Renesas Synergy™ Platform

Cellular Framework Module Guide

R11AN0253EU0100
Rev.1.00
Nov 9, 2018

Introduction

The Cellular Framework module provides a high-level application layer interface for the cellular modem integration in SSP. The Cellular Framework provides a common interface for applications to interface with the Cellular modems from various vendors.

The Cellular Framework provides a set of APIs to provision, configure, and communicate with the cellular network for data communication. The Cellular Framework uses the Console Framework to communicate with Cellular modems over a serial interface by using AT commands. The Cellular Framework creates the serial data pipe over serial interface for data communication, leveraging the PPP WAN protocol provided by NetX™. Data communication using TCP/IP can be established over this Wide Area Network (WAN) link using NetX Application protocols, sockets or IoT protocols such as MQTT.

The Cellular Framework also provides the framework level Socket APIs to communicate with the TCP/IP stack present on-chip (inside cellular hardware module) in certain cellular hardware modules and with the TCP/IP link for the network using socket APIs.

Contents

1. Cellular Framework Module Features

Features of the Cellular Framework module are as follows:

- Supports connectivity using:
  - CAT1, CAT3, and CAT M1 Cellular Modems
  - BSD Socket interface for On-Chip stack present on the Cellular Module
  - NetX Stack on Synergy MCU (Host) using NSAL interface
- Supports a common a set of APIs to interface to the networking stack and generic interface for the different Cellular hardware modules
- Using generic APIs and abstraction, applications developed for the cellular hardware module can be easily migrated to work with another cellular hardware module
- Supported Cellular modems:
  - NimbeLink CAT3 (NL-SW-LTE-TSVG) Verizon-US
  - NimbeLink CAT3 (NL-SW-LTE-TEUG) India and Europe
  - NimbeLink CAT1 (NL-SW-LTE-GELS3-B and NL-SW-LTE-GELS3-C) Verizon-US
  - Quectel CAT M1-BG96

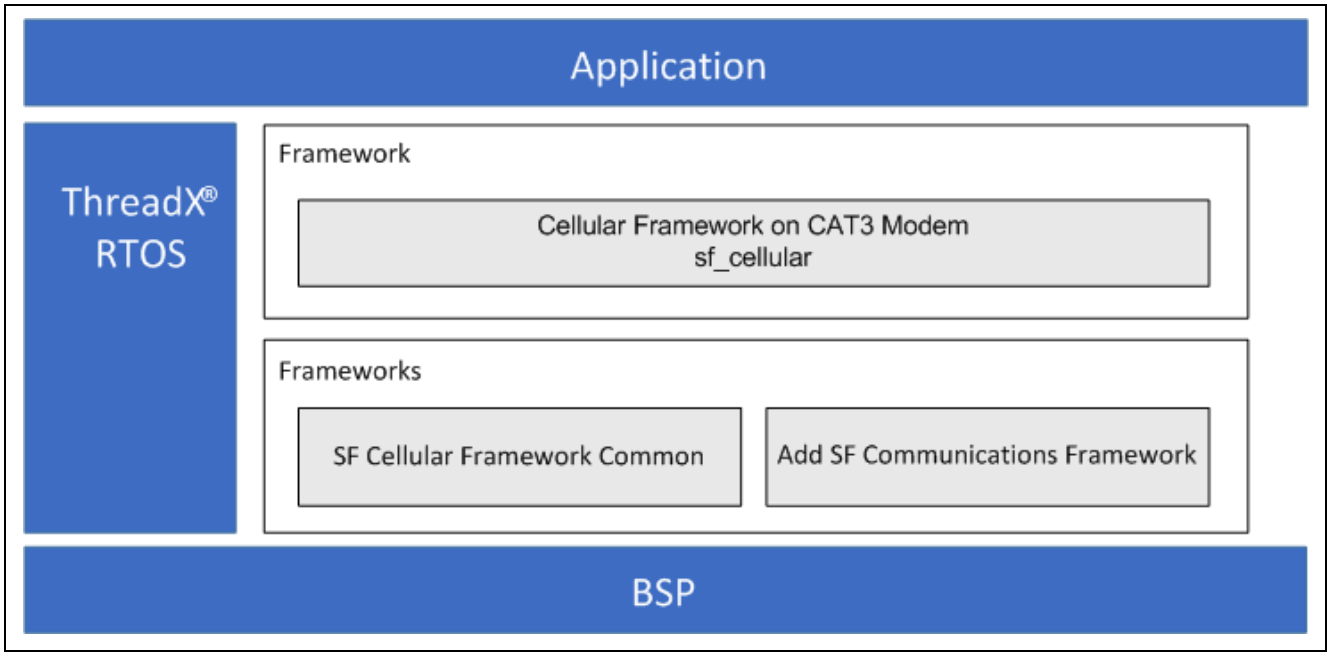

Figure 1 Cellular Framework on CAT3 Modem Module Organization, Options, and Stack Implementations

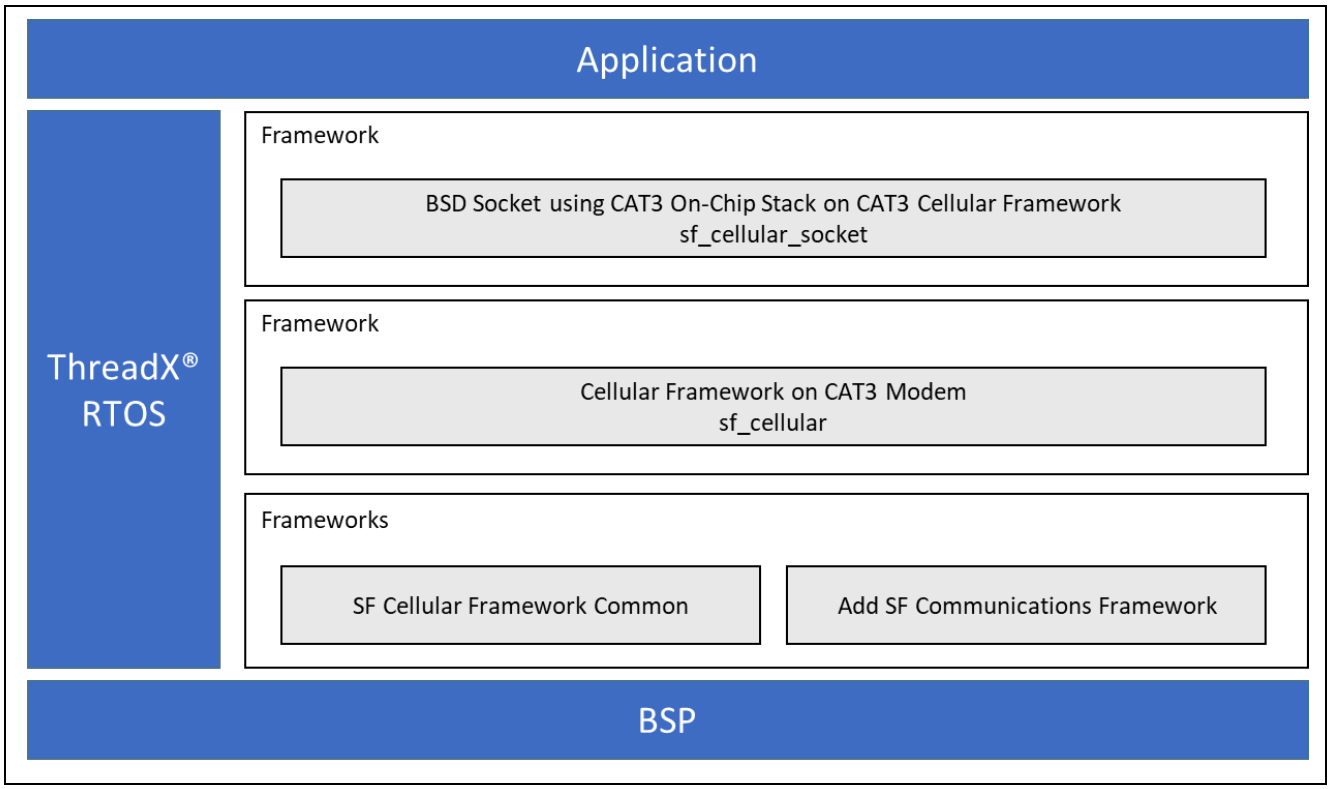

Figure 2 BSD Socket using CAT3 On-Chip Stack on CAT3 Cellular Framework Module Organization, Options, and Stack Implementations

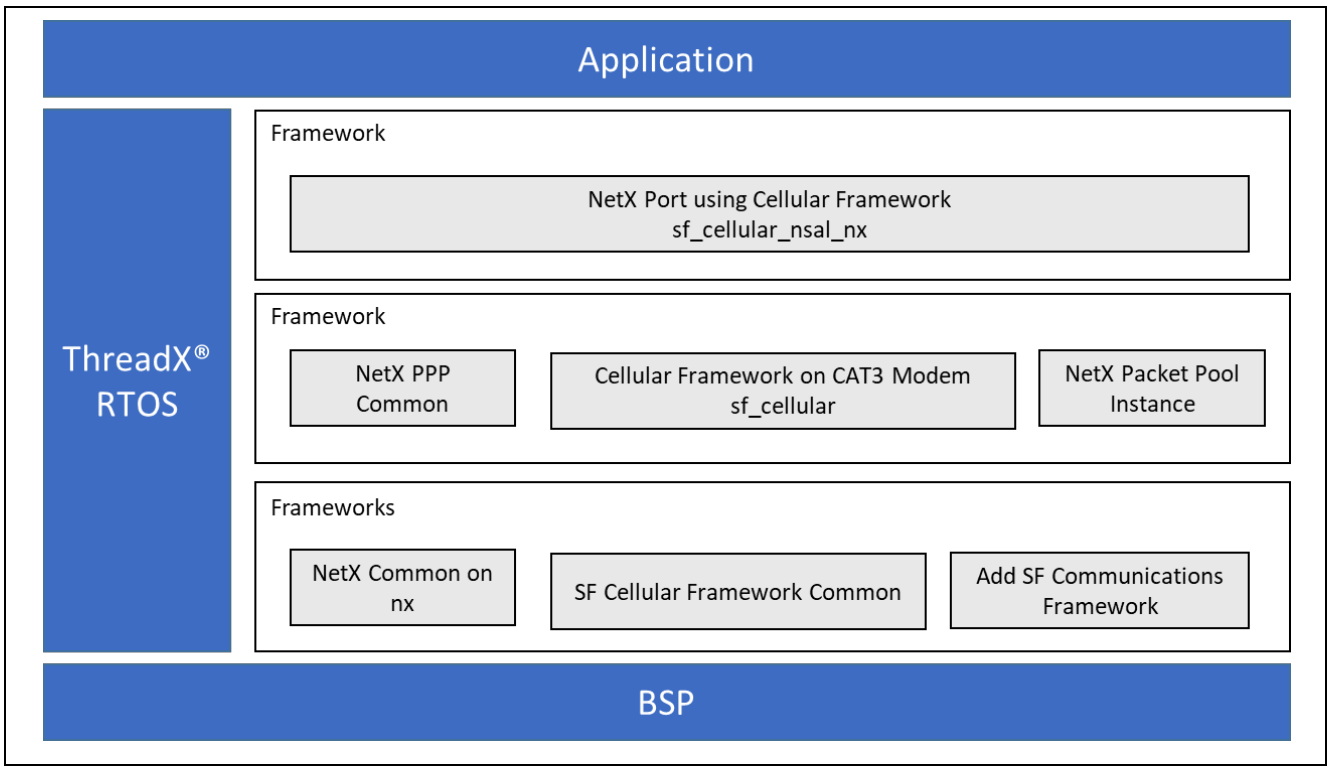

Figure 3 NetX Port using Cellular Framework Module Organization, Options, and Stack Implementations

2. Cellular Framework Module APIs Overview

The Cellular Framework defines APIs for each of the related modules. The following descriptions explain the operation of each API.

Note: A more detailed description of the Cellular framework module APIs are available in the Cellular Application Note downloadable from the Renesas web site. Search for R30AN0311 in the search bar to see the application note and application project listed in the search results.

The Cellular framework provides a generic interface for the network stack and applications. The following table shows the APIs provided by the framework.

Table 1 Cellular Framework Module API Summary

Function Name	Example API Call and Description
`.open`	`g_sf_cellular0.p_api->open (g_sf_cellular0.p_ctrl, g_sf_cellular0.p_cfg);` This API function initializes and enables the Cellular module. The open function returns the Cellular control structure, uniquely identifying the instance of the Cellular framework. The Cellular framework open function accepts the Cellular module configuration as an argument, with the following parameters: • Operator Selection Mode (enumeration) • Operator Name Format (enumeration) • Operator Name (string) • Preferred Operator List (array of structures) • Time zone update policy (enumeration)
`.close`	`g_sf_cellular0.p_api->close (g_sf_cellular0.p_ctrl);` This API un-initializes the Cellular module and disables it. It takes the Cellular control structure as an argument.
`.infoGet`	`g_sf_cellular0.p_api->infoGet (g_sf_cellular0.p_ctrl, p_cellular_info);` This API takes the Cellular control structure as an argument and returns the following information obtained from the Cellular module: • Chipset/driver information (string) • Manufacturer Name (string) • Firmware version (string) • IMEI Number (string) • RSSI value (unsigned integer 16 bits) • Bit Error Rate (unsigned integer 16 bits)
`.statisticsGet`	`g_sf_cellular0.p_api->statisticsGet (g_sf_cellular0.p_ctrl, p_stats);` This API gets the data statistics from the Cellular module. It takes the Cellular control structure as an argument and returns the following statistics: • Received packets (unsigned integer 32 bits) • Transmitted packets (unsigned integer 32 bits) • Transmit packet errors (unsigned integer 32 bits)
`.transmit`	`g_sf_cellular0.p_api->transmit (g_sf_cellular0.p_ctrl, p_buffer, length);` This API sends the data/packet out. It takes the Cellular control structure, the data buffer and the data buffer length as an argument. The Cellular framework transmit function passes the data buffer to the PPP driver for transmission.

.provisioningGet	g_sf_cellular0.p_api->provisioningGet (g_sf_cellular0.p_ctrl, p_cellular_provisioninfo); This API takes the Cellular control structure as an argument and returns the following parameters: • Authentication type(enumeration) • Username (string) • Password (string) • APN Name(string) • PDP Context ID (integer) • PDP Context Type (enumeration) • Airplane mode (enumeration)
.provisioningSet	g_sf_cellular0.p_api->provisioningSet (g_sf_cellular0.p_ctrl, p_cellular_provisioninfo); This API sets the authentication credential information. It takes the Cellular control structure and the following parameters as argument to provision the Cellular module: • Authentication type(enumeration) • Username (string) • Password (string) • APN Name(string) • PDP Context ID (integer) • PDP Context Type (enumeration) • Airplane mode (enumeration)
.networkConnect	g_sf_cellular0.p_api-> networkConnect (g_sf_cellular0.p_ctrl); This API establishes the Network connection over Cellular Network using which application can communicate to remote host with the help of Network stack. It takes the Cellular control structure as an argument.
.networkDisconnect	g_sf_cellular0.p_api->networkDisconnect (g_sf_cellular0.p_ctrl); This API terminates the Network connection established using networkConnect API. It takes the Cellular control structure as an argument.
.simPinSet	g_sf_cellular0.p_api->simPinSet (g_sf_cellular0.p_ctrl, p_pin); This API allows the application/user to change the PIN required to register on Cellular Network. It takes the Cellular control structure, old PIN and New PIN as arguments.
.simLock	g_sf_cellular0.p_api->simLock (g_sf_cellular0.p_ctrl, p_pin); This API locks the SIM. It takes the Cellular control structure and Sim PIN as arguments.
.simUnlock	g_sf_cellular0.p_api->simUnlock (g_sf_cellular0.p_ctrl, p_pin); This API unlocks the SIM. It takes the Cellular control structure and Sim PIN as arguments.
.simIDGet	g_sf_cellular0.p_api->simIDGet (g_sf_cellular0.p_ctrl, p_sim_id); This API reads the Sim ID from the Cellular module. It takes the Cellular control structure as argument and returns the SIM ID read from the Cellular module.
.commandSend	g_sf_cellular0.p_api->commandSend (g_sf_cellular0.p_ctrl, p_input_at_command, p_output, timeout); This API will send AT command provided by user to the Cellular Modem and will collect the response from the Modem and will send it back to the user.

.networkStatusGet	g_sf_cellular0.p_api->networkStatusGet (g_sf_cellular0.p_ctrl, p_status); This API gets Cellular Module Network Status information. It takes the Cellular control structure as argument and returns following parameters: • Country code (integer) • Operator code (integer) • RSSI (integer) • Cell ID (string) • IMSI (string) • Operator name (string) • Service Domain (integer) • Active Band (integer)
.fotaCheck	g_sf_cellular0.p_api->fotaCheck (g_sf_cellular0.p_ctrl, p_fota_check); This API checks for available firmware Upgrade. It takes the Cellular control structure and fota check specific structure as argument.
.fotaStart	g_sf_cellular0.p_api->fotaStart (g_sf_cellular0.p_ctrl, p_fota_start); This API starts the firmware upgrade. It takes the Cellular control structure and fota start specific structure as argument.
.fotaStop	g_sf_cellular0.p_api->fotaStop (g_sf_cellular0.p_ctrl, p_fota_stop); This API stops the firmware upgrade. It takes the Cellular control structure and fota stop specific structure as argument.
.versionGet	g_sf_cellular0.p_api->versionGet (p_version); This API retrieves the version for the API using the version pointer.
.reset	g_sf_cellular0.p_api->reset (g_sf_cellular0.p_ctrl, reset_type); Reset the cellular hardware module.

The Cellular socket framework provides additional APIs as shown in the following table.

Table 2 Cellular Socket Framework Module API Summary

Function Name	Example API Call and Description
.open	g_sf_cellular_socket0.p_api->open (g_sf_cellular_socket0.p_ctrl, g_sf_cellular_socket0.p_cfg); This API calls the Cellular Framework's lower level Open () API to Initialize the Cellular Device Driver.
.close	g_sf_cellular_socket0.p_api->close (g_sf_cellular_socket0.p_ctrl); This API calls the Cellular Framework's lower level Close () API to close the Cellular Device Driver
.versionGet	g_sf_cellular_socket0.p_api->versionGet (p_version); This API retrieves the version for the API using the version pointer.

These APIs can be used by the application to perform data transfer using sockets. It includes socket APIs which are compliant with BSD APIs:

- socket
- close
- connect
- send
- recv
- sendto
- recvfrom

- select

The following table lists the Cellular Framework specific error codes. These error codes are part of ssp_err_t.

Table 3 Cellular Framework Error Codes

Error Codes	Description
SSP_SUCCESS	Call successful
SSP_ERR_CELLULAR_CONFIG_FAILED	Configuration failed
SSP_ERR_CELLULAR_INIT_FAILED	Initialization failed
SSP_ERR_CELLULAR_TRANSMIT_FAILED	Transmit failed
SSP_ERR_CELLULAR_FW_UPTODATE	Up to date
SSP_ERR_CELLULAR_FW_UPGRADE_FAILED	Upgrade failed
SSP_ERR_CELLULAR_FAILED	General failure
SSP_ERR_CELLULAR_INVALID_STATE	Invalid state

3. Cellular Framework Module Operational Overview

3.1 Cellular Framework Module Operational Introduction

The Cellular framework provides a generic interface for applications to seamlessly communicate with the Cellular hardware module from various vendors without the necessity of changing the applications. The framework mainly consists of common set of APIs, to interface to the networking stack and different Cellular hardware modules. This section introduces the Cellular framework's basic blocks and key features that enable you to determine whether the intended Cellular application can be developed using the Cellular framework.

Note: Additional operational descriptions of the Cellular framework module are available in the Cellular Application Note downloadable from the Renesas web site. Just search, in the top search bar, for R30AN0311 and the application note and application project will be listed in the search results.

With the Generic API and abstraction, the applications developed for the cellular hardware module can be easily ported to use another cellular hardware module. The networking stack NetX is also integrated with the framework using the Network Software Abstraction Layer (NSAL).

The Synergy Cellular Framework consists of the following logical blocks:

- Synergy Cellular Framework Application Interface
- Network Stack Abstraction Layer (NSAL) for NetX TCP/IP stack
- Cellular Device Driver (AT command interface for interacting with the cellular chipset)
- BSD Socket compatible APIs for interfacing with Cellular hardware module that supports on-chip networking stack
- Synergy Software Package (SSP) HAL Interface

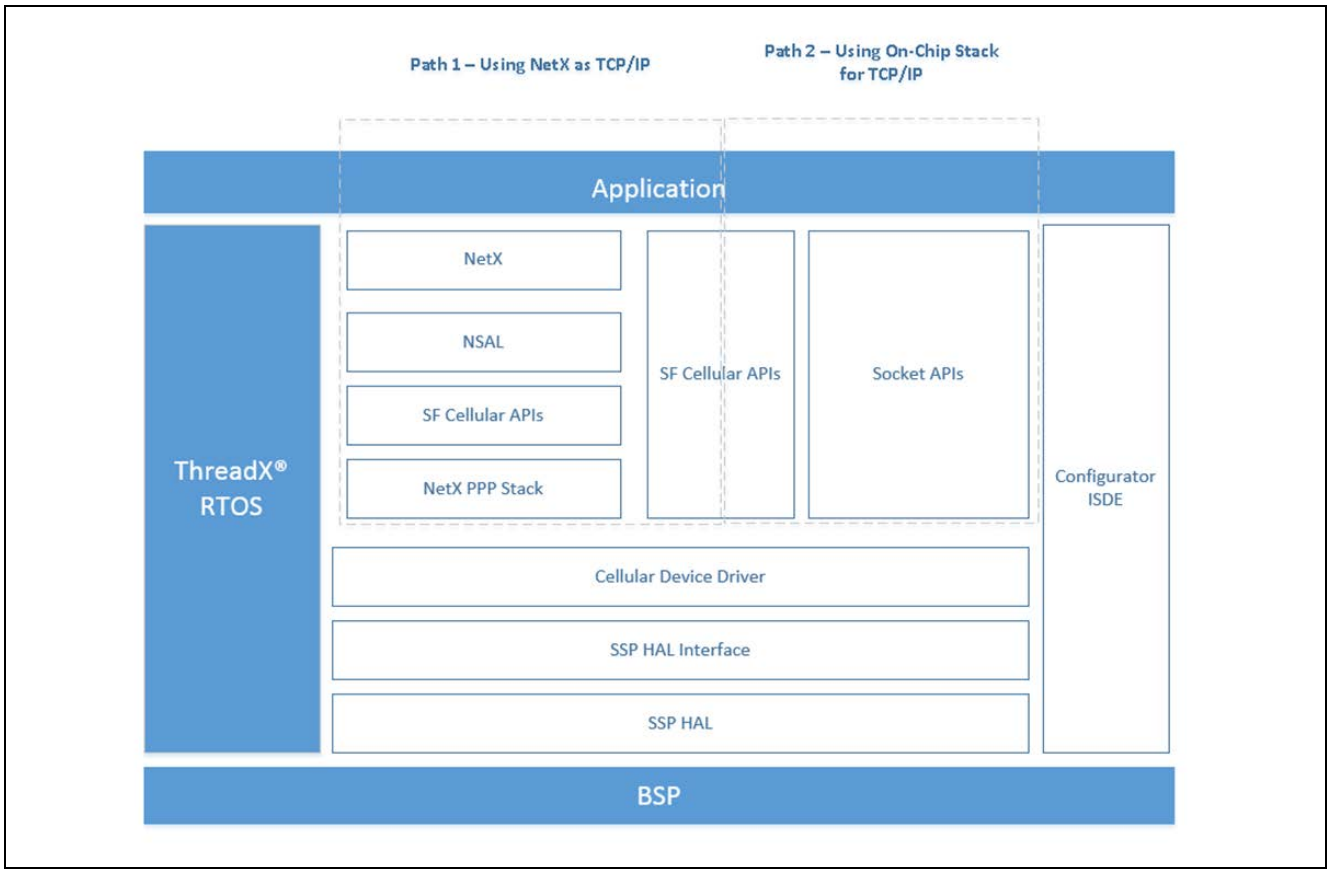

Figure 4 Cellular Framework Module Application Perspective

The Cellular Framework provides a common set of interfaces for the application to configure, provision and to communicate with the Cellular hardware module. By using these generic interfaces, the user can develop the Cellular based application using Synergy MCUs. The Cellular hardware module has various configuration parameters as specified by the family of 3GPP standards. It is possible that individual device drivers and/or Cellular chipsets/modules will not support configuration of all parameters. At a bare minimum, the network operator, Access Point Name (APN) and security credentials are required to make the module functional.

Network stack abstraction layer

The Cellular Framework provides a network stack abstraction layer (NSAL). NSAL is layer which connects the NetX and the Cellular driver by using (PPP) stack that is used for the data communication over WAN link.

Socket interface layer

The Cellular Framework provides a Socket level API for the application to interact with the on-chip networking stack present on the Cellular hardware module. This requires the Cellular hardware module/driver to support an on-chip networking stack and socket interface. When the application uses these APIs, it uses the on-chip networking stack present on the Cellular hardware module and does not use the NSAL or the NetX and its Socket APIs and does not use the Networking stack running on the Synergy MCU Group.

PPP stack

Point to point protocol (PPP) is a widely used WAN protocol in data communication. PPP Stack is part of NetX available in the SSP. NSAL leverages the PPP stack to communicate over the serial interface to the cellular service provider's network. PPP configuration provides options, for authentication methods like PAP/CHAP for the user. These authentication mechanisms are optional for the link establishment. NSAL makes use of framework APIs to send/receive data from the Cellular hardware module. NSAL allows the cellular device driver to be re-used without any changes specific to the network stack.

Cellular device driver (AT command interface for interacting with the cellular chipset)

Cellular Framework uses the AT command set to interact with the Cellular modem using the serial driver. The serial interface used to interact with the modem is UART. The UART speed used in the framework defaults are up to 115200 bits/sec.

3.2 Cellular Framework Module Operational Description

From Figure 4, illustrating the user application perspective, the application can be used in two different paths for the communication using the framework depending on the support available on the Cellular modems. Some modules provide options to use the TCP/IP stack at the Host end and other modules provide options to use the TCP/IP stack present on the Cellular modem itself. In some cases, cellular hardware module provides both. When the host TCP/IP stack (NetX) is used, the logical layers of NetX, NSAL, PPP are used as described in the Architecture diagram. When the on-chip stack is used, the socket APIs are used to communicate with the TCP/IP stack present on the Cellular modem. However, the user cannot use both at the same time.

3.2.1 Cellular Framework Module Initialization

As shown in the following control flow diagram, during the initialization using the configuration supplied by the user as required for the Cellular modem, nx_ip_create is called, which internally invokes the NSAL driver entry function that takes care of the link level initialization and initializes the cellular hardware module. In addition, it provisions the module and establishes the Network connection using the PPP interface.

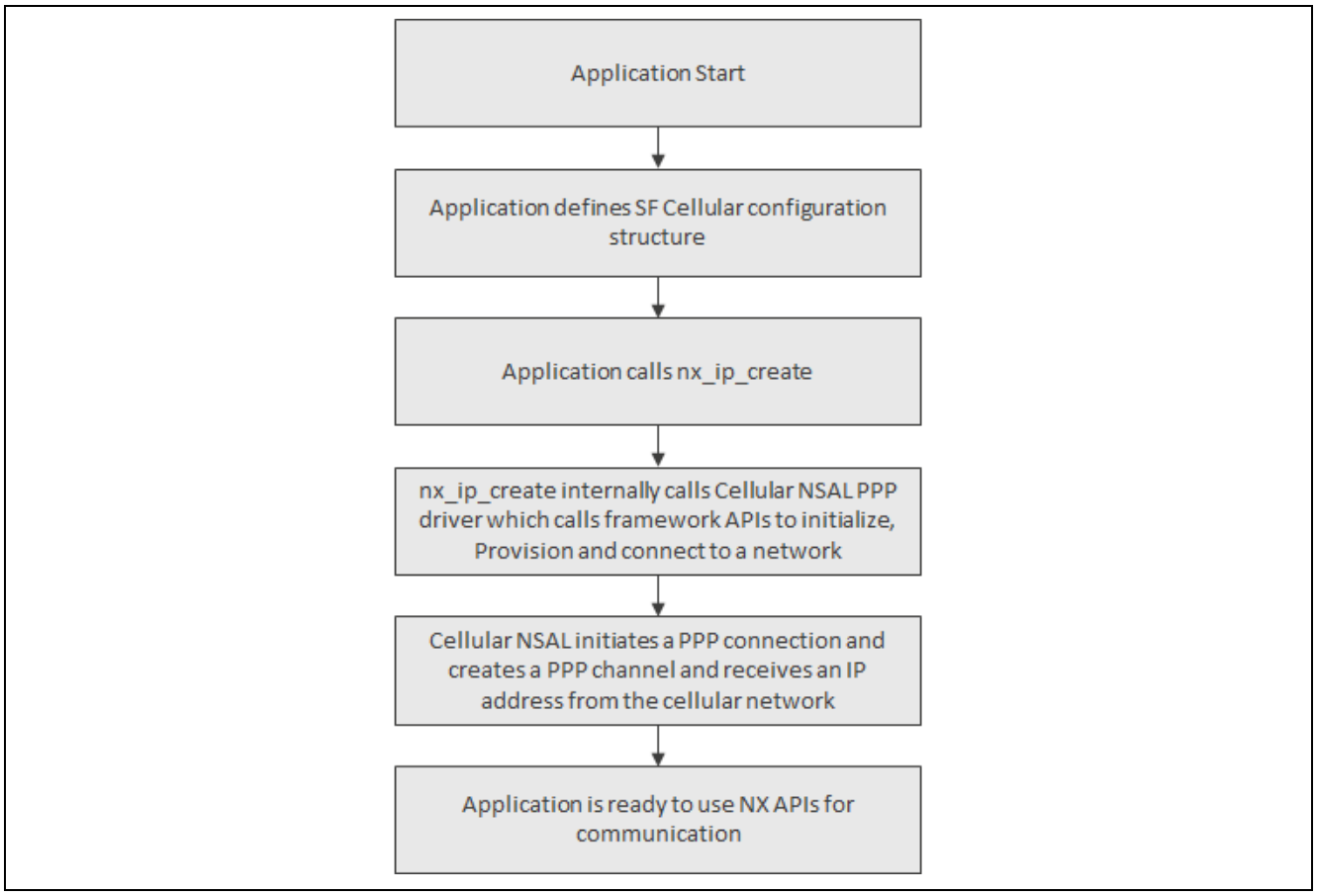

Figure 5 Cellular Framework Module Initialization Sequence

3.2.2 Cellular Hardware Module Provisioning

During the provisioning of the Cellular Modem, control structure and user configuration structures are passed as arguments. The details of the user arguments used for provisioning are the authentication, APN, username, and password.

3.2.3 Application Flow Control Using Socket Interface

The following flow diagram shows the flow for the on-chip stack path usage with the Cellular Socket interface.

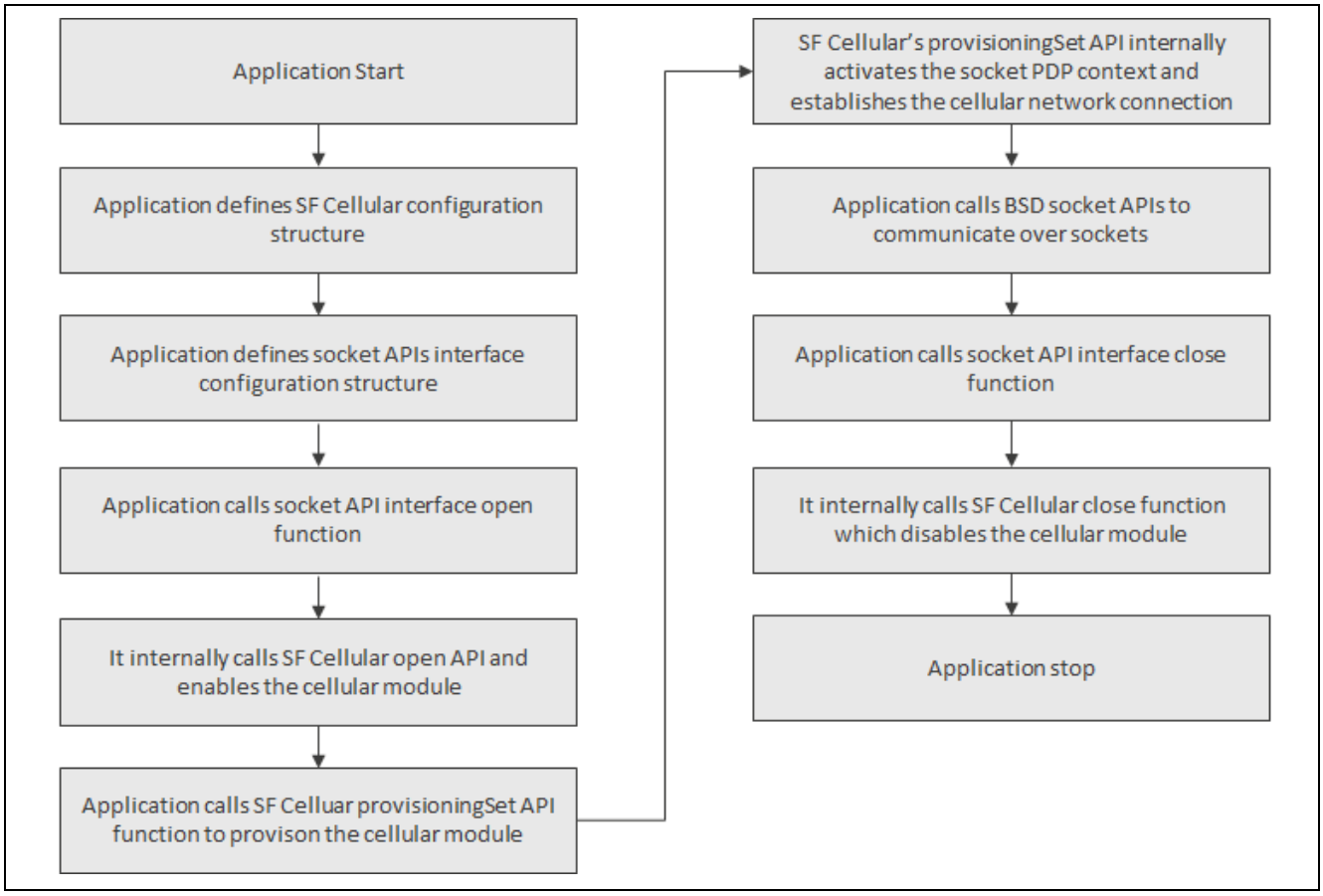

Figure 6 Cellular Framework Module Socket Interface

3.2.4 Cellular Packet Transmission

The following flow diagram shows the sequence of steps that the Packet transmission uses for the NetX application.

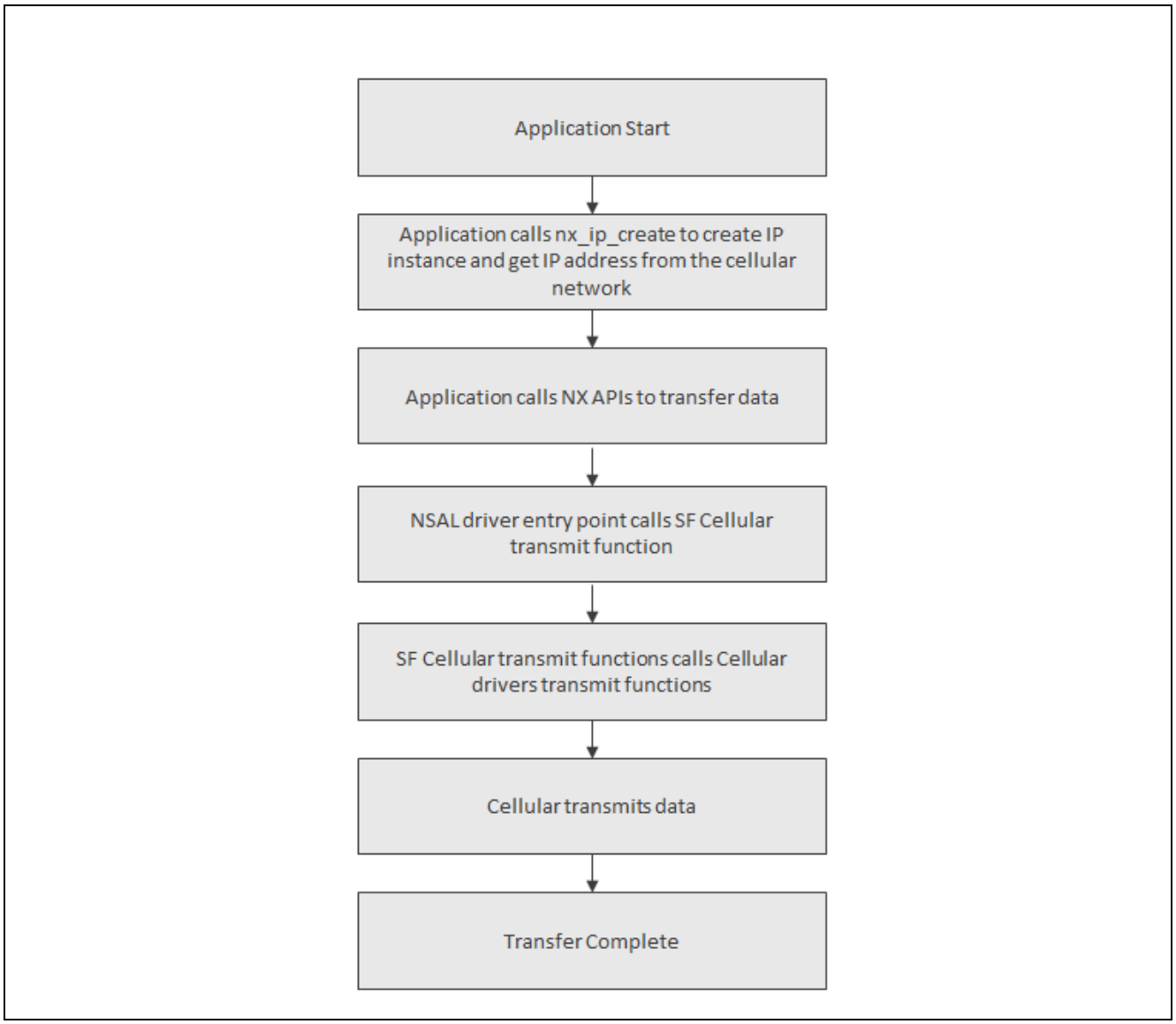

Figure 7 Cellular Framework Packet Transmission Sequence

3.2.5 Cellular Packet Reception

The following flow diagram shows the Packet reception for the Cellular Framework using NetX. In the case of receive when the data is received on the serial interface, the processing thread triggers the callback function and the callback functions handles the data and sends it to the NetX layers for further processing.

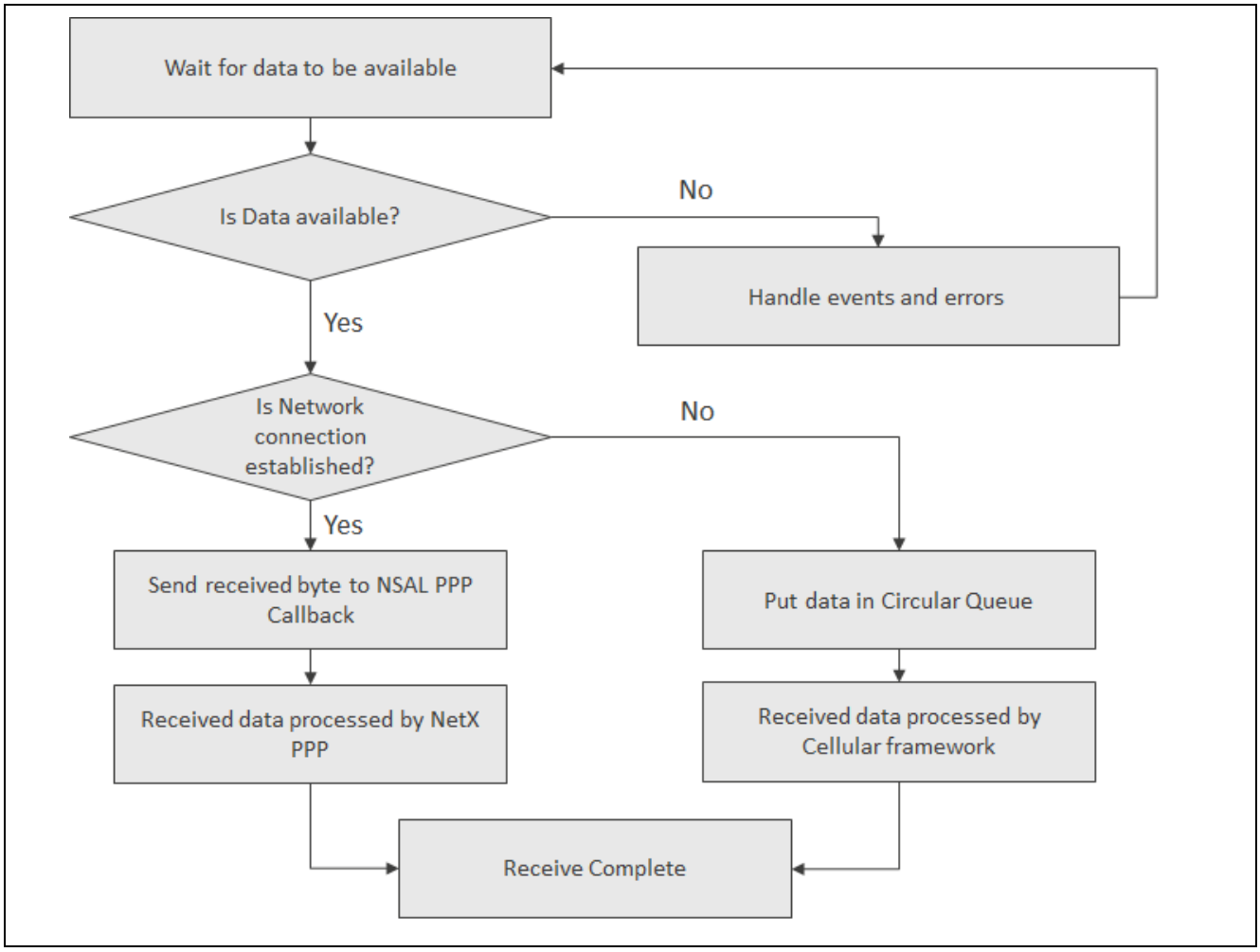

Figure 8 Cellular Framework Packet Reception Sequence

3.3 Cellular Framework Module Important Operational Notes and Limitations

- The current framework supports the below Cellular module:
  - NimbeLink CAT3 (NL-SW-LTE-TSVG) Verizon-US
  - NimbeLink CAT3 (NL-SW-LTE-TEUG) India and Europe
  - NimbeLink CAT1 (NL-SW-LTE-GELS3-B and NL-SW-LTE-GELS3-C)
  - Quectel CATM1 BG96 Verizon-US
- Firmware upgrade over air (FOTA) is not supported by NimbeLink CAT3 and CAT1 Cellular hardware module.
- Refer to the latest *SSP Release Note* for any additional limitations for this module.

4. Including the Cellular Framework Module in an Application

This section describes how to include the Cellular 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. Refer to the first few chapters of the *SSP User's Manual* if you are unfamiliar with any of these steps.

To add the Cellular Framework module to an application, simply add it to a thread using the stacks selection sequence given in the following table.

Table 4 Cellular Framework Module Selection Sequences

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_cellular_socket0 BSD Socket using CAT3 On-Chip Stack on CAT3 Cellular Framework	Threads	New Stack> Framework> Networking> Cellular > BSD Socket using CAT3 On-Chip Stack on CAT3 Cellular Framework
g_sf_cellular_0 Cellular Framework on CAT3 Modem	Threads	New Stack> Framework> Networking> Cellular > Cellular Framework on CAT3 Modem
g_sf_el_nx0 NetX Port using Cellular Framework on sf_cellular_nsal_nx	Threads	New Stack> Framework> Networking> Cellular > NetX Port using Cellular Framework on sf_cellular_nsal_nx

When the Cellular framework module is added to the thread stack as shown in the following figure, the configurator automatically adds the needed lower-level drivers. Any drivers that need additional configuration information will be highlighted in Red. Modules with a Gray band are individual modules that stand alone. Modules with a Pink band require the selection of an implementation option.

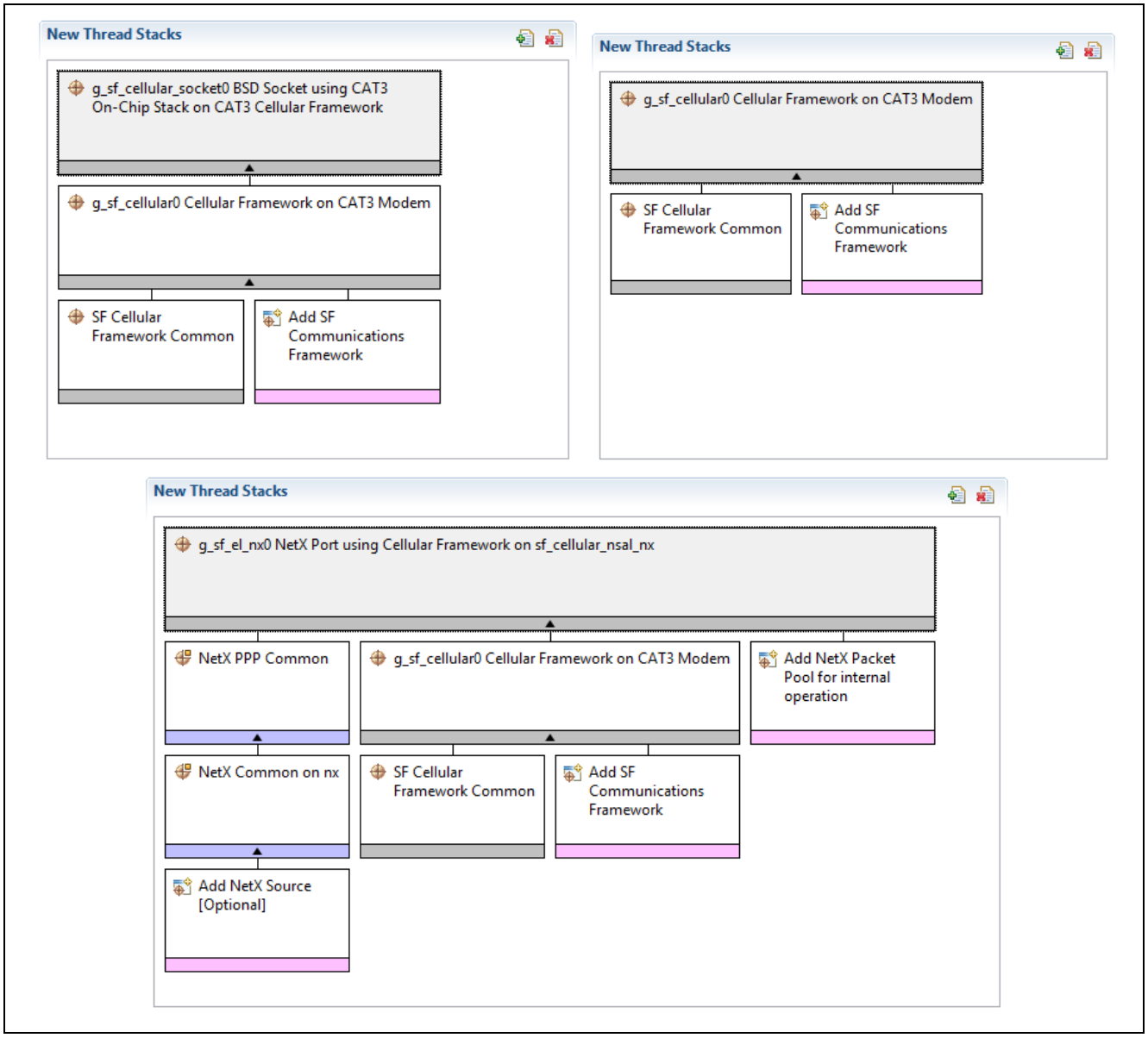

Figure 9 Cellular Framework Module Stack Options

4.1 Selecting sf_cellular_nsal_nx

The sf_cellular_nsal_nx implementation can be used in conjunction with a NetX IP Instance. In this case, it is added as an option underneath the IP Instance stack. Just add the NetX IP instance using the selection steps in the following table.

Resource	ISDE Tab	Stacks Selection Sequence
g_ip0 NetX IP Instance	Threads	New Stack> X-Ware> NetX> NetX IP Instance

Use the available option to add sf_cellular_nsal_nx under the NetX IP Instance as a NetX Network Driver. The resulting stack is shown in the following figure.

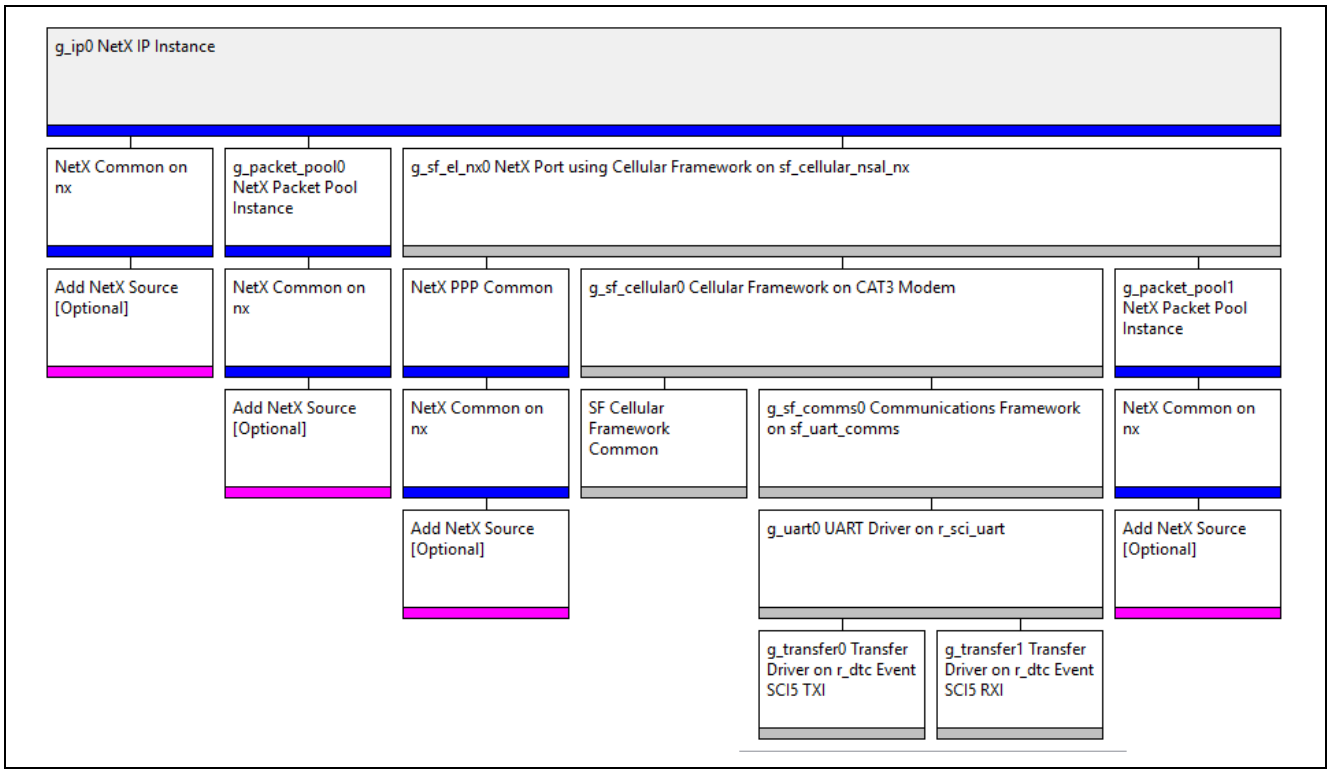

Figure 10 NetX IP Instance Use of sf_cellular_nsal_nx

5. Configuring the Cellular Framework Modules

The Cellular framework module must be configured by the user 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. Furthermore, only those properties that can be changed without causing conflicts are available for modification. Other properties are 'locked' and not available for changes and are identified with a lock icon for the 'locked' property in the Property window in the ISDE. This approach simplifies the configuration process and makes it much less error prone than previous 'manual' approaches to configuration. The available configuration settings and defaults for all the user accessible properties are given in the properties tab within the SSP Configurator, and are shown in the following tables for easy reference.

One of the properties most often identified as requiring a change is the Interrupt Priority. This configuration setting is available with the Properties window of the associated module. Simply select the indicated module and then view the properties window. The Interrupt settings are often toward the bottom of the properties list, so scroll down until they become available. Also note that the Interrupt Priorities listed in the properties window in the ISDE 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 with the ISDE when configuring Interrupt Priority levels.

Note: You may want to open your ISDE and create the Cellular Framework and explore the property settings in parallel with looking over the Configuration Table Settings given below. This will help orient you and can be a useful 'hands-on' approach to learning the ins and outs of developing with SSP.

A separate section is provided below for each of the Cellular framework module implementations.

5.1 Configuring the Cellular Framework on CAT3 Modem

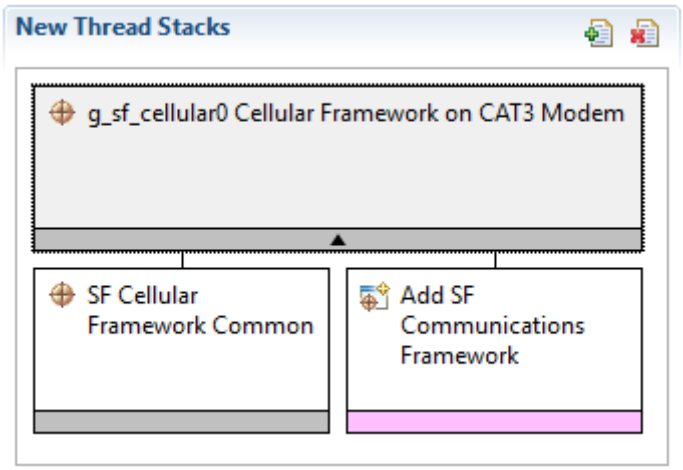

Figure 11 Cellular Framework on CAT3 Modem Thread Stack

Table 5 Configuration Settings for the Cellular Framework on CAT3 Modem on sf_cellular

ISDE Property	Setting	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking
On-Chip Stack Support	Enabled, Disabled Default: Disabled	On-chip stack support selection
Modem	TEUG, TSVG Default: TEUG	Modem selection
Name	g_sf_cellular0	Module name
SIM Pin (Used to Unlock SIM)	1111	SIM Pin selection
SIM PUK Pin (Used to Unlock SIM)	12345678	SIM PUK Pin selection
Number of Preferred Operator	0	Number of preferred operator selection
Preferred Operator 1 Name	40422	Preferred operator 1 name selection
Preferred Operator 1 Name Format	Long, Short, Numeric Default: Numeric	Preferred operator 1 name format selection
Preferred Operator 2 Name	40424	Preferred operator 2 name selection
Preferred Operator 2 Name Format	Long, Short, Numeric Default: Numeric	Preferred operator 2 name format selection
Preferred Operator 3 Name	40422	Preferred operator 3 name selection
Preferred Operator 3 Name Format	Long, Short, Numeric Default: Numeric	Preferred operator 3 name format selection
Preferred Operator 4 Name	40424	Preferred operator 4 name selection
Preferred Operator 4 Name Format	Long, Short, Numeric Default: Numeric	Preferred operator 4 name format selection
Preferred Operator 5 Name	40422	Preferred operator 5 name selection
Preferred Operator 5 Name Format	Long, Short, Numeric Default: Numeric	Preferred operator 5 name format selection

ISDE Property	Setting	Description
Operator Select Mode	Auto, Manual, Deregister, Manual Fallback Default: Auto	Operator select mode selection
Operator Name (Manual Mode Selection)	40422	Operator name selection
Operator Name Format (Manual Mode Selection)	Long, Short, Numeric Default: Numeric	Operator name format selection
Time Zone Update Policy	Enabled, Disabled Default: Enabled	Time zone update policy selection
Receive Data Callback	sf_cellular_nsal_recv_callback	Receive data callback selection
Provisioning Callback	celr_prov_callback	Provisioning callback selection
Circular Queue Size in Bytes	256	Circular queue size selection
SF Communications Framework Thread Stack Size	512	SF communications framework thread stack size selection
Numerical priority of SF Communication Framework Thread. Legal values range from 0 through (TX_MAX_PRIORITIES-1), where a value of 0 represents the highest priority.	5	Numerical priority of SF communication framework thread selection
Cellular Module Reset IO Pin	IOPORT_PORT_06_PIN_03	Cellular module reset IO pin selection

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

Table 6 Configuration Settings for the SF Cellular Framework Common

ISDE Property	Setting	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking

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

5.2 Configuring the BSD Socket using CAT3 On-Chip Stack on CAT3 Cellular Framework

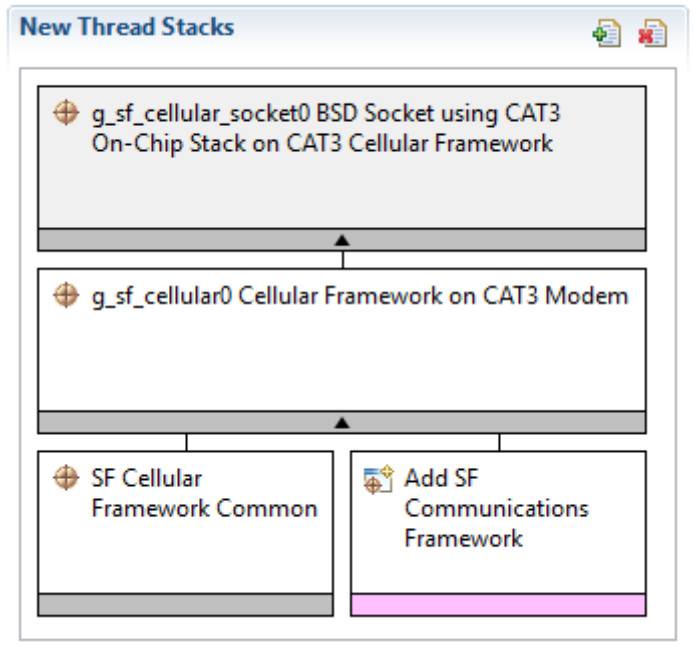

Figure 12 BSD Socket using CAT3 On-Chip Stack on CAT3 Cellular Framework Thread Stack

Table 7 Configuration Settings for the BSD Socket using CAT3 On-Chip Stack on CAT3 Cellular Framework

ISDE Property	Setting	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking
Name	g_sf_cellular_socket0	Module name

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

Table 8 Configuration Settings for the Cellular Framework on CAT3 Modem

ISDE Property	Setting	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking
On-Chip Stack Support	Enabled, Disabled Default: Disabled	On-chip stack support selection
Modem	TEUG, TSVG Default: TEUG	Modem selection
Name	g_sf_cellular0	Module name
SIM Pin (Used to Unlock SIM)	1111	SIM Pin selection
SIM PUK Pin (Used to Unlock SIM)	12345678	SIM PUK Pin selection
Number of Preferred Operator	0	Number of preferred operator selection
Preferred Operator 1 Name	40422	Preferred operator 1 name selection
Preferred Operator 1 Name Format	Long, Short, Numeric Default: Numeric	Preferred operator 1 name format selection
Preferred Operator 2 Name	40424	Preferred operator 2 name selection

Preferred Operator 2 Name Format	Long, Short, Numeric Default: Numeric	Preferred operator 2 name format selection
Preferred Operator 3 Name	40422	Preferred operator 3 name selection
Preferred Operator 3 Name Format	Long, Short, Numeric Default: Numeric	Preferred operator 3 name format selection
Preferred Operator 4 Name	40424	Preferred operator 4 name selection
Preferred Operator 4 Name Format	Long, Short, Numeric Default: Numeric	Preferred operator 4 name format selection
Preferred Operator 5 Name	40422	Preferred operator 5 name selection
Preferred Operator 5 Name Format	Long, Short, Numeric Default: Numeric	Preferred operator 5 name format selection
Operator Select Mode	Auto, Manual, Deregister, Manual Fallback Default: Auto	Operator select mode selection
Operator Name (Manual Mode Selection)	40422	Operator name selection
Operator Name Format (Manual Mode Selection)	Long, Short, Numeric Default: Numeric	Operator name format selection
Time Zone Update Policy	Enabled, Disabled Default: Enabled	Time zone update policy selection
Receive Data Callback	sf_cellular_nsal_recv_callback	Receive data callback selection
Provisioning Callback	celr_prov_callback	Provisioning callback selection
Circular Queue Size in Bytes	256	Circular queue size selection
SF Communications Framework Thread Stack Size	512	SF communications framework thread stack size selection
Numerical priority of SF Communication Framework Thread. Legal values range from 0 through (TX_MAX_PRIORITIES-1), where a value of 0 represents the hightest priority.	5	Numerical priority of SF communication framework thread selection
Cellular Module Reset IO Pin	IOPORT_PORT_06_PIN_03	Cellular module reset IO pin selection

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

5.3 Configuring the NetX Port using the Cellular Framework

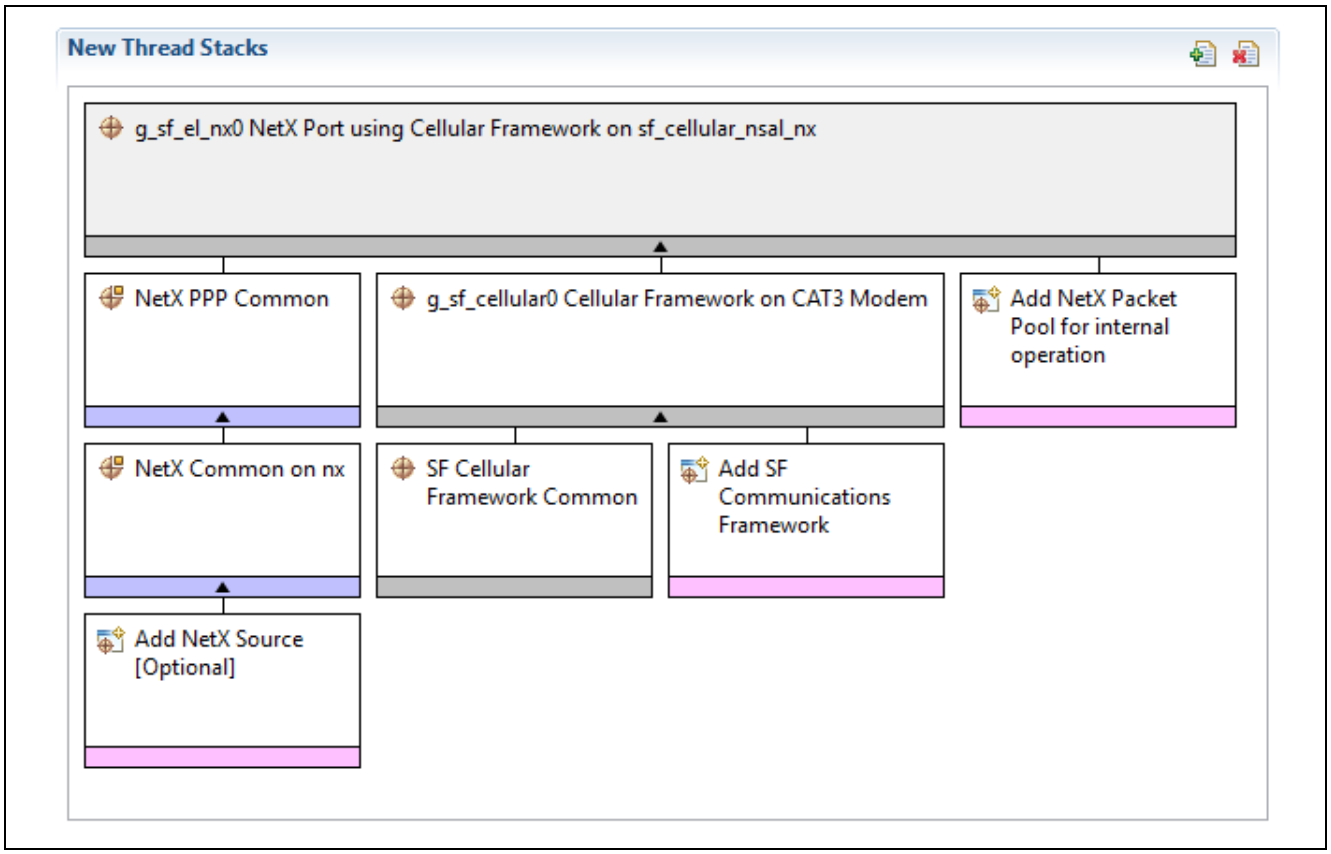

Figure 13 NetX Port using Cellular Framework Thread Stack

Table 9 Configuration Settings for the Cellular Framework on sf_cellular_nsal_nx

ISDE Property	Setting	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking
Name	g_sf_el_nx0	Module name
PPP Stack Size in Bytes	2048	PPP stack size selection
Name	g_nx_ppp0	Module name
Numerical priority of PPP Thread (Priority must be lower than IP Helper thread). Legal values range from 0 through (TX_MAX_PRIORITIES-1), where a value of 0 represents the highest priority.	3	Numerical priority of PPP thread selection
Authentication Method	None, PAP, CHAP Default: None	Authentication method selection
Invalid Packet Handler Callback	NULL	Invalid packet handler callback selection
Link Down Callback	ppp_link_down_callback	Link down callback selection
Link Up Callback	ppp_link_up_callback	Link up callback selection
PAP Login Callback	NULL	PAP login callback selection
PAP Verify Login Callback	NULL	PAP verify login callback selection
Get Challenges Values Callback	NULL	Get challenges values callback selection

ISDE Property	Setting	Description
Get Responder Values Callback	NULL	Get responder values callback selection
Get Verification Callback	NULL	Get verification callback selection
Local IPv4 Address (use commas for separation)	0,0,0,0	Local IPv4 address selection
Peer IPv4 Address (use commas for separation)	0,0,0,0	Peer IPv4 address selection

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

Table 10 Configuration Settings for the NetX PPP Common

ISDE Property	Setting	Description
Name	g_nx_ppp_common0	Module name

Table 11 Configuration Settings for the Cellular Framework on CAT3 Modem

ISDE Property	Setting	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking
On-Chip Stack Support	Enabled, Disabled Default: Disabled	On-chip stack support selection
Modem	TEUG, TSVG Default: TEUG	Modem selection
Name	g_sf_cellular0	Module name
SIM Pin (Used to Unlock SIM)	1111	SIM Pin selection
SIM PUK Pin (Used to Unlock SIM)	12345678	SIM PUK Pin selection
Number of Preferred Operator	0	Number of preferred operator selection
Preferred Operator 1 Name	40422	Preferred operator 1 name selection
Preferred Operator 1 Name Format	Long, Short, Numeric Default: Numeric	Preferred operator 1 name format selection
Preferred Operator 2 Name	40424	Preferred operator 2 name selection
Preferred Operator 2 Name Format	Long, Short, Numeric Default: Numeric	Preferred operator 2 name format selection
Preferred Operator 3 Name	40422	Preferred operator 3 name selection
Preferred Operator 3 Name Format	Long, Short, Numeric Default: Numeric	Preferred operator 3 name format selection
Preferred Operator 4 Name	40424	Preferred operator 4 name selection
Preferred Operator 4 Name Format	Long, Short, Numeric Default: Numeric	Preferred operator 4 name format selection
Preferred Operator 5 Name	40422	Preferred operator 5 name selection
Preferred Operator 5 Name Format	Long, Short, Numeric Default: Numeric	Preferred operator 5 name format selection
Operator Select Mode	Auto, Manual, Deregister, Manual Fallback Default: Auto	Operator select mode selection
Operator Name (Manual Mode Selection)	40422	Operator name selection
Operator Name Format (Manual Mode Selection)	Long, Short, Numeric Default: Numeric	Operator name format selection
Time Zone Update Policy	Enabled, Disabled	Time zone update policy selection

ISDE Property	Setting	Description
	Default: Enabled	
Receive Data Callback	sf_cellular_nsal_recv_callback	Receive data callback selection
Provisioning Callback	celr_prov_callback	Provisioning callback selection
Circular Queue Size in Bytes	256	Circular queue size selection
SF Communications Framework Thread Stack Size	512	SF communications framework thread stack size selection
Numerical priority of SF Communication Framework Thread. Legal values range from 0 through (TX_MAX_PRIORITIES-1), where a value of 0 represents the highest priority.	5	Numerical priority of SF communication framework thread selection
Cellular Module Reset IO Pin	IOPORT_PORT_06_PIN_03	Cellular module reset IO pin selection

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

Table 12 Configuration Settings for the SF Cellular Framework Common

ISDE Property	Setting	Description
Parameter Checking	BSP, Enabled, Disabled Default: BSP	Enable or disable the parameter checking

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

Table 13 Configuration Settings for the NetX Packet Pool Instance

ISDE Property	Setting	Description
Name	g_packet_pool0	Module name
Packet Size (bytes)	128	Packet size selection
Number of Packets in Pool	16	Number of packets in pool selection
Name of generated initialization function	packet_pool_init0	Name of generated initialization function selection
Auto initialization	Enable, Disable Default: Enable	Auto initialization selection

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

Table 14 Configuration Settings for the NetX Common

ISDE Property	Setting	Description
Name of generated initialization function	nx_common_init0	Name of generated initialization function selection
Auto initialization	Enable, Disable Default: Enable	Auto initialization selection

5.4 Cellular Framework Module Clock Configuration

The Cellular Framework module uses the clocks required for the specific selections of the low-level modules.

5.5 Cellular Framework Module Pin Configuration

The Cellular Framework module uses input and output pins depending on the selections of the low-level modules.

6. Using the Cellular Framework Module in an Application

In a typical Cellular application, much of the work is done by SSP based on the configured Cellular module stack. When the IP instance along with the Cellular framework is added using the configurator it includes the PPP stack as part of the framework. In addition, it also includes the NSAL and Cellular device driver code. The auto generated code is responsible Cellular initialization.

The user added code is responsible for the data connections and for the ICMP ping. It is responsible for sending the Ping request to the user entered Public IP address and verifying the Ping response. Callback functions can be implemented for PPP link down, PPP link up, and cellular provisioning.

The steps in using the Cellular framework module in a simple application are:

1. Initialization from generated code
2. Wait for link to come up using the nx_ip_status_check API
3. Ping the network using the nx_icmp_ping API
4. Release the packet once the response is received using nx_packet_release API.

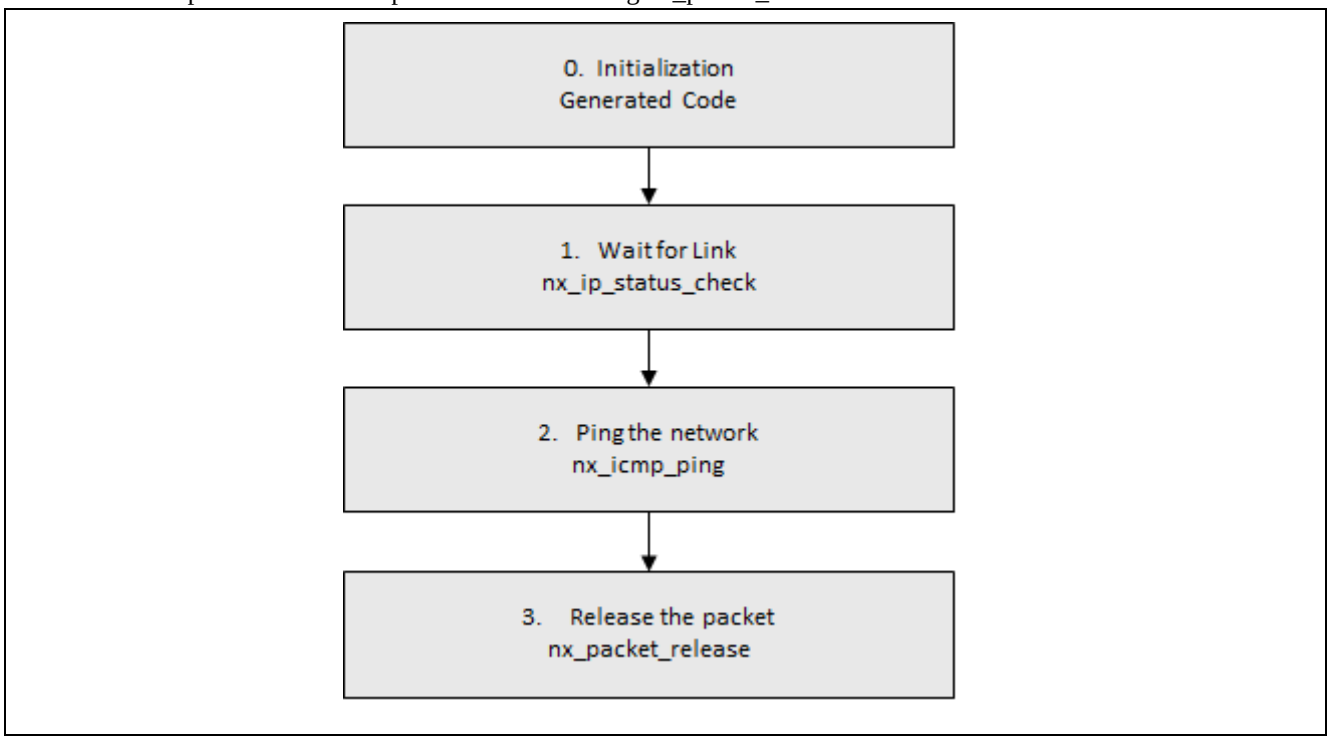

Figure 14 Flow Diagram of a Simple Cellular Framework Module Application

A more detailed description and a complete application project illustrating the typical steps in implementing a Cellular framework application are available in the Cellular Application Note downloadable from the Renesas web site.

7. The Cellular Framework Module Application Project

The application project associated with this module guide demonstrates the steps in an example application. The project can be found as described in the References section in this document. You may want to import and open the application project within the ISDE and view the configuration settings for the Cellular Framework module. You can also read over the code (in Cellular_thread_entry.c), which is used to illustrate the Cellular Framework module APIs in a complete design.

The Cellular Framework Module Application consists of following two projects:

1. sf_cellular_cat3_socket_MG_AP (based on Cellular Framework Socket APIs).
2. sf_cellular_nsal_nx_MG_AP (based on Cellular Framework NetX Stack APIs).

The application project demonstrates the typical use of the Cellular Framework module APIs. The application project Cellular thread entry initializes the Cellular Framework module. The steps from establishing connection to sending and receiving packets over network are demonstrated using Socket or NetX APIs. The messages are displayed on the Debug Console using the common semi-hosting function. The following table identifies the target versions for the associated software and hardware used by the application project.

Table 15 Software and Hardware Resources Used by the Application Project

Resource	Revision	Description
e² studio	6.2.0 or later	Integrated Solution Development Environment
SSP	1.5.0 or later	Synergy Software Platform
IAR EW for Synergy	8.21.1 or later	IAR Embedded Workshop® for Renesas Synergy™
SSC	6.2.0 or later	Synergy Standalone Configurator
SK-S7G2	3.1 or later	Starter Kit
NimbeLink CAT3 (NL-SW-LTE-TSVG) Verizon-US NimbeLink CAT3 (NL-SW-LTE-TEUG) India and Europe		CAT3 cellular module

Note: The application project assumes that you are familiar with using printf(), semi-hosting, and the Debug Console in your ISDE. If you are unfamiliar with these techniques, refer to the "How do I Use Printf() with the Debug Console in the Synergy Software Package" Knowledge Base article given in the References section at the end of this document.

A simple flow diagram of the application project is given in the following figure:

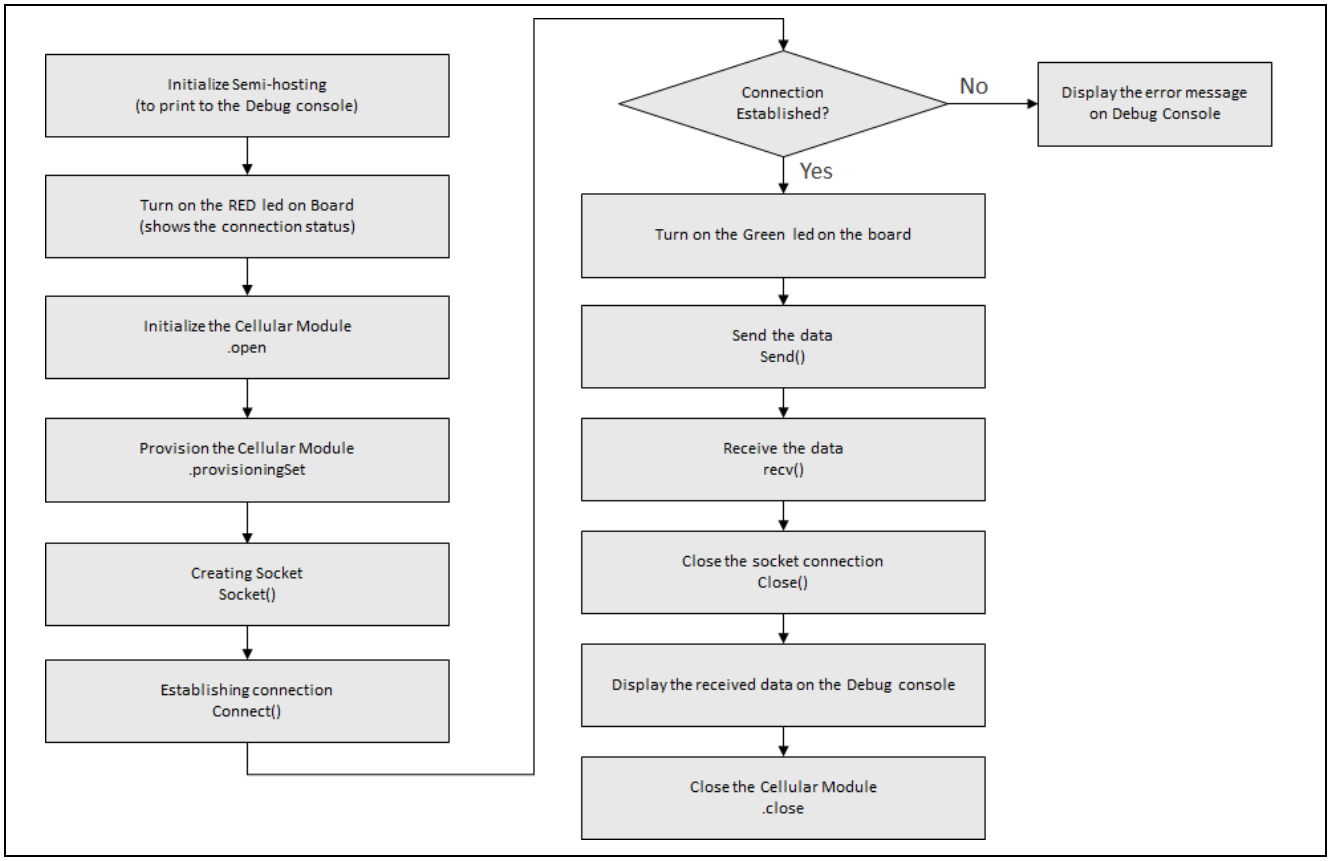

Figure 15 sf_cellular_cat3_socket_MG_AP Cellular Framework Application Project Flow

Referring to Cellular_thread_entry.c in sf_cellular_cat3_socket_MG_AP, you can follow along with the flow outlined in the Figure 15. The first section of the Cellular_thread_entry.c has the header files, which are referred to in the Cellular Framework instance structure to allow semi-hosting support to display results using printf(). The application defines the SF Cellular configuration structure prior to calling any Socket interface function. Once the Socket API interface configuration structure is defined, the application calls the Socket API interface open function. It internally calls the SF cellular open function and enables the cellular module. The application then calls the SF Cellular provisioningSet function that provisions the cellular module, activates the socket PDP context and establishes the cellular network connection. The application can now use the BSD socket APIs to communicate over the socket. The application calls the Socket API interface close function. It internally calls the SF Cellular close function which closes the cellular module.

A few key properties configured in this Application Project support the required operations and the physical properties of the target board and MCU. The following tables show properties that are listed 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.

Notes:

1. You need to select the Modem property as shown in Table 17 for your region.
2. The APN name, Context ID, and PDP type needs to be changed in DEFAULT_APN_NAME, DEFAULT_CONTEXT_ID, and DEFAULT_PDP_TYPE in cellular_thread_entry.c according to your service provider.

Table 16 BSD socket using On-Chip stack on Cellular Framework

ISDE Property	Value set
Name	g_sf_cellular_socket0

Table 17 Cellular Framework on CAT3 modem

ISDE Property	Value set
On-Chip Stack Support	Enabled
Modem	TSVG (For America) TEUG (For Europe and Asia)
AT Command Retry Count	5
Name	g_sf_cellular0
Numerical priority of SF Communication Framework Thread	5
Cellular Module Reset IO Pin	IOPORT_PORT_06_PIN_03
Provisioning Callback	celr_prov_callback
SF Communication Framework Thread Stack Size	1024

Table 18 Communications Framework on sf_uart_comms

ISDE Property	Value set
Name	g_sf_comms0
Name of generated initialization function	sf_comms_init0

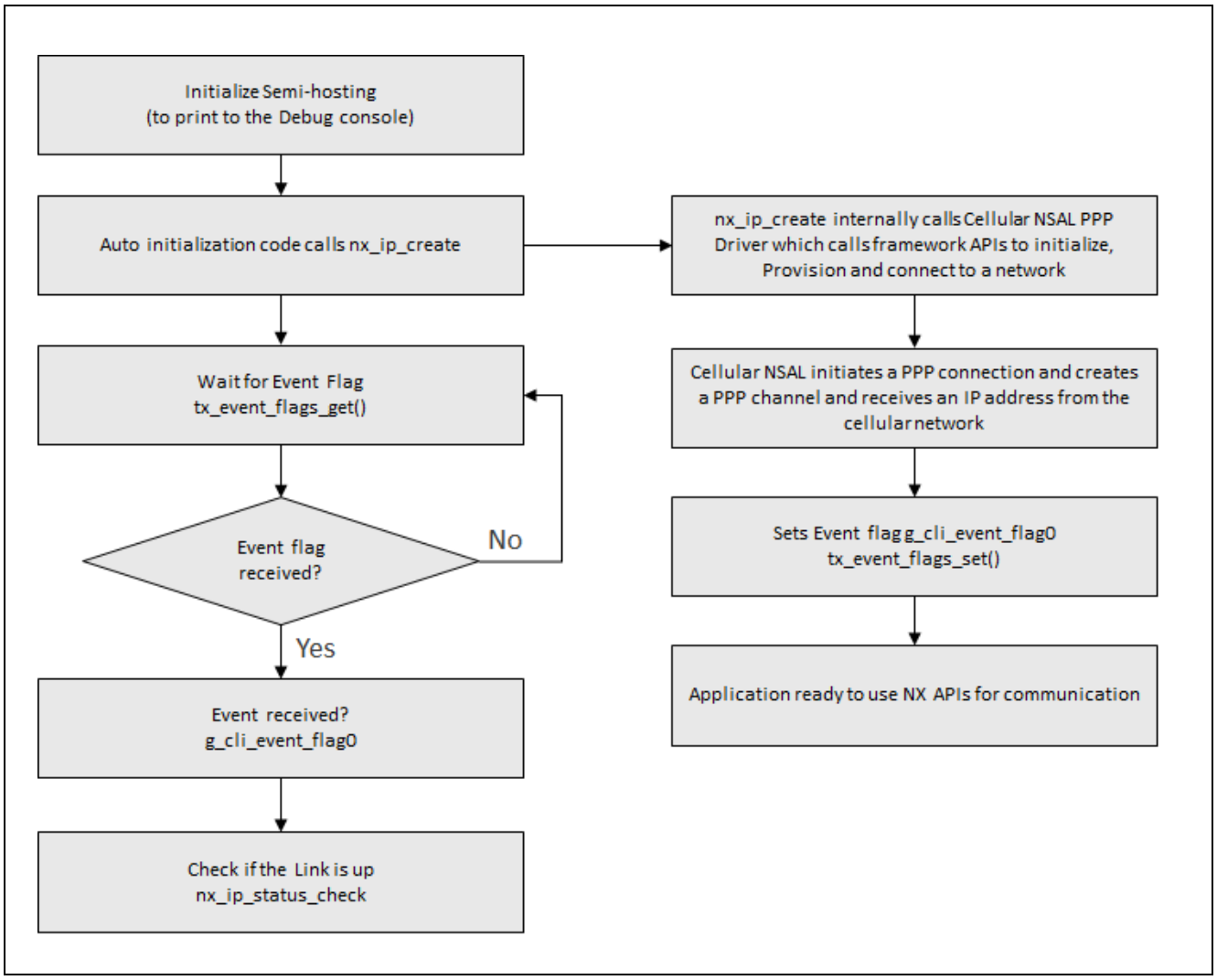

Figure 16 sf_cellular_nsal_nx_MG_AP Cellular Application Project Flow

While referring to Cellular_thread_entry.c in sf_cellular_nsal_nx_MG_AP, you can follow along with the flow outlined in Figure 16. The first section of Cellular_thread_entry.c has the header files which are referred to in the Cellular Framework instance structure to allow semi-hosting support to display results using printf(). The application defines the SF Cellular configuration structure prior to calling any function. The application calls the netx_ip_create which uses the Cellular NSAL PPP driver to call the framework APIs to initialize, provision, and connect to the network. The board receives the IP address from the cellular network. After initialization of the cellular module, the event flag g_cli_event_flag0 is sent. Upon receiving the flag, the application checks for the Link status using nx_ip_status_check. It pings the user provided IP address using nx_icmp_ping. The application releases the NetX packet using nx_packet_release once the data transfer over connection is done.

A few key properties configured in this Application Project support the required operations and the physical properties of the target board and MCU. The following tables show properties that are listed 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.

Notes:

1. You need to select the Modem property as shown in Table 23 for your region.
2. The APN name, Context ID, and PDP type needs to be changed in DEFAULT_APN_NAME, DEFAULT_CONTEXT_ID, and DEFAULT_PDP_TYPE in cellular_thread_entry.c according to your service provider.

Table 19 NetX IP Instance

ISDE Property	Value set
Name	g_ip0
IPv4 Address	192,168,0,2
IP Helper Thread Stack Size (bytes)	4096
IP Helper Thread Priority	3
ARP	Enable
ARP Cache Size in Bytes	512
UDP	Enable
ICMP	Enable
Name of generated initialization function	ip_init0
Auto Initialization	Enable

Table 20 NetX Common on nx

ISDE Property	Value set
Name of generated initialization function	nx_common_init0

Table 21 NetX Packet Pool Instance

ISDE Property	Value set
Name	g_packet_pool0
Packet Size in Bytes	2048
Number of Packets in Pool	32
Name of generated initialization function	packet_pool_init0

Table 22 NetX Port using Cellular Framework on sf_cellular_nsal_nx

ISDE Property	Value set
Name	g_sf_el_nx0
PPP Stack Size in Bytes	2048
Name	g_nx_ppp0
Numerical priority of PPP Thread	3
Authentication Method	None
Link Down Callback	ppp_link_down_callback
Link UP Callback	ppp_link_up_callback

Table 23 Cellular Framework on CAT3 modem

ISDE Property	Value set
On-Chip Stack Support	Enabled
Modem	TSVG (For America) TEUG (For Europe and Asia)
AT Command Retry Count	5
Name	g_sf_cellular0
Numerical priority of SF Communication Framework Thread	5
Cellular Module Reset IO Pin	IOPORT_PORT_06_PIN_03
Provisioning Callback	celr_prov_callback
SF Communication Framework Thread Stack Size	1024

Table 24 Communications Framework on sf_uart_comms

ISDE Property	Value set
Name	g_sf_comms0
Name of generated initialization function	sf_comms_init0

8. Customizing the Cellular Framework Module for a Target Application

Some configuration settings will normally be changed by the developer from those shown in the application project. For example, the user can change the modem setting in Cellular Framework on CAT3 Modem by using TEUG or TSVG according to the region that the cellular module is being used for. SIM PIN option can be used to unlock the SIM.

9. Running the Cellular Framework Module Application Project

To run the Cellular Framework application project and to see it executed on a target kit, you can simply import it into your ISDE, compile and run debug.

1. Import the attached Cellular Framework example project (sf_cellular_cat3_socket_MG_AP or sf_cellular_nsal_nx_MG_AP) to e^2 studio or IAR EW for Synergy. For steps to import an example project, see the Renesas Synergy™ Project Import Guide (r11an0023eu0121-synergyssp-import-guide.pdf) document attached.
2. Navigate to the Cellular thread under Threads tab. Under **g_sf_cellular0 Cellular Framework on CAT3 Modem**, Change the Modem setting to ‘TEUG’ or ‘TSVG’ according to your region.
3. In Cellular_thread_entry.c, change the DEFAULT_APN_NAME, DEFAULT_CONTEXT_ID and DEFAULT_PDP_TYPE according to your cellular network provider settings.
4. Compile the application without errors and warnings.
5. Make the connections between pmod cellular module and SK-S7G2 board as shown in the Figure 17.

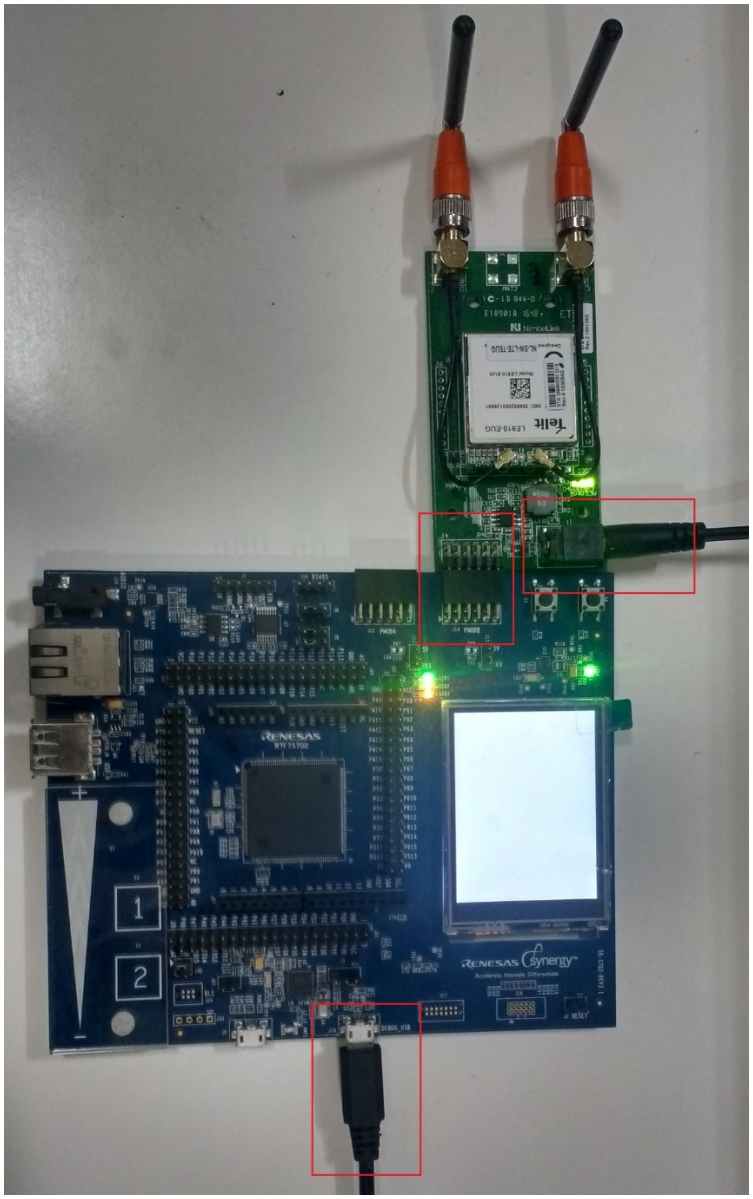

Figure 17 Board Connections

6. Connect to the host PC using a micro USB cable to J19 on SK-S7G2 as shown in the Figure 18.

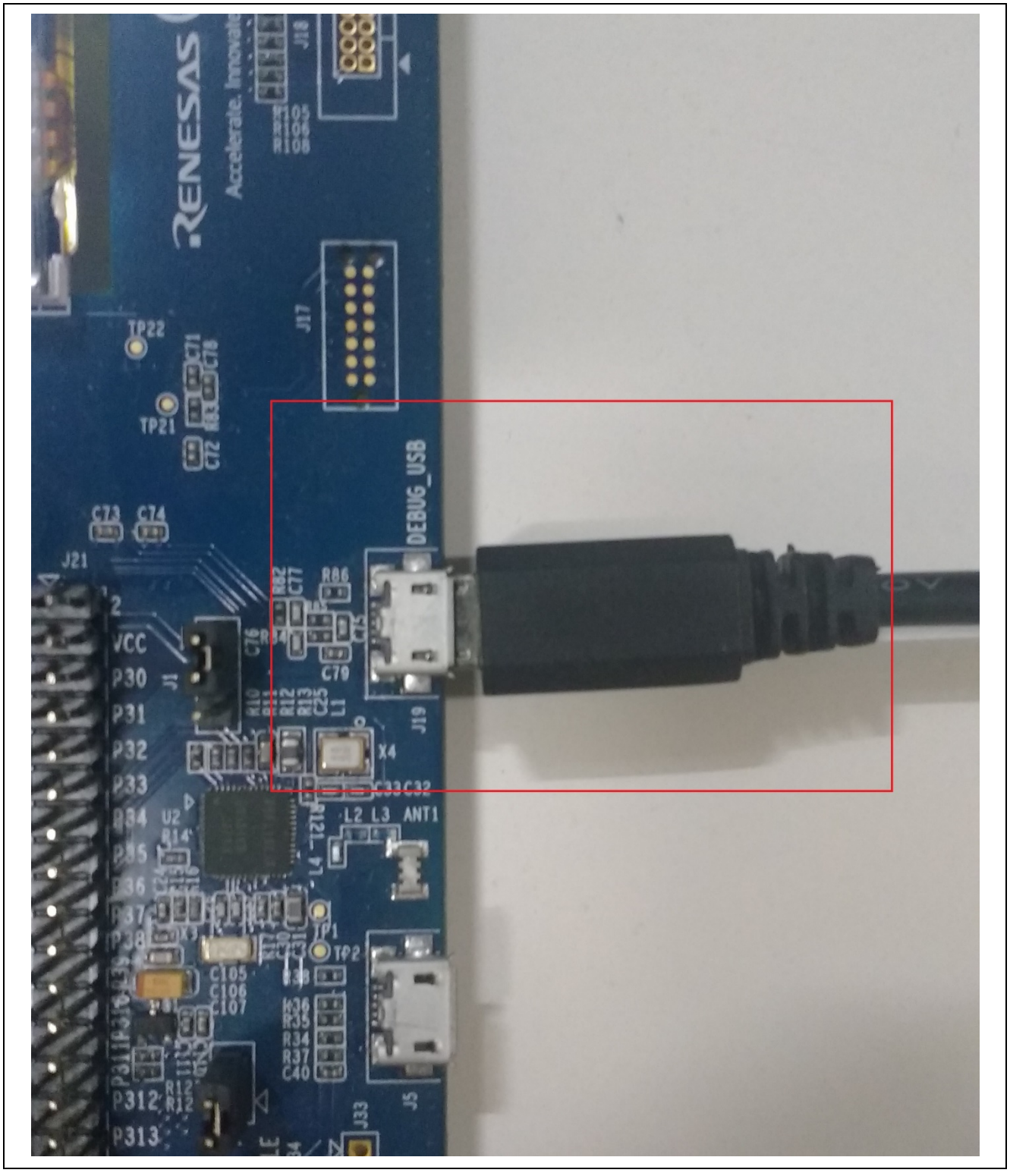

Figure 18 Debug USB

7. Connect the Cellular module to the PmodB(J14). Also connect the external supply to the cellular module as shown in the Figure 19.

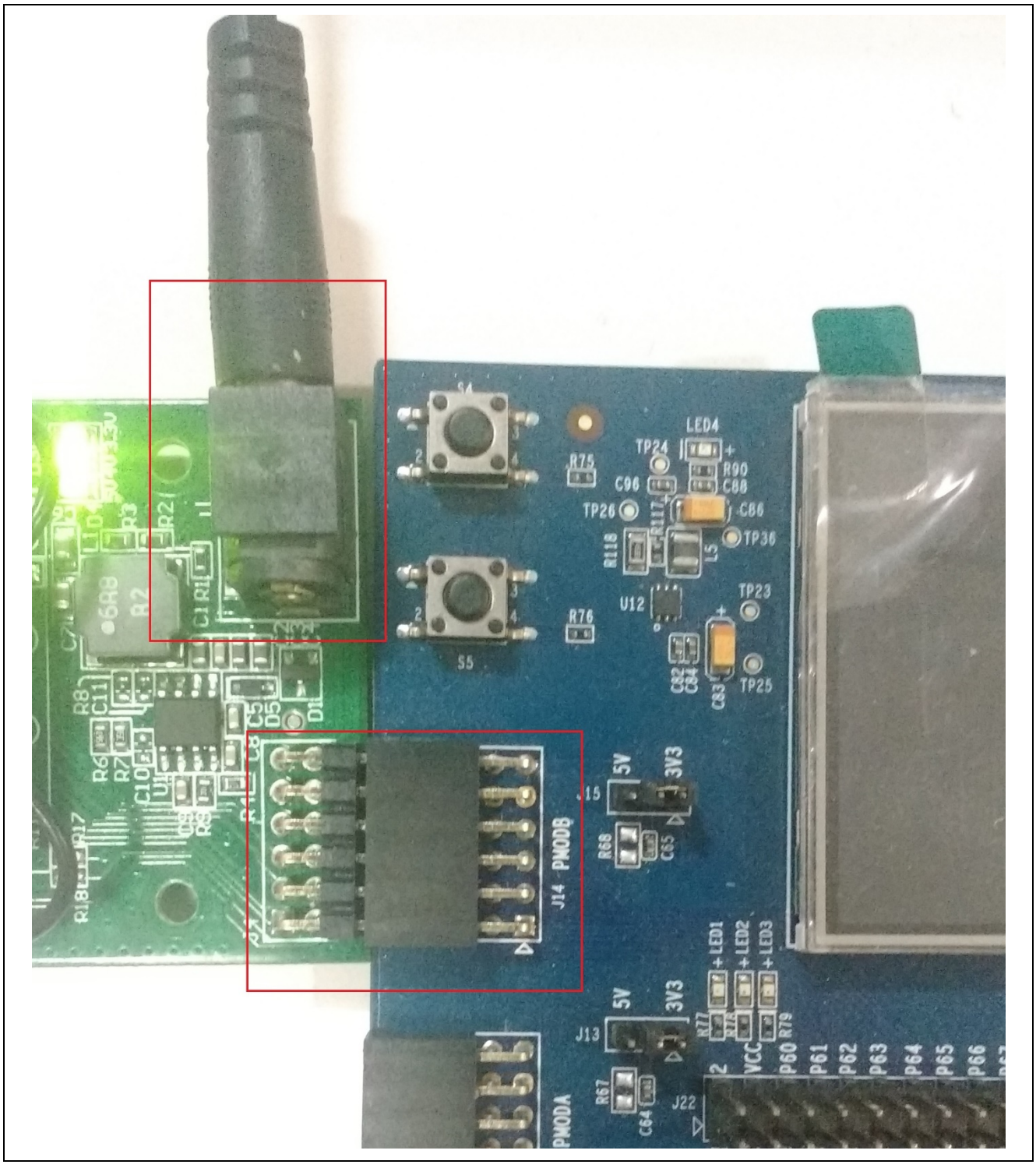

Figure 19 PMOD Connections

8. Start to debug the application.
9. For the sf_cellular_cat3_socket_MG_AP application project, the on-board red LED will glow before the connection is established. As soon as the connection is established, the red LED will be turned off and the green LED will glow.
10. The output from the application is to be viewed on Virtual Debug Console.
    Following is the output sample for the application projects.

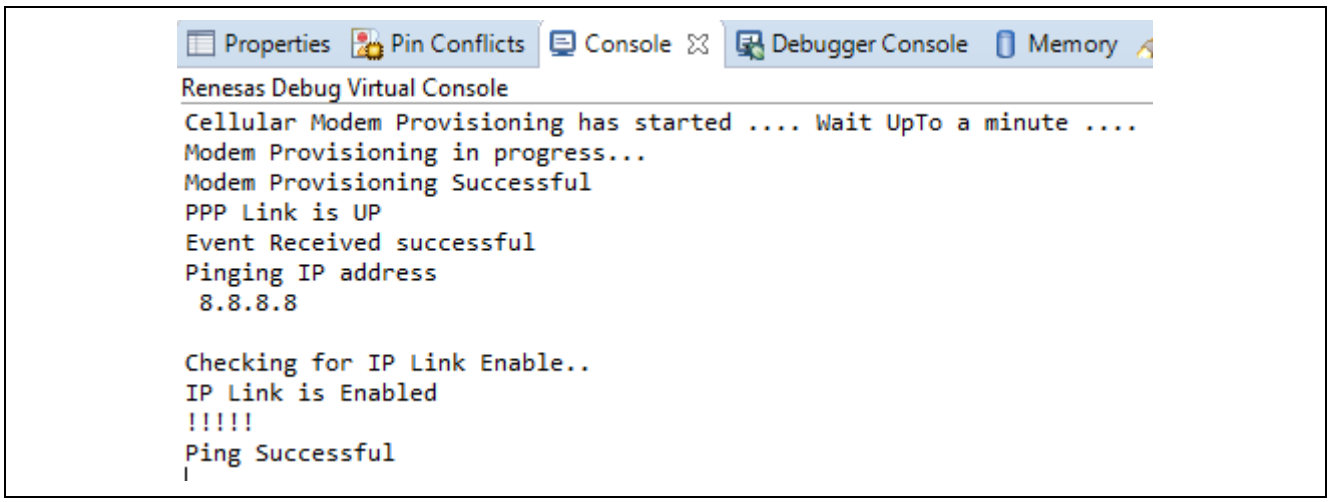

Figure 20 sf_cellular_nsal_nx_MG_AP Application Project Output

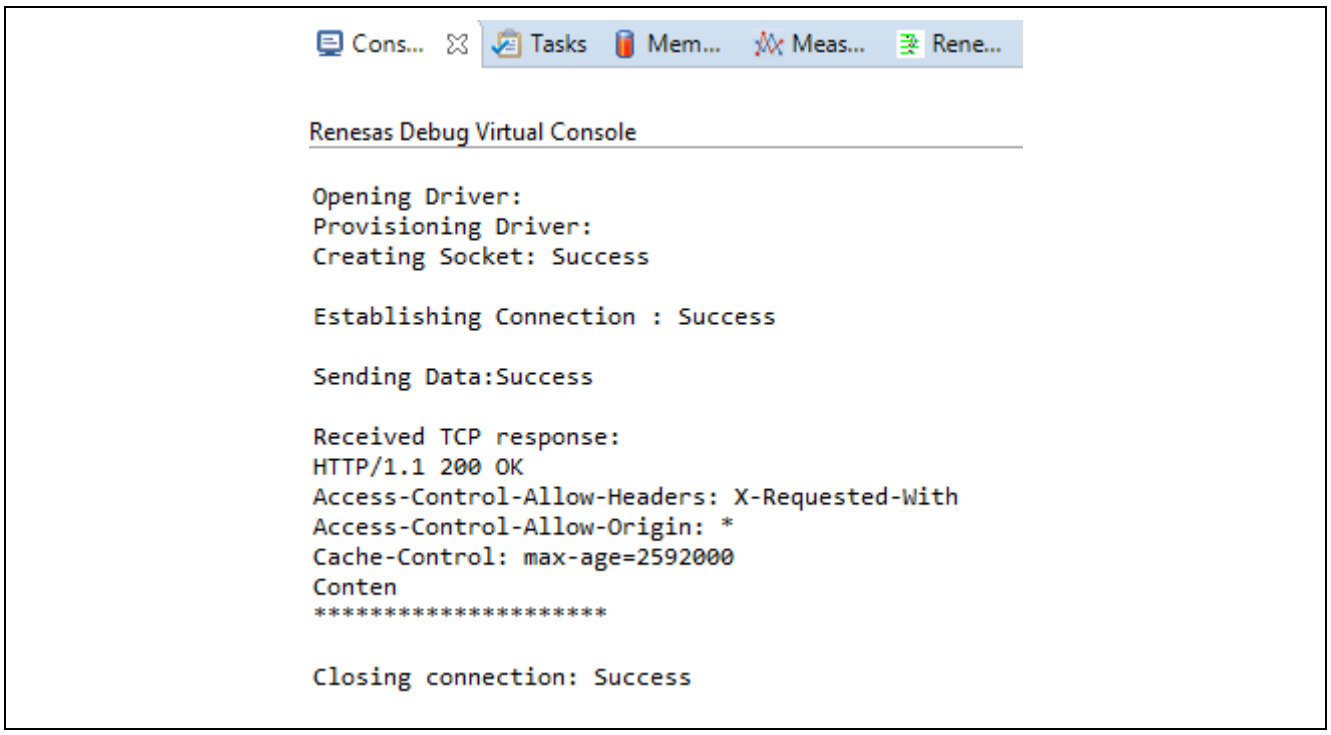

Figure 21 sf_cellular_cat3_socket_MG_AP Application Project Output

10. Cellular 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 the incorrect selection of lower-level drivers. The use of high-level APIs (as demonstrated in the application project) illustrates additional development time savings by allowing work to begin at a high level and avoiding the time required in older development environments to use or, in some cases, create, lower-level drivers.

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

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Nov 9, 2018	-	Initial version

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

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

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

(Rev.4.0-1 November 2017)

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