Renesas Synergy™ Platform

NetX™ Telnet Client Module Guide

Introduction

This Module Guide will enable you to effectively use a module in your own design. On completion of this guide, you will be able to add this module to your own design, configure it correctly for the target application and write code, using the included Application Project code as a reference and an efficient starting point. References to more detailed API descriptions and suggestions of other Application Projects that illustrate more advanced uses of the module are included in this document and should be valuable resources for creating more complex designs.

The Telnet Protocol (Telnet) protocol is designed for transferring commands and responses between two nodes on the Internet. Telnet is a simple protocol that utilizes reliable Transmission Control Protocol (TCP) services to perform its transfer function. Because of this, Telnet is a highly reliable transfer protocol. The NetX™ Telnet Client is a high-level API for applications planning to implement a Telnet Client.

Note: Except for internal processing, NetX Duo™ Telnet Client is nearly identical in application set up and running a Telnet session as the NetX™ Telnet Client, except where noted.

This document provides an overview of the key elements related to the NetX™ Telnet Client implementation on the Renesas Synergy Platform. Its primary focus is the addition and configuration of the NetX™ Telnet Client module to a Renesas Synergy Platform project. For other details on the operation of this module, consult the “NetX™ Telnet Protocol for Clients User Guide for the Renesas Synergy™ Platform” document. This document is included in the X-Ware™ and NetX™ Component Documents for Renesas Synergy™ zip file. The zip file is available as a download from the Synergy Software Package site (www.renesas.com/synergy/ssp) on the Renesas Synergy Gallery.

Contents

1. NetX™ Telnet Client Module Features

- The NetX™ Telnet Client Module is compliant with RFC854 and related RFCs
- Provides high-level APIs to:
  - Create and delete a Telnet Client instance
  - Connect and disconnect a Telnet Client instance
  - Send to and receive packets from a Telnet Server
  - Support multi-thread operation

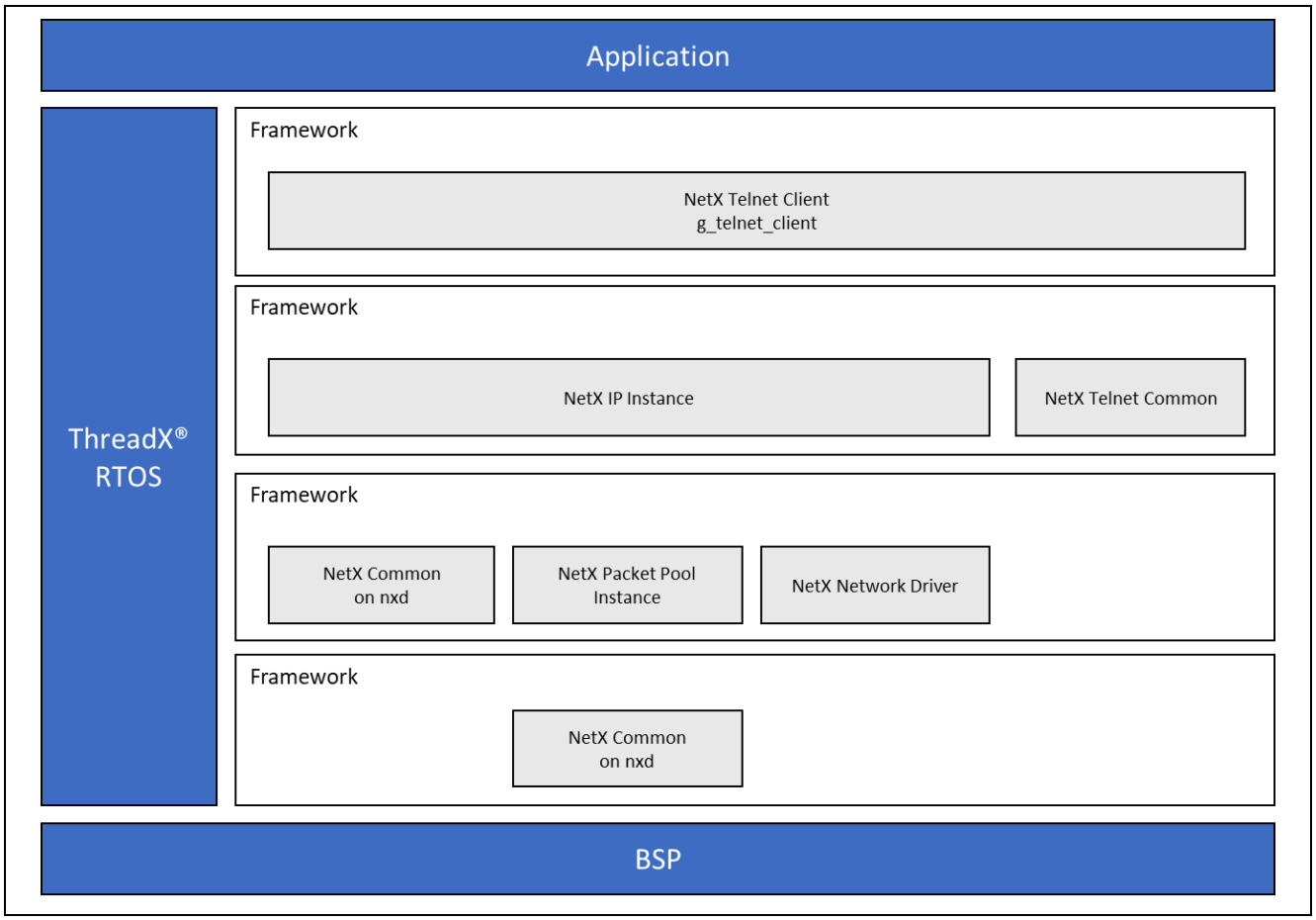

Figure 1. NetX™ Telnet Client Module Organization, Options and Stack Implementations

2. NetX™ Telnet Client Module APIs Overview

The NetX™ Telnet Client Module defines telnet APIs to create, delete, connect, disconnect, receive, and send communications. A complete list of the available APIs, an example API call, and a short description of each can be found in the following table. A table of status return values follows the API summary table.

Table 1. NetX™ Telnet Client Module API Summary

Function Name	Example API Call and Description
nx_telnet_client_connect	`nx_telnet_client_connect(&g_telnet_client0, server_ip_address, server_port, NX_WAIT_FOREVER);` Connect a Telnet Server. Supports only IPv4.
nxd_telnet_client_connect**	`nxd_telnet_client_connect(&g_telnet_client0, &server_ip_address_v6, server_port, NX_WAIT_FOREVER);` Connect to a Telnet Server. Supports IPv4 and IPv6.
nx_telnet_client_create	`nx_telnet_client_create(&g_telnet_client0, “Telnet Client", &g_ip0, 1024);` Create a Telnet Client.
nx_telnet_client_delete	`nx_telnet_client_delete(&g_telnet_client0);` Delete a Telnet Client.
nx_telnet_client_disconnect	`nx_telnet_client_disconnect(&g_telnet_client0, NX_WAIT_FOREVER);` Disconnect from the Telnet Server (IPv4 or IPv6).
nx_telnet_client_packet_receive	`nx_telnet_client_packet_receive(&g_telnet_client0, &packet_ptr, NX_WAIT_FOREVER);` Receive packet from the Telnet Server.
nx_telnet_client_packet_send	`nx_telnet_client_packet_send(&g_telnet_client0, packet_ptr, NX_WAIT_FOREVER);` Send packet to the Telnet Server (IPv4 or IPv6).

Note: For details on operation and definitions for the function data structures, typedefs, defines, API data, API structures and function variables, review the associated Express Logic User’s Manual in the Reference section later in this document.
All the above APIs are available in the NetX Duo™ Telnet Client.

** This API is only available in NetX Duo™ Telnet Client. Please refer to the *NetX Duo™ User Guide for the Renesas Synergy™ Platform* for definition of NetX Duo™ specific data types.

Table 2. Error Status Return Values

Name	Description
NX_SUCCESS	API Call Successful
NX_TELNET_ERROR	Error while creating TCP socket as part of creating the Telnet Client instance
NX_TELNET_NOT_CONNECTED	Client not disconnected
NX_TELNET_NOT_DISCONNECTED	Client socket is not in closed state (cannot make a TCP connection; cannot delete the Telnet Client if the socket is still connected).
NX_TELNET_INVALID_PARAMETER*	Invalid non-pointer input to Telnet Client create
NX_PTR_ERROR*	Invalid pointer input
NX_IP_ADDRESS_ERROR*	Invalid IP address to connect to Telnet Server
NX_CALLER_ERROR*	Invalid caller of this service

Note: Lower level drivers may return Common Error Codes. Refer to the *SSP User’s Manual* API Reference section for the associated module for a definition of all relevant status return values

* These error codes are only returned if error checking is enabled. See the *NetX™ User Guide for the Renesas Synergy™ Platform* or *NetX Duo™ User Guide for the Renesas Synergy™ Platform* for more details on error checking services in NetX™ and NetX Duo™, respectively.

3. NetX™ Telnet Client Module Operational Introduction

In the NetX™ Telnet Client Module, the IP thread task for NetX™ is created and runs, the Telnet Client is created, and the TCP socket for connecting to the Telnet Server is ready for use automatically. The Telnet Client connects to the Server by calling the `nx_telnet_client_connect` API. (This API is available in NetX Duo™ Telnet and is limited to IPv4 TCP connections. In NetX Duo™, the application can also use the `nxd_telnet_client_connect` which supports both IPv4 and IPv6.)

If the connection succeeds, then the Telnet Client should wait to receive a Server 'banner' announcing itself using the `nx_telnet_client_packet_receive` API. Thereafter, the Telnet Client can create packets with single characters of data using the `nx_packet_allocate` and `nx_packet_data_append` API. These packets are sent to the Telnet Server by calling the `nx_telnet_client_packet_send` API.

For details on NetX™ Telnet Client and NetX Duo™ Telnet Client operations, see the X-Ware™ and NetX™ Component Documents for Renesas Synergy™ zip file. The zip file is available as a download from the Synergy Software Package site (www.renesas.com/synergy/ssp) on the Renesas Synergy Gallery. Open the zip file and navigate to relevant User Guides for details. This Telnet Client Module document only provides an introduction to component operations. Module selection, configuration, and example uses are covered in the following sections.

3.1 NetX™ Telnet Client Module Important Operational Notes and Limitations

3.1.1 NetX™ Telnet Client Module Operational Notes

- The NetX Duo™ Telnet Client Module services can be called from multiple threads simultaneously. However, read or write requests for a particular Telnet Client instance should be done in sequence from the same thread.

3.1.2 NetX™ Telnet Client Module Limitations

- The Telnet Client does not support Telnet negotiation or send IAC and command code sequences.
- See the SSP Release Notes for any additional operational limitations for this module.

4. Including the NetX™ Telnet Client Module in an Application

This section describes how to include the NetX™ Telnet Client Module in an application using the SSP configurator.

Note: This section assumes you are familiar with creating a project, adding threads, adding a stack to a thread and configuring a block within the stack. If you are unfamiliar with any of these items, refer to the first few chapters of the SSP User's Manual to learn how to manage each of these important steps in creating SSP based applications.

To add the NetX™ Telnet Client Module to an application, simply add it to a thread using the Stacks Selection Sequence given in the table below. (The default name for the NetX™ Telnet Client is g_telnet_client0 is listed in the following table. This name can be changed in the associated Properties window.)

Table 3. NetX™ Telnet Client Module Selection Sequence

Resource	ISDE Tab	Stacks Selection Sequence
g_telnet_client0 NetX™ Telnet Client	Threads	New Stack> X-Ware> NetX™> Protocols> NetX™ Telnet Client
g_telnet_client0 NetX Duo™ Telnet Client	Threads	New Stack> X-Ware> NetX Duo™> Protocols> NetX Duo™ Telnet Client

When the NetX™ Telnet Client Module on g_telnet_client0 is added to the Thread Stack (see the following figure), the configurator automatically adds the needed lower level drivers. Any drivers that need additional configuration information are highlighted in red. Modules with a gray band are individual modules that stand alone. Modules with a blue band are shared or common and need only be added once, since they can be used by multiple stacks. Modules with a pink band can require the selection of lower level drivers. Sometimes these are optional or recommended and they are indicated in the block with the inclusion of this text. If the addition of lower level drivers is required, the module description will include "Add" in the text. Clicking on any pink banded modules will bring up the "New" icon and show possible choices.

The Telnet Client application needs to allocate packets to send to the Telnet Server. The following figure shows that it can use the same packet pool as the IP instance, g_packet_pool0. Or, it can create a separate packet pool specifically for Telnet message it is sending. To do so, select the **Add NetX™ Packet Pool** with the pink band connected to the NetX™ Telnet Client Module block and choose **New**. A separate packet pool for the Telnet Client might be more efficient use of memory, if the expected Telnet message packets carry a much smaller payload than the payload of the IP instance packet pool.

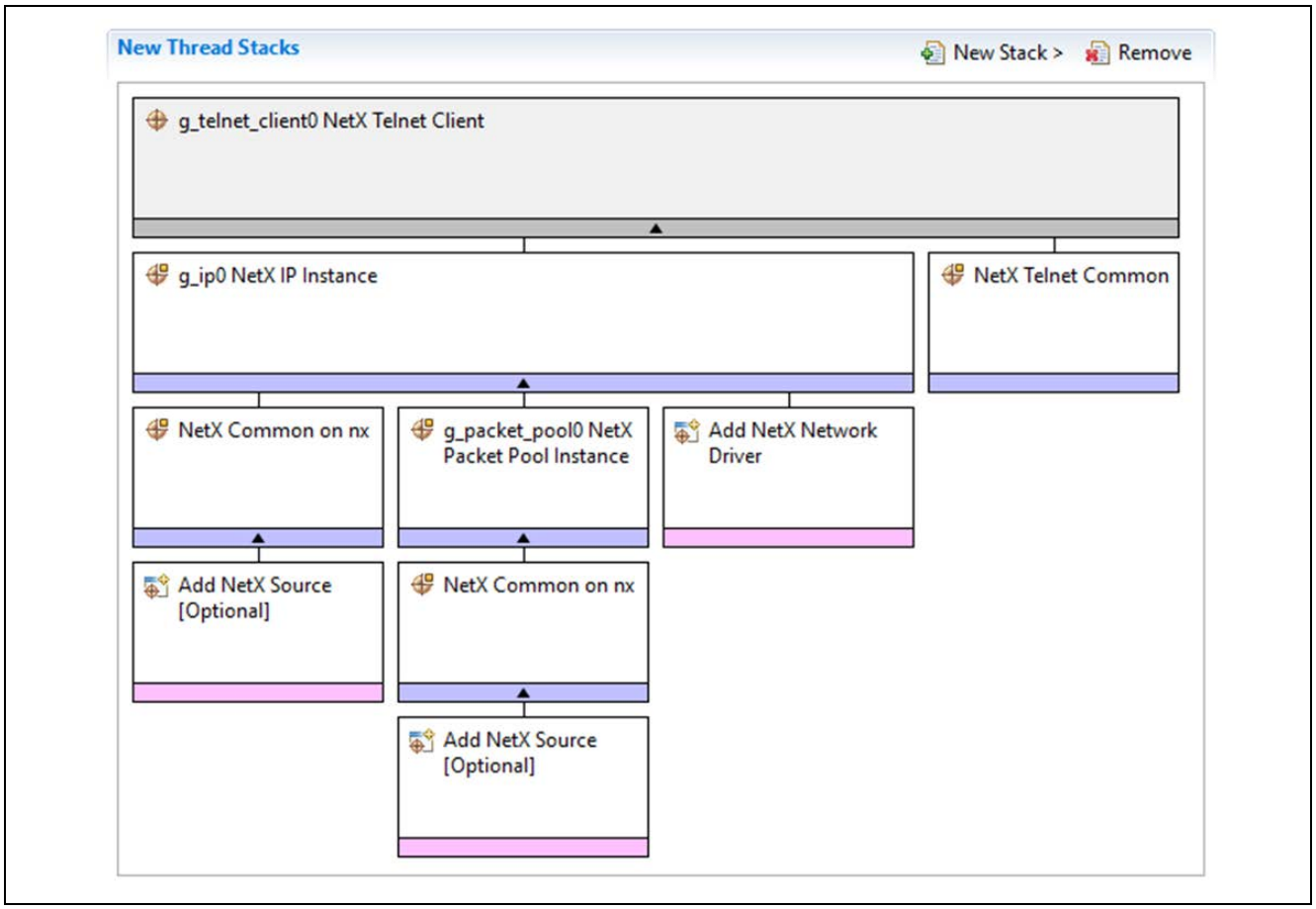

Figure 2. NetX™ Telnet Client Module Stack

5. Configuring the NetX™ Telnet Client Module

The NetX™ Telnet Client module must be configured by the user for the desired operation. The SSP configuration window automatically identifies, by highlighting the block in red, any required configuration selections, such as interrupts or operating modes, which must be configured for lower level modules, for successful operation. 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 NetX™ Telnet Client and explore the property settings in parallel with reviewing the Configuration Table Settings. This can help orient you and be a useful 'hands-on' approach to learning the ins and outs of developing with SSP.

Table 4. Configuration Settings for NetX™ Telnet Client Module

ISDE Property	Value	Description
Name	g_telnet_client0	Module name
TCP Socket Window Size in Bytes	1024	TCP socket receive window size selection

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

In some cases, settings other than the defaults for stack modules can be desirable. For example, it might be useful to select different IP addresses and subnet masks. The configurable properties for the lower level stack modules are given in the following sections for completeness and as a reference.

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

5.1 Configuration Settings for the NetX™ Telnet Client Module Low-Level Module

Typically, only a small number of settings must be modified from the default for the IP layer, and these are indicated via the red text in the Thread Stack block. Notice that some of the configuration properties must be set to a certain value for proper framework operation and are locked to prevent user modification. The following table identifies all the settings within the properties section for the module.

Table 5. Configuration for the NetX™ IP Instance

ISDE Property	Value	Description
Name	g_ip0	Module name
IPv4 Address (use commas for separation)	192,168,0,2	IPv4 Address selection
Subnet Mask (use commas for separation)	255,255,255,0	Subnet Mask selection
IPv6 Global Address (use commas for separation)	0x2001,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x1	IPv6 Global Address selection, valid for NetX Duo™ only
IPv6 Link Local Address (use commas for separation; all zeros means user MAC address)	0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0	IPv6 Local Address selection, valid for NetX Duo™ only
IP Helper Thread Stack Size (bytes)	2048	IP Helper Thread Stack Size (bytes) selection
IP Helper Thread Priority	3	IP Helper Thread Priority selection
ARP	Enable	ARP selection
ARP Cache Size in Bytes	520	ARP Cache Size in Bytes selection
Reverse ARP	Enable, Disable (Default: Disable)	Reverse ARP selection
TCP	Enable	TCP selection
UDP	Enable, Disable (Default: Enable)	UDP selection
ICMP	Enable, Disable (Default: Enable)	ICMP selection
IGMP	Enable, Disable (Default: Enable)	IGMP selection
IP fragmentation	Enable, Disable (Default: Disable)	IP fragmentation selection

Table 6. Configuration for the NetX™ Common

ISDE Property	Value	Description
No configurable Settings		

Table 7. Configuration for the NetX™ Packet Pool Instance

ISDE Property	Value	Description
Name	g_packet_pool0	Module name
Packet Size in Bytes	512(Default: 640)	Packet size selection
Number of Packets in Pool	16	Number of packets in pool selection

Table 8. Configuration for the NetX™ Port ETHER

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	Enable or disable the parameter checking
Channel 0 Phy Reset Pin	IOPORT_PORT_09_PIN_03	Channel 0 Phy reset pin selection
Channel 0 MAC Address High Bits	0x00002E09	Channel 0 MAC address high bits selection
Channel 0 MAC Address Low Bits	0x0A0076C7	Channel 0 MAC address low bits selection
Channel 1 Phy Reset Pin	IOPORT_PORT_08_PIN_06 (Default: IOPORT_PORT_07_PIN_06)	Channel 1 Phy reset pin selection
Channel 1 MAC Address High Bits	0x00002E09	Channel 1 MAC address high bits selection
Channel 1 MAC Address Low Bits	0x0A0076C8	Channel 1 MAC address low bits selection
Number of Receive Buffer Descriptors	8	Number of receive buffer descriptors selection
Number of Transmit Buffer Descriptors	32	Number of transmit buffer descriptors selection
Ethernet Interrupt Priority	Priority 0 (highest)-15 (lowest), Disabled (Default: Disabled)	Ethernet interrupt priority selection
Name	g_sf_el_nx	Module name
Channel	1 (Default: 0)	Channel selection
Callback	NULL	Callback selection

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

5.2 NetX™ Telnet Client Module Clock Configuration

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

5.3 NetX™ Telnet Client Module Pin Configuration

The ETHERC peripheral module uses pins on the MCU to communicate to external devices. I/O pins must be selected and configured as required by the external device. The following table indicates how to select these pins within the SSP configuration window and the subsequent table is an example selection for ETHERC.

Note: For some peripherals, the Operation Mode selection mode determines what peripheral signals are available and thus what MCU pins are required.

Table 9. Pin Selection Sequence for ETHERC Module

Resource	ISDE Tab	Pin selection Sequence
ETHERC	Pins	Select Peripherals > Connectivity:ETHERC > ETHERC1.RMII

Note: This selection sequence assumes ETHERC1 is the desired hardware target for the driver.

Table 10. Pin Configuration Settings for ETHERC1

Pin Configuration Property	Value	Description
Operation Mode	Disabled, Custom, RMII (Default: Disabled)	Select RMII as the Operation Mode for ETHERC1
Pin Group Selection	Mixed, _A only (Default: _A only)	Pin group selection
REF50CK	P701	REF50CK Pin
TXD0	P700	TXD0 Pin
TXD1	P406	TXD1 Pin
TXD_EN	P405	TXD_EN Pin
RXD0	P702	RXD0 Pin
RXD1	P703	RXD1 Pin
RX_ER	P704	RX_ER Pin
CRS_DV	P705	CRS_DV Pin
MDC	P403	MDC Pin
MDIO	P404	MDIO Pin

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

6. Using the NetX™ Telnet Client in an Application

The NetX™ Telnet Client Module does not need the usual initialization by an application. A configurator generates initialization process. The user application only needs Telnet communication processing.

The typical steps in using the NetX™ Telnet Client Module in an application are:

1. Use the `nx_ip_status_check` API to check that the IP instance is initialized and the application can start using NetX™ services.
2. Connect to the Telnet Server via the `nx_telnet_client_connect` API. For NetX Duo™ the preferred API is `nxd_telnet_client_connect`.
3. Receive the Telnet Server welcome banner using `nx_telnet_client_packet_receive` API [Optional]
4. Create a packet to send with the `nx_packet_allocate` and `nx_packet_data_append` APIs.
5. Send the packet using the `nx_telnet_client_packet_send` API
6. Receive the Server's response packet using the `nx_telnet_client_receive` API.
7. Disconnect communication using the `nx_telnet_client_disconnect` API
8. Delete the instance using the `nx_telnet_client_delete` API when done sending and receiving.

The following operational flow diagram shows these common steps.

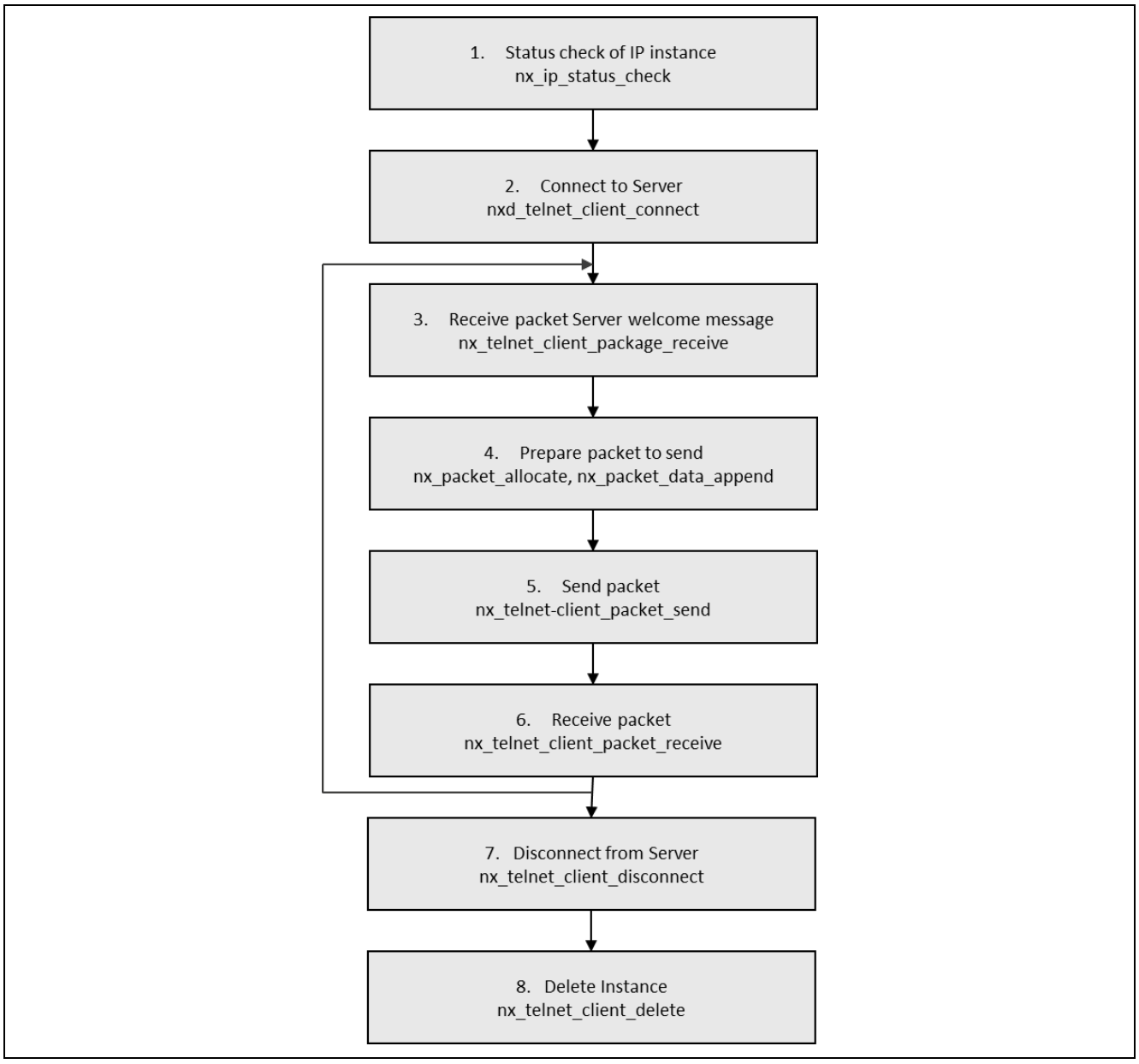

Figure 3. Flow Diagram of a Typical NetX™ Telnet Client Module Application

7. The NetX™ Telnet Client Module Application Project

The Application Project associated with this Module Guide demonstrates these operational flow steps in a full design. The project can be found using the link provided in the Reference Section at the end of this document. You may want to import and open the Application Project within ISDE and view the configuration settings for the NetX™ Telnet Client Module. You can also read over the code, in `Telnet_Client_MG.c`, which is used to illustrate the NetX™ Telnet Client APIs in a complete design.

The Application Project demonstrates the typical use of the NetX™ Telnet Client APIs. The main thread entry of the application project uses the run_telnet_client_session function in the Telnet_Client_MG.c file to connect to a Telnet Server and repeats sending and receiving data. This application connects to the well-known telnet port 23 and receives echo characters. The table below identifies the target versions for the associated software and hardware used by the Application Project.

Table 11. Software and Hardware Resources Used by the Application Project

Resource	Revision	Description
e² studio	7.3.0	Integrated Solution Development Environment
SSP	1.6.0	Synergy Software Platform
IAR EW for Renesas Synergy	8.23.3	IAR Embedded Workbench® for Renesas Synergy™
SSC	7.3.0	Synergy Standalone Configurator
SK-S7G2	v3.0 to v3.3	Starter Kit
Telnet Server	-	This can be a PC or another board running the telnet server module. (Note: Windows 10 does not provide a telnet server.)

Figure 4 and Figure 5 show a simple flow diagram of the NetX™ Telnet Client and the NetX Duo™ Telnet Client Module Application project.

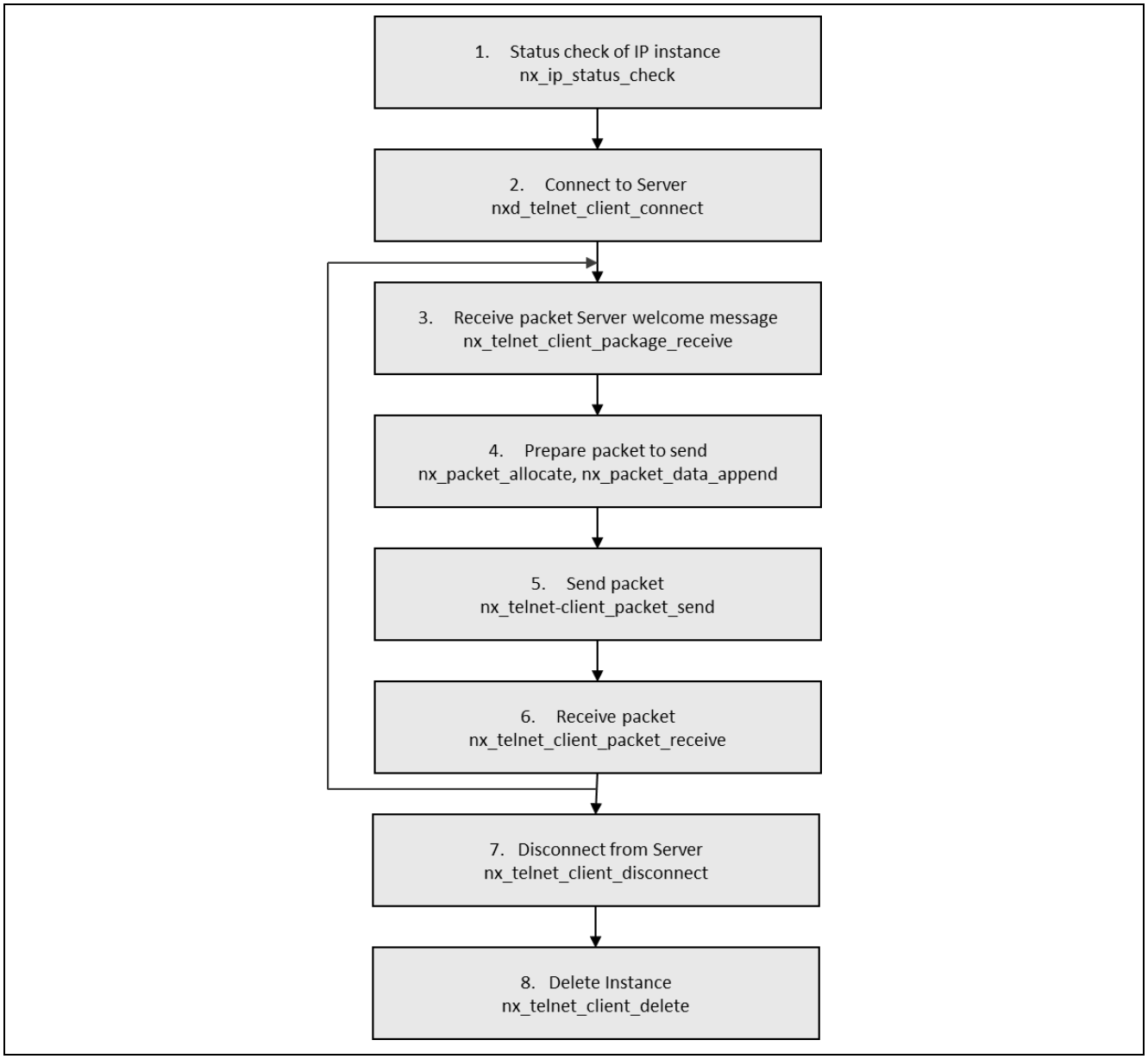

Figure 4. NetX™ Telnet Client Module Application Project Flow Diagram

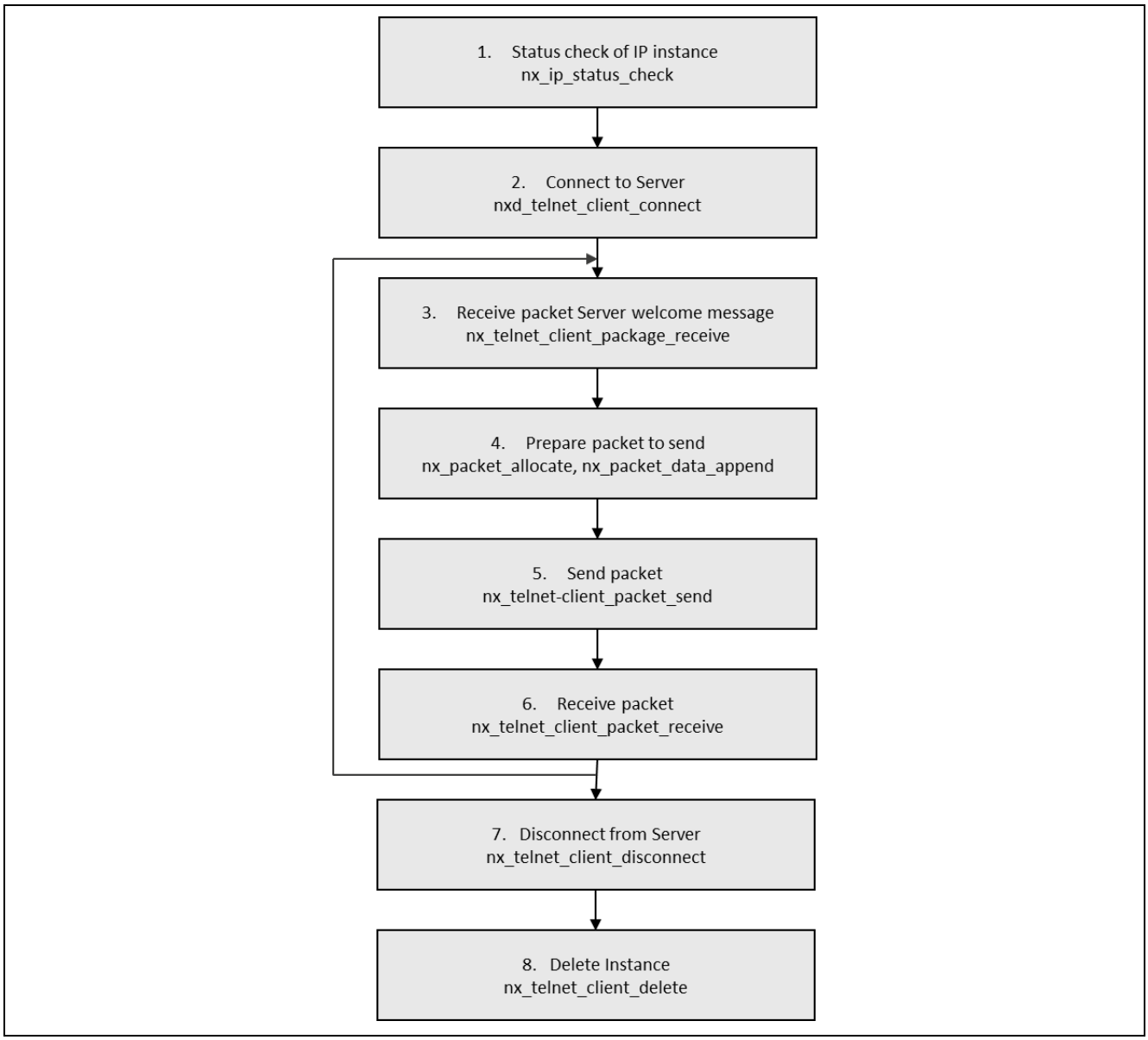

Figure 5. NetX Duo™ Telnet Client Module Application Project Flow Diagram

The `telnet_client_thread_entry.c` file is an auto-generated file that is located in the project once it has been imported into the ISDE. The `Telnet_Client_MG.c` file also located in the project has the actual logic needed to run a Telnet Client session (see Figure 4). You can open these files within the ISDE, read along, and learn how to identify key uses of the APIs.

If using NetX Duo™ Telnet Client, regardless of IPv4 or IPv6 connections, the developer must define NETX_DUO in the list of preprocessors for the project. Right-click the project in e^2 studio **-> Properties -> C/C++ Build -> Settings -> Cross ARM C Compiler -> Preprocessor**. Click the **(+)** icon to add NETX_DUO. If developing a Telnet Client for IPv6, make sure USE_IPV6 is defined in the `Telnet_Client_MG.h` file.

For both NetX™ Telnet Client and NetX Duo™ Telnet Client, define the option SEMI_HOSTING in `Telnet_Client_MG.h` for debug output to be directed to the Renesas Debug Virtual Console.

The `telnet_client_thread_entry` calls `run_telnet_client_session` that sets up and processes a connection to Telnet Server, and transmits and processes Telnet data on the “echo” port. The application repeats transmission and reception up to the time specified in the input to run_telnet_client_session. After that, it disconnects and deletes the instance.

Note that it is the application’s responsibility to release a received packet received from the Telnet Server. In `run_telnet_client_session` function, if `nx_telnet_client_packet_receive` returns a successful

result, it calls `nx_packet_release` on the received packet. If it receives an unsuccessful status return, no packet was received, so no packet needs to be released.

Similarly, if `nx_telnet_client_packet_send` returns an unsuccessful result, the application must release the packet it allocated but did not send to the Server. If the send call is successful, the underlying NetX™ thread task will release the packet.

A few key properties in this Application Project to support are required for correct operation. There are also a few physical properties of the target board and MCU that need to be modified from default values. The following table lists the properties with the values set for this specific project. You can also open the Application Project and view these settings in the property window as a hands-on exercise.

Table 12. NetX™ Telnet Client Module Configuration Settings for the Application Project

ISDE Property	Value Set
NetX™ Port ETHER on sf_el_nx Channel 1 Phy Reset Pin	IOPORT_PORT_08_PIN_06 (only for SK-S7G2)
NetX™ Port ETHER on sf_el_nx Ethernet Interrupt Priority	Priority 4
NetX™ Port ETHER on sf_el_nx Channel	1
NetX™ IP Instance IPv4 Address (use commas for separation)	<set to device address on the local network>
Subnet Mask (use commas for separation)	<set to subnet mask on the local network>
NetX™ IP Instance IPv6 Global Address (use commas for separation)	<set to device address on the local network>

8. Customizing the NetX™ Telnet Client for a Target Application

The only configuration specific to NetX™ or NetX Duo™ Telnet Client likely to be changed are the IPv4 Client and, if applicable, the IPv6 Client. Otherwise, the Telnet Client does not have any other options to be configured.

One possible customization might be in the IP layer, by sharing a packet pool with the IP instance, such as g_packet_pool0. This would reduce memory usage in memory constrained systems.

Some configuration settings are normally changed by the developer from those shown in the application project. For example, the user can easily change the configuration settings for the NetX™ IP Instance. They can set a static IP of their choice or use a NetX™ DHCP Client to obtain an IP address provided by a DHCP server in the network.

9. Running the NetX™ Telnet Client Module Application Project

To run the NetX™ Telnet Client Module Application project and to see it executing on a target kit, you can simply import it into your ISDE, compile and run debug. See the *Renesas Synergy™ Project Import Guide* (r11an0023eu0121-synergy-ssp-import-guide) included in the package for instructions on importing the project into e^2 studio ISDE, or IAR EW for Synergy, building the project, and running the application.

To implement the NetX™ Telnet Client Module application in a new project, use the following steps to define, configure, auto-generate the files, add code, compile, and debug the project on the target kit. This hands-on approach helps make the development process with SSP more practical than simply reading this guide.

Note: The following steps provide sufficient detail for someone experienced with the basic flow through the Synergy development process. If these steps are not familiar, refer to the first few chapters of the *SSP User's Manual* for a description of how to accomplish these steps.

To create and run the NetX™ Telnet Client Application Project simply follow these steps:

1. Create a new Renesas Synergy project for S7G2-SK called, **Telnet_Client_MG_Project**.
2. Select the **Threads** tab.
3. Add a thread element called, **telnet_client_thread**. Stack size should be 2048 and priority set to 3 (lower than the NetX IP thread task priority).
4. In the **Telnet Client Threads Stack** pane, click on the **(+)** icon and choose **X-ware -> NetX -> Protocols -> NetX Telnet Client**.

5. Click the **(+)** icon and select **X-ware -> NetX -> Net Packet Pool** to create a separate packet pool used only by the Telnet Client to transmit packets (optional).
6. Click the **Generate Project Content** button.
7. Add the code from the supplied project file `Telnet_Client_MG.c`. There are two ways to do this:

   A. Add a new file and copy the code from the `Telnet_Client_MG.c` file.
   B. Copy the source code from `Telnet_Client_MG.c` directly into the auto-generated file `telnet_client_thread_entry` (if your thread is named telnet_client_thread) and fill in the server IP address on line 34.
8. Make sure NETX_DUO is not defined in the list of preprocessor symbols; by default, USE_IPV6 is not and should not be defined in the header file.
9. Connect to the host PC via a micro USB cable to J19 on the SK-S7G2 target.
10. Connect the SK-S7G2 device to the telnet server via RJ-45 cable at the J11 connector.
11. Start the Telnet Server on a PC in the local network(If available on your PC or in your network) or Telnet Server can be run on another Synergy board (R11AN0118EU — **NetX™ Telnet Server Module Guide - Application Project**)..
12. Run the application and start debugging.
13. The output can be viewed in the Renesas Debug Virtual Console.

```
Connected to Telnet Server.
Send H to server
Send e to server
Send l to server
Send l to server
Send o to server
Send   to server
Send S to server
Send e to server
Send r to server
Send v to server
Send e to server
Send r to server
Disconnecting from Telnet Server.
Client session completed successfully!
```

Figure 6. Example Output from NetX™ Telnet Client Application Project

To create and run the NetX Duo™ Telnet Client Application Project simply follow these steps:

1. Create a new Renesas Synergy project for S7G2-SK called, **Telnet_Client_MG_Project**.
2. Select the **Threads** tab.
3. Add a thread element called **telnet_client_thread**. Stack size should be 2048 and priority set to 3 (lower than the NetX Duo IP thread task priority).
4. In the Telnet Client Threads Stack pane, click on the **(+)** icon and choose **X-ware -> NetX Duo -> Protocols -> NetX Duo Telnet Client**.
5. Click on the **(+)** icon and choose **X-ware -> NetX Duo-> Net Duo Packet Pool** to create a separate packet pool used only by the Telnet Client to transmit packets (not required).
6. Click on the **Generate Project Content** button.
7. Add the code from the supplied project file `Telnet_Client_MG.c`. There are two ways to do this:

   A. Add a new file and copy the code from the `Telnet_Client_MG.c` file.
   B. Copy the source code from `Telnet_Client_MG.c` directly into the auto-generated file `telnet_client_thread_entry` (if your thread is named **telnet_client_thread**) and fill in the server IP address (IPv4 or IPv6) on line 34 or line 76-79.
8. Make sure NETX_DUO is defined in the list of preprocessors of the project.

9. To run NetX Duo™ Telnet Client over IPv6, define USE_IPV6 in the header file. To run NetX Duo™ Telnet Client over IPv4, leave this undefined (default it is not defined).
10. Connect to the host PC via a micro USB cable to J19 on SK-S7G2.
11. Connect the SK-S7G2 to the telnet server via RJ-45 cable at the J11 connector.
12. Start the Telnet Server on a PC in the local network(If available on your PC or in your network) or Telnet Server can be run on another Synergy board (R11AN0118EU - **NetX™ Telnet Server Module Guide - Application Project**)..
13. Run the application and start debugging.
14. The output can be viewed in the Renesas Debug Virtual Console.

```
Connected to Telnet Server.
Send H to server
Send e to server
Send l to server
Send l to server
Send o to server
Send   to server
Send S to server
Send e to server
Send r to server
Send v to server
Send e to server
Send r to server
Disconnecting from Telnet Server.
Client session completed successfully!
```

Figure 7. Example Output from NetX Duo™ Telnet Client Application Project

10. NetX™ Telnet Client Module Conclusion

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

11. NetX™ Telnet Client Module Next Steps

After you have mastered a simple NetX™ Telnet Client Module project, you may want to review a more complex example. The NetX™ Telnet Client has only basic services and not many configurable options, so it is an inherently simple module component to develop. Other Application Projects and Application Notes that demonstrate Telnet Client use can be found as described in the Reference section at the end of this document.

You may find that the setting a DHCP Client is a better fit for your target application than setting a static IP. The *NetX™ DHCP Client Module Guide* illustrates how to use it to obtain a dynamic provided IP address from a DHCP Server.

12. NetX™ Telnet Client Module Reference Information

The SSP User Manual is available in HTML in the SSP distribution package, and as a pdf from the Synergy Gallery: www.renesas.com/synergy/ssp.

Links to all the most up-to-date NetX Telnet Client Module reference materials and resources are available on the Synergy Knowledge Base: https://en-support.renesas.com/knowledgeBase/16977462.

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	Dec.01.17	—	Initial release
1.01	Dec.12.19	—	Updated to v1.5.0
1.02	Mar.22.19	—	Updated to v1.6.0

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.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Corporate Headquarters

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

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

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/.