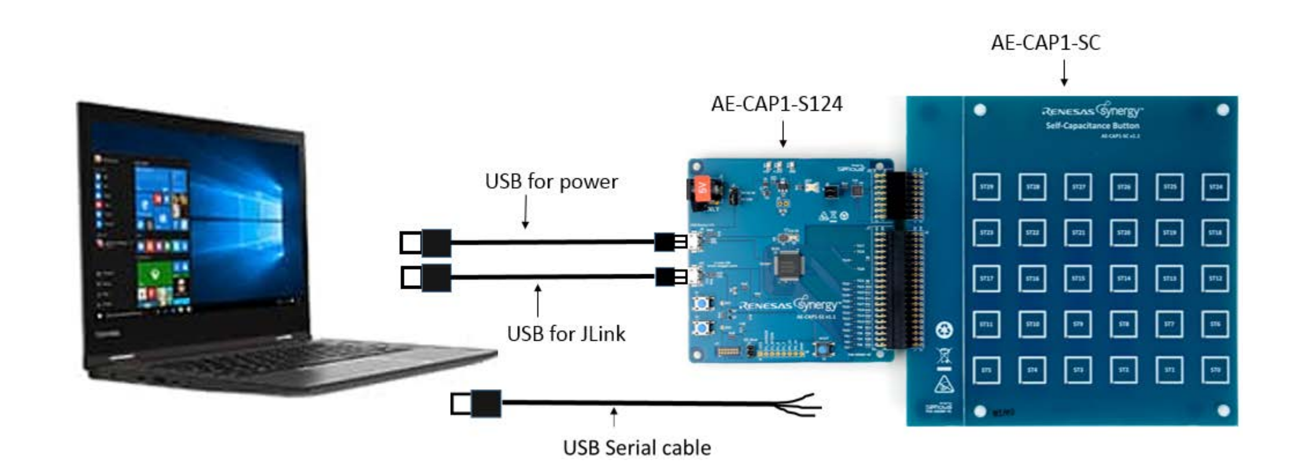

Figure 33. Project AE_CAP1_S124_ST_UART_Monitoring_App Setup

1. On AE-CAP1-S1, leave jumper J4 open, orient jumper J2 towards USB Device and **un-install jumpers from J5 (leave J5 open)**. See Figure 29 for locations of J4, J2, and J5.
2. Connect AE-CAP1-S1 from J8 to PC using the USB cable included to provide power to the system.
3. Connect AE-CAP1-S124 from J10 to PC using the other USB cable included to provide the J-Link connection.
4. Build and download the project.

4.3.1 Operation for Project AE_CAP1_ST_UART_Monitoring_App

4.3.1.1 Standalone Mode

Note: If you are running the system without connecting to the debugger, it is recommended to power on the system then press the **Reset** button to start the system.

Upon running, events happen in the system in the following sequence:

1. The system performs an auto tuning test on the self-capacitance button board (about 2 seconds).
2. When this auto tuning is finished, LED1 (Red) and LED2 (Green) on the AE-CAP1-S1 board blinks five times. **Please do not touch board before LED1 and LED2 start to blink.**
3. If auto tuning is successful, LED2 (Green) continues to blink and the board is ready to accept your touch sensing. Upon pressing, the LED1 (Red) lights up and turn off upon pressing release.

   Note: S124 implements only 28 touch buttons. Button ST24 and ST25 on the upper right corner are not active.

4. If auto tuning fails, LED1 (Red) alone blinks to indicate a hardware issue. If you get a hardware issue, reference the debugging tips listed in item 4, Section 4.1.1 for trouble shooting.

4.3.1.2 Communication with CTW for Synergy

For communication with CTW for Synergy, set UART baud rate to 115200 to establish the communication:

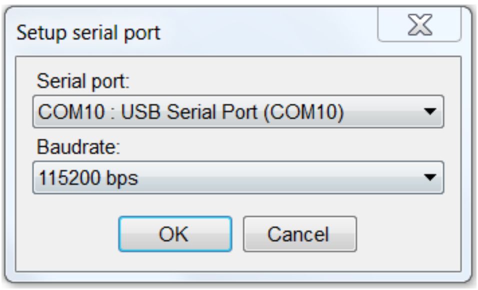

Figure 34. UART Baudrate Setting

See Table 4 to map the marking on the AE-CAP1-SC to the touch sensor channel on the S124 MCU.

Table 4. S124 AE-CAP1-SC Board Silk Screen to Sensor Channel Mapping

CTW for Synergy Assignment	AE-CAP1-SC Mark	CTW for Synergy Assignment	AE-CAP1-MC Mark	CTW for Synergy Assignment	AE-CAP1-MC Mark
TS00	ST5	TS13	ST10	TS23	ST21
TS01	ST4	TS14	ST11	TS24	ST20
TS02	ST6	TS15	ST12	TS25	ST19
TS04	ST3	TS16	ST13	TS26	ST14
TS05	ST2	TS17	ST29	TS27	ST15
TS06	ST1	TS18	ST28	TS28	ST16
TS07	ST0	TS19	ST27	TS30	ST18
TS08	ST7	TS20	ST26	TS31	ST17
TS09	ST8	TS21	ST23		
TS12	ST9	TS22	ST22		

See the *AE-CAP1 Quick Start Guide* to review how to communicate with CTW for Synergy.

4.4 Project AE_CAP1_S3_BWS_App

Follow the hardware settings in Figure 35 to bring up AE-CAP1-S3A7 and AE-CAP1-BWS.

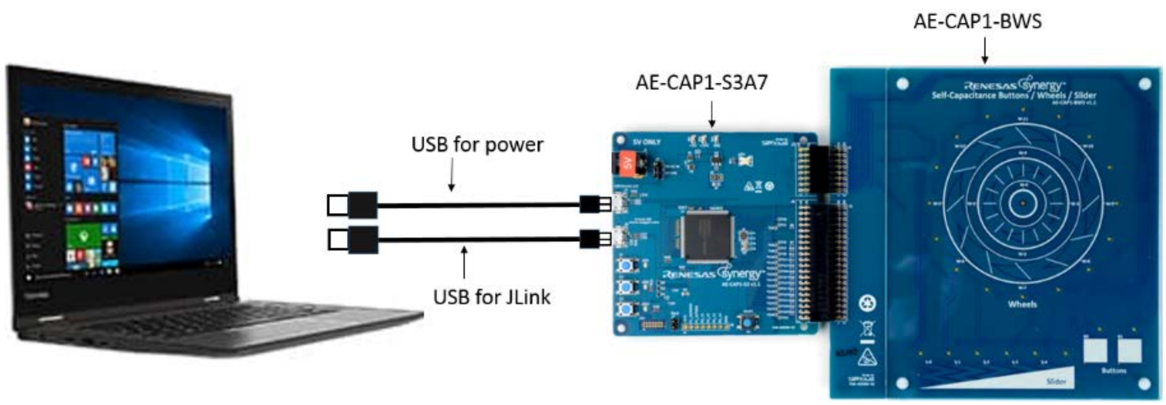

Figure 35. Project AE_CAP1_S3A7_BWS_App Hardware Setup

Leave jumper J4 open, and orient jumper J2 towards the USB Device.

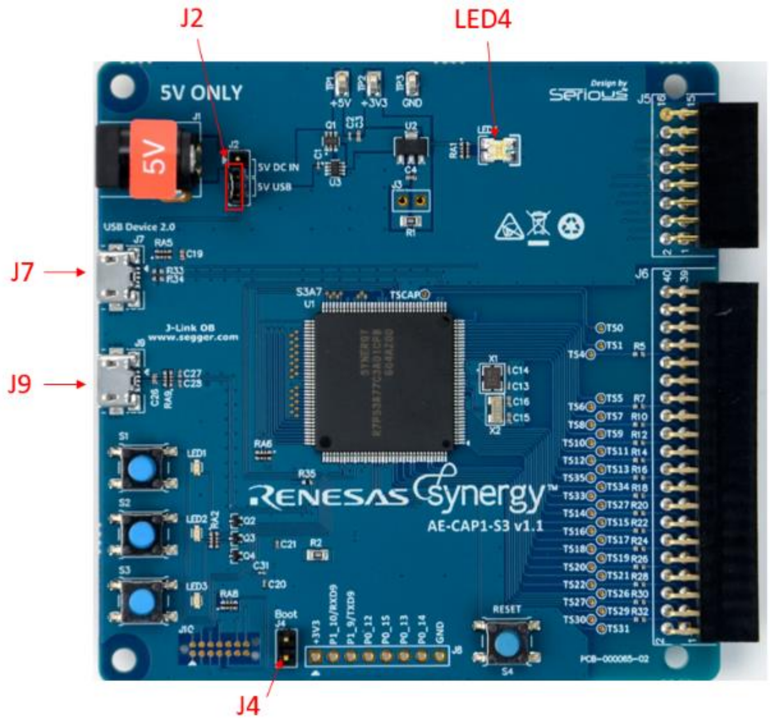

Figure 36. AE-CAP1-S3 Setting

1. Connect AE-CAP1-S3 from J7 to the PC using the USB cable included to provide power to the system.
2. Connect AE-CAP1-S3A7 from J10 to the PC using the other USB cable included to provide J-Link connection.
3. Build and download the project.

4.4.1 Operations for Project AE_CAP1_S3A7_BWS_App

4.4.1.1 Standalone Mode

Note: If you are running the system without connecting to the debugger, it is recommended to power on the system then press the **Reset** button to start the system.

Upon running, events happen in the system in the following sequence:

1. The system performs an auto tuning test on the Buttons/Wheels/Slider board (about 2 seconds).
2. When this auto tuning is finished, LED1 (Red), LED2 (Yellow) and LED3 (Green) on the AE-CAP1-S3 board blinks 5 times. **Do not touch the board before LED1, LED2 and LED3 start to blink.**
3. If auto tuning is successful, LED3 (Green) continues to blink and the board is ready to accept your touch. Below is the LED signaling scheme on the AE-CAP1-BWS Application board:
   A. Touching any of the button illuminates the LED associated with that button. Try touch buttons B0 and B1 and W-0 and see the corresponding LED light up.
   B. Sliding a finger up and down the slider area illuminates the LEDs along the top of the slider.
   C. Your finger along the inner wheel and see the LEDs follow your finger movement.
   D. Move your fingers along the outer wheel and see the LEDs follow your finger movement.
4. If auto tuning fails, LED1 (Red) alone blinks to indicate a hardware issue. If you get a hardware issue, reference the debugging tips listed in item 4, Section 4.1.1 for trouble shooting

4.4.1.2 Communication with CTW for Synergy

See the *AE-CAP1 Quick Start Guide* to review how to communicate with the CTW for Synergy. **Make sure you connect J7 USB Device port to PC when working with this program.** See Table 5 to map the AE-CAP1-BWS silk screen marking to the S3A7 sensor touch channels.

Table 5. S3A7 AE-CAP1-BWS Silk Screen Button Marking to Touch Sensor Channel Mapping

CTW for Synergy Assignment	AE-CAP1-BWS Mark
TS31	B0
TS30	B1

4.5 Project AE_CAP1_S3A7_ST_App

Use the following hardware settings to bring up the AE-CAP1-S3A7 and AE-CAP1-ST:

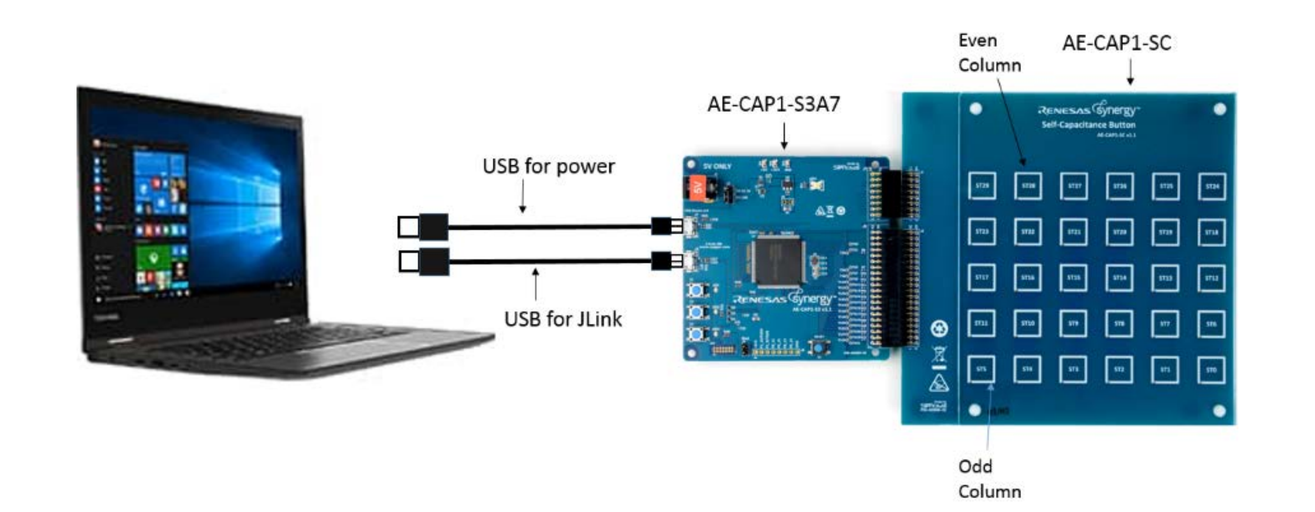

Figure 37. Project AE_CAP1_S3A7_ST_App Hardware Setup

1. See Figure 36, AE-CAP1-S3 setting to setup jumpers on AE-CAP1-S3. Leave jumper J4 open; orient jumper J2 towards the USB Device.
2. Connect AE-CAP1-S3 from J7 to PC using the USB cable included to provide power to the system.
3. Connect AE-CAP1-S3A7 from J9 to PC using the other USB cable included to provide J-Link connection.
4. Build and download the project.

4.6 Operations for Project AE_CAP1_S3A7_ST_App

4.6.1.1 Standalone Mode

Note: If you are running the system without connecting to the debugger, it is recommended to power on the system and then press the **Reset** button to start the system.

Upon running, events happen in the system in the following sequence:

1. The system performs an auto tuning test on the self-capacitance board.
2. When this auto tuning is finished, LED1 (Red), LED2 (Yellow), and LED3 (Green) on the AE-CAP1-S3 board blinks five times. **Do not touch board before LED1, LED2, and LED3 start to blink.**
3. If auto tuning is successful, LED3 (Green) continue to blink and the board is ready to sense user touch.
   A. Upon pressing the buttons on the odd columns of the AE-CAP1-SC board (see Figure 37 for the odd column definition), LED1 (Red) lights up and turns off when the press is released.
   B. Upon pressing the buttons on the even columns of the AE-CAP1-SC board (see Figure 37 for the even column definition), LED2 (Yellow) lights up and turns off when the press is released.
4. If auto tuning fails, LED1 (Red) alone blinks to indicate a hardware issue. If you get a hardware issue, see the debugging tips listed in item 4, Section 4.4.1 for troubleshooting.

4.6.1.2 Communication with CTW for Synergy

See the *AE-CAP1 Quick Start Guide* to setup communication and review how to communicate with the CTW for Synergy. **Make sure you connect J7 USB Device port to PC when working with this program**. See Table 6 to map AE-CAP1-SC silk screen marking to S3A7 MCU sensor touch channels.

Table 6. S3A7 AE-CAP1-SC Silk Screen Marking to Touch Sensor Channel Mapping

CTW for Synergy Assignment	AE-CAP1-SC Mark	CTW for Synergy Assignment	AE-CAP1-MC Mark	CTW for Synergy Assignment	AE-CAP1-MC Mark
TS00	ST29	TS12	ST18	TS22	ST4
TS01	ST28	TS13	ST19	TS26	ST5
TS04	ST27	TS14	ST12	TS27	ST2
TS05	ST26	TS15	ST13	TS29	ST3
TS06	ST24	TS16	ST10	TS30	ST0
TS07	ST25	TS17	ST11	TS31	ST1
TS08	ST22	TS18	ST8	TS32	ST15
TS09	ST23	TS19	ST9	TS33	ST14
TS10	ST20	TS20	ST6	TS34	ST17
TS11	ST21	TS21	ST7	TS35	ST16

5. References

1. Visit www.renesas.com/synergy/tools to learn more about development tools & utilities. Visit www.renesas.com/synergy/solutionsgallery to download them.
2. Download the application note *Capacitive Touch Hardware Design and Layout Guidelines for Synergy, RX200, and RX100* from www.renesas.com/synergy/docs to learn about the hardware design guidelines for Renesas Synergy Capacitive Touch.
3. Download the application project *Tuning the Capacitive Touch Solution* from www.renesas.com/us/en/software/D6003780.html to learn about the capacitive touch tuning with CTW for Synergy.
4. Download the Button and Slider Framework Module Guides:
   - *Capacitive Touch Button Framework Module Guide* (www.renesas.com/us/en/software/D6003125.html)
   - *Capacitive Touch Slider Framework Module Guide* (www.renesas.com/us/en/software/D6003126.html)

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.04.17	—	First release
1.01	Nov.08.17	—	Update to use SSP v1.3.0 and later; Update to use CTW for Synergy First Step Guide version 1.05.0000.28 and later.
1.02	Dec.05.17	—	Source code updated
1.03	Apr.27.18	—	Updated for SSP v1.4.0
1.04	Oct.11.18	—	Updated for SSP v1.5.0
1.05	Mar.22.19	—	Updated for SSP v1.6.0
1.06	Oct.14.19	—	Updated for SSP v1.7.0

Corporate Headquarters

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

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/.