Renesas Synergy™ Platform

NetX™ Telnet Server Module Guide

Introduction

This module guide will enable you to effectively use a module in your own design. Upon completion of this guide, you will be able to add this module to your own design, configure it correctly for the target application, and write code using the included application project code as a reference and 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) is a protocol 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. Telnet is also one of the most used application protocols.

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

This overview covers key elements related to the NetX Telnet Server implementation on the Renesas Synergy™ Platform. The primary focus is adding and configuring the NetX Telnet Server module to a Renesas Synergy Platform project. For more details on the operation of this module, consult *NetX Telnet Server Guide* for the Renesas Synergy™ Platform. This document is part of an X-Ware™ Component Documents for Renesas Synergy™ zip file available from the Renesas Synergy Gallery (https://www.renesas.com/synergy/software).

Contents

1. NetX Telnet Server Module Features

- The NetX Telnet Server module is compliant with RFC854 and related RFCs.
- Provides high-level APIs to:
  - Support multi-thread operation
  - Support callbacks for common functions
    - Authentication
    - New Connection
    - Receive Data
    - End Connection
  - Support some Option negotiation
    - Echo
    - Suppress Go Ahead

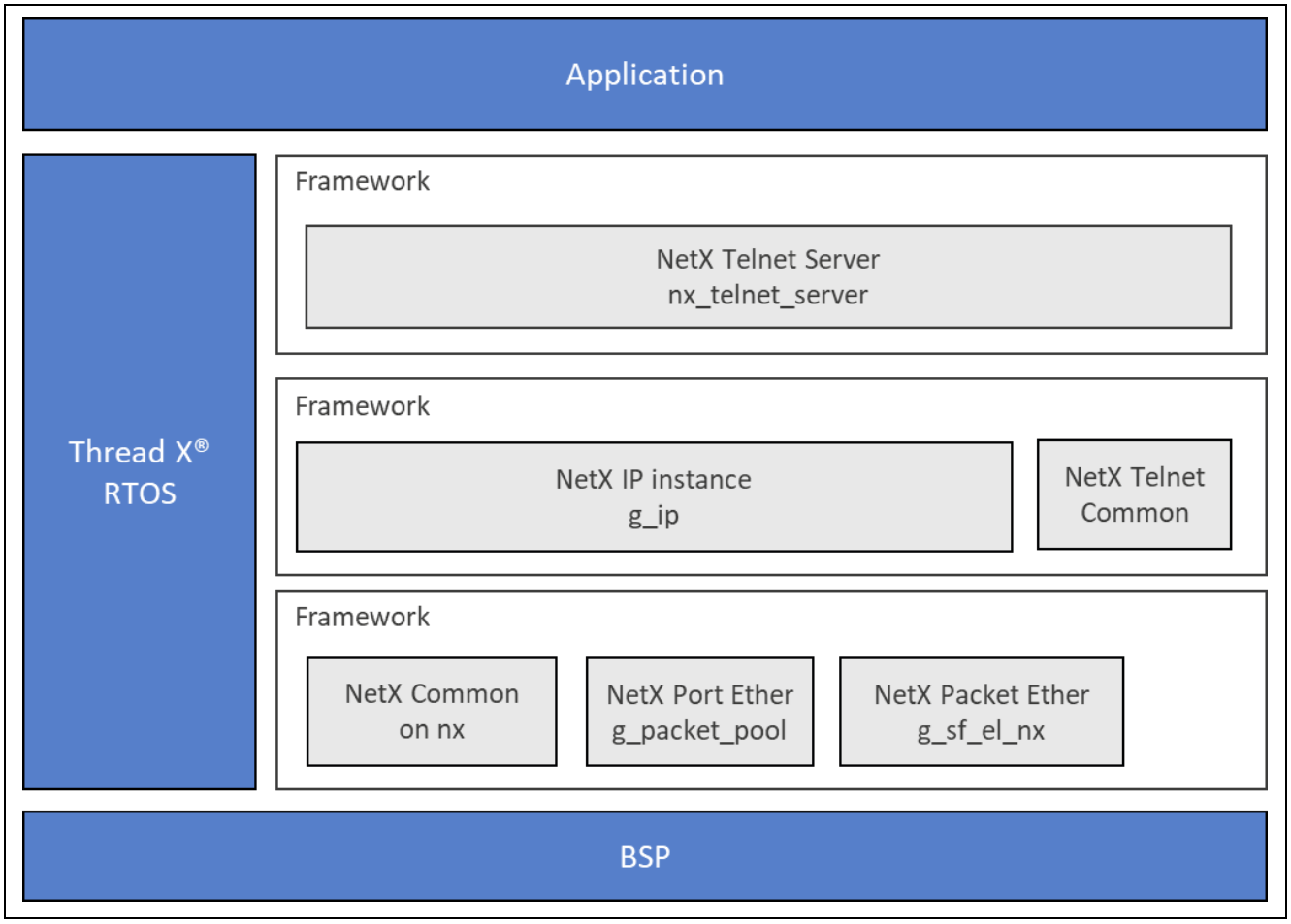

Figure 1. NetX Telnet Server Module Organization, Options and Stack Implementations

2. NetX Telnet Server APIs Overview

The NetX Telnet Server module defines APIs for creating, deleting, sending packets, starting, and stopping. 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 Server Module API Summary

Function Name	Example API Call and Description
nx_telnet_server_create	`nx_telnet_server_create(&my_server, "Telnet Server", &ip_0,pointer, 2048, telnet_new_connection, telnet_receive_data, telnet_connection_end);` Create a Telnet Server.
nx_telnet_server_delete	`nx_telnet_server_delete(&my_server);` Delete a Telnet Server.
nx_telnet_server_disconnect	`nx_telnet_server_disconnect(&my_server, 2);` Disconnect the Telnet Client specified by the client list index (2nd input).
nx_telnet_server_get_open_connection_count	`nx_telnet_server_get_open_connection_count(&my_server, &conn_count);` Retrieve the number of open connections.
nx_telnet_server_packet_send	`nx_telnet_server_packet_send(&my_server, 2, my_packet, 100);` Send packet to Telnet Client specified by client list index (second input).
nx_telnet_packet_pool_set	`nx_telnet_server_packet_pool_set(&my_server, &telnet_server_packet_pool);` Set packet pool as Telnet Server packet pool.
nx_telnet_server_start	`nx_telnet_server_start(&my_server);` Start the Telnet Server.
nx_telnet_server_stop	`nx_telnet_server_stop(&my_server);` Stop the Telnet Server.

Note: For more complete descriptions of 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* accessible as described in the Reference section later in this document.

All the above APIs are available in the NetX Duo Telnet Server.

Table 2. Status Return Values

Name	Description
NX_SUCCESS	Successful telnet function
NX_PTR_ERROR*	Invalid Server, IP, stack, or application callback pointers
NX_CALLER_ERROR*	Invalid caller of this service
NX_OPTION_ERROR*	Invalid logical connection
NX_IP_ADDRESS_ERROR*	Invalid IP address
NX_TELNET_FAILED	Server packet send failed
NX_TELNET_NO_PACKET_POOL	Cannot start Telnet Server, no packet pool available
NX_TELNET_NOT_CONNECTED	Cannot disconnect Telnet Server because it is not connected
NX_TELNET_NOT_DISCONNECTED	Cannot connect or delete Telnet Server because it is not disconnected

Note: Lower-level drivers may return common error codes. Refer to the *SSP User's Manual* API References for the associated module for a definition of all relevant status return values.

*These error codes are only returned if error-checking is enabled. Refer to the *NetX (or NetX Duo) User's Guide* for the Renesas Synergy Platform with more details on error-checking services in NetX.

3. NetX Telnet Server Module Operational Overview

The NetX Telnet package requires that a NetX IP instance has already been created. In addition, TCP must be enabled on that same IP instance. The Telnet Client portion of the NetX Telnet package has no further requirements. It also requires complete access to TCP well-known port 23 for handling all Telnet Client requests. The Telnet Server keeps a list of client connections and uses an index into this list to specify certain clients when needed. The size of this list is set in the **Maximum clients to server simultaneously** property.

The Telnet Server can be enabled for limited **Option negotiation** with the Telnet Client. To enable this feature, set the **Option negotiation** to enable. If this feature is enabled, the Telnet Server needs a packet pool. The application can set the packet payload and number of packets; `packet_size_in` the pool and Total `packet_pool_size` Properties, respectively and let the Telnet Server create the packet pool. When the Telnet Server is deleted, the packet pool is deleted with it.

Alternatively, it can create the packet pool directly and be set as the Telnet Server packet pool by **1)** enabling **Use application packet pool** or by **2)** creating the packet pool by calling the NetX `nx_packet_pool_create` API and 3) setting the packet pool in the Telnet Server using the `nx_telnet_server_packet_pool_set` API. When the Telnet Server is deleted, the application must delete the packet pool directly (`nx_packet_pool_delete` API).

Telnet New Connection Callback

The NetX Telnet Server calls the new connection callback function when a new Telnet Client request is received. The callback function is set in the NetX Telnet Server **Name of Client Connect Callback Function** property. Actions of the **new connection** callback include sending a banner or prompt to the client. It could also include a prompt for login information if authentication is required. The second argument of the new connection callback specifies the client is connecting.

Telnet Receive Data Callback

The NetX Telnet Server Module calls the data received callback function when a new Telnet Client data is received. The second input of the callback is an index into the Telnet Server's list of clients; the Telnet Server knows which client wants to disconnect. The callback function is set in the **Name of Receive Data Callback Function** property. Typical actions of the receive data callback include echoing the data back and/or parsing the data and providing data because of interpreting a command from the client.

Note that this callback routine must release the received packet.

Telnet End Connection Callback

The NetX Telnet Server calls the end connection callback function when it receives a Telnet Client disconnect request. The second input of the callback is an index into the Telnet Server's list of clients; the Telnet Server knows which Client wants to disconnect. The callback function is set in the **Name of Client Disconnect Callback Function** property. Typical actions of the end connection callback include cleaning up resources used for the Telnet Client session.

Telnet Option Negotiation

Upon making a connection with the Telnet Client, the Telnet Server will send out this set of Telnet options to the client if it has not received option requests from the client:

```
will echo
don't echo
will sga
```

When it receives Telnet data from the client, the Telnet Server checks if the first byte is the IAC (Interpret as Command) code; if so, it will process all the options in the client packet. Options not in the list above are not supported and will be ignored.

3.1 NetX Telnet Server Module Important Operational Notes and Limitations

3.1.1 NetX Telnet Server Module Operational Notes

- For the connect callback, the application can use any packet pool it has created or the IP default packet pool. If `nx_telnet_server_packet_send` fails, the callback must release that packet.
- The number of active client connections can be obtained at any time by calling the `nx_telnet_server_get_open_connection_count` API.
- The Telnet Server thread task periodically checks the time remaining on each client connection inactivity timeout. If the timeout has expired, the client connection is dropped. To set the length of the inactivity timeout, set the value of the Client inactivity timeout property of the Telnet Server to the desired value. The interval that the Telnet Server thread task checks the inactivity timeout is the Timeout check period property; this must be less than the client inactivity timeout.
- The Telnet Server can be stopped using the `nx_telnet_server_stop` API and restarted using the `nx_telnet_server_start` API. When the Telnet Server is stopped, all client connections are dropped, and the server stops listening on the Telnet port.
- Deleting the Telnet Server is like stopping the Telnet Server, but additionally releases all resources used for the Telnet Server; timer, thread, TCP socket and if created by the Telnet Server, the Telnet Packet pool.
- The interpretation and response to Telnet Client commands, indicated by a byte with the value of 255, is the responsibility of the application.

3.1.2 NetX Telnet Server Module Limitations

- The NetX Telnet Server supports only a limited set of Telnet Option commands.
- Refer to the most recent *SSP Release Notes* for any additional operational limitations for this module.

4. Including the NetX Telnet Server Module in an Application

This section describes how to include the NetX Telnet Server 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 Server module to an application, add it to a thread using the stacks selection sequence given in the following table. The default name for the NetX Telnet Server is `g_telnet_server0`. This name can be changed in the associated **Properties** window.

Table 3. NetX Telnet Server Selection Sequence

Resource	ISDE Tab	Stacks Selection Sequence
g_telnet_server0 NetX Telnet Server	Threads	New Stack> X-Ware> NetX> Protocols> NetX Telnet Server
g_telnet_server0 NetX Duo Telnet Server**	Threads	New Stack> X-Ware> NetX Duo> Protocols> NetX Duo Telnet Server

**Stack Selection for NetX Duo Telnet Server

When the NetX Telnet Server module is added to the thread stack as shown in the following figure, the configurator automatically adds the needed lower-level drivers and NetX components. Any drivers or components that need additional configuration information will be box text 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 this is 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 then will show the possible choices.

The Telnet Server application will need to allocate packets to send to the Telnet Client if Telnet options are enabled; it can use the same packet pool as the IP instance (`g_packet_pool0` in the following figure) or

create a separate packet pool specifically for Telnet Server message sending. To do so, select the **Add NetX Packet Pool** with the pink band connected to the NetX Telnet Server block and choose **New**. A separate packet pool for the Telnet Server 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.

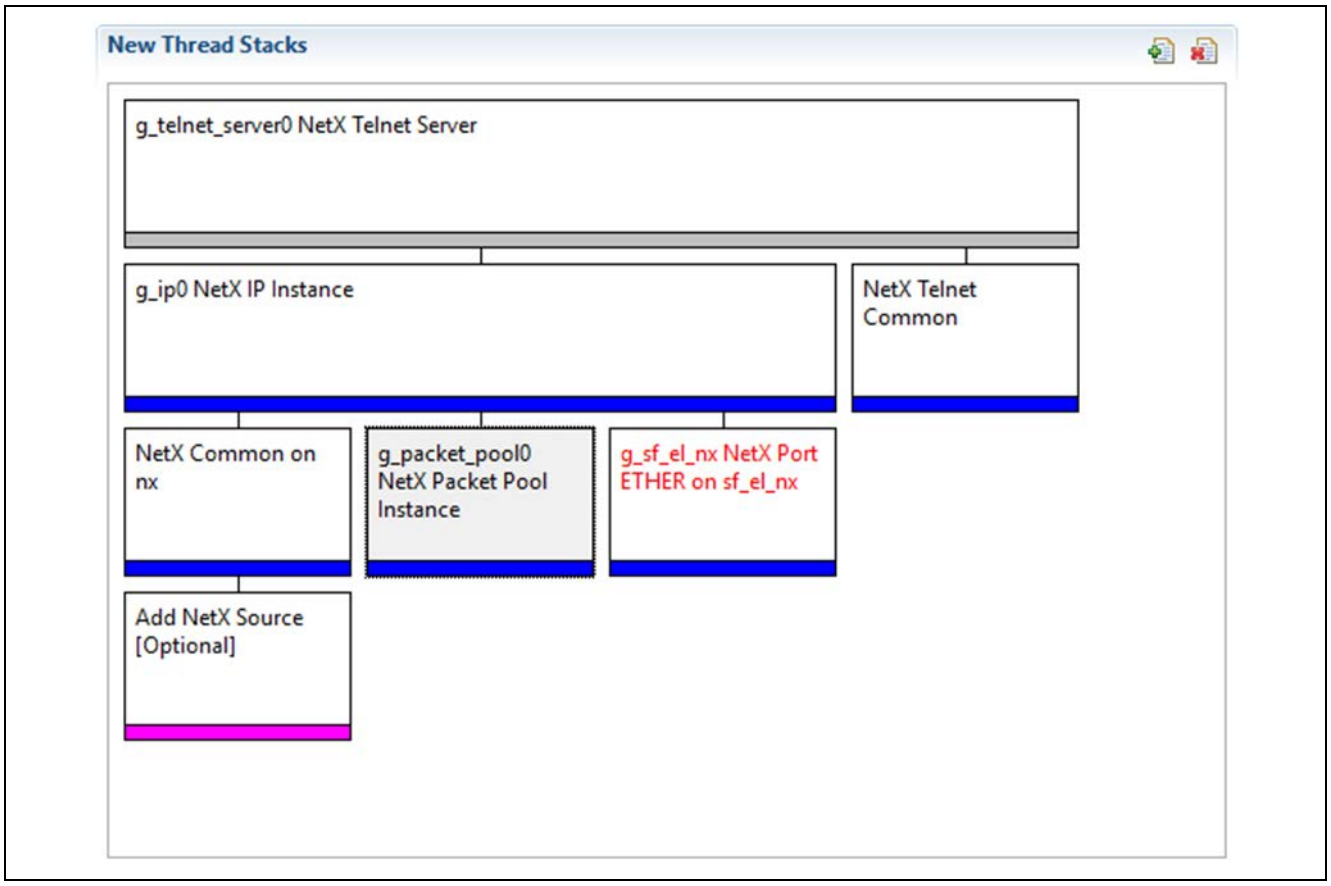

Figure 2. NetX Telnet Server Module Stack

5. Configuring the NetX Telnet Server Module

The NetX Telnet Server 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. 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 **Properties** 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.

It is recommended to change the default setting of the **Timeout check period** property to an even fraction of Client inactivity timeout. This is the interval on which the Telnet Server checks if a client connection inactivity timeout has expired. If Client inactivity timeout is set to 600, a reasonable Timeout check period value is 60.

One of the properties most often identified as requiring a change is the interrupt priority; this configuration setting is available in 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. 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, yet is easily visible within the ISDE when configuring interrupt-priority levels.

Note: You may want to open your ISDE and create the NetX Telnet Server and explore the property settings in parallel with looking over the following configuration table settings. This will help orient you and can be a useful hands-on approach to learning the ins and outs of developing with SSP.

Table 4. Configuration Settings for the NetX Telnet Server Module

ISDE Property	Value	Description
Internal thread priority	16	Internal thread priority selection
Maximum clients to serve simultaneously	4	Maximum number of clients that the Telnet Server can connect to serve simultaneously
Socket window size (bytes)	2048	TCP `Socket` receive window size (bytes) selection
Server time out (seconds)	10	Server time out (seconds) for connecting and disconnecting to `Client.selection`
Client inactivity timeout (seconds)	600	Client inactivity timeout (seconds) `selectionTimeout` for Client session. If this expires, the Server drops the connection.
Timeout check period (seconds)	600	Timeout check period (seconds) selection
Option negotiation	Enable, Disable (Default: Enable)	Option negotiation selection
Use application packet pool	Enable, Disable (Default: Disable)	Use Enable setting the Telnet Server application packet pool from the application (Option negotiation must be enabled) selection.
Packet size in the pool (bytes)	300	Packet size payload when the Telnet Server needs to create its own packet pool. Option negotiation must be enabled, and Use application packet pool must be disabled.in the pool (bytes) selection.
Total packet pool size (bytes)	2048	Packet size when the Telnet Server needs to create its own packet pool. Option negotiation must be enabled, and Use application packet pool must be disabled. Total packet pool size (bytes) selection.
Name	`g_telnet_server0`	Name selection
Internal thread stack size (bytes)	2048	Internal thread stack size (bytes) selection
Name of Client Connect Callback Function	`telnet_client_connect`	Name of Client Connect Callback Function selection
Name of Receive Data Callback Function	`telnet_receive_data`	Name of Receive Data Callback Function selection
Name of Client Disconnect Callback Function	`telnet_client_disconnect`	Name of Client Disconnect Callback Function selection

Note: The example settings and defaults are for a project using the S7G2 Synergy MCU. Other Synergy MCU Groups 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 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 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 Server Module Low-Level Modules

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

Table 5. Configuration Settings 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 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 indicates the 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	512	ARP Cache Size in Bytes selection
Reverse ARP	Enable, Disable (Default: Enable)	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

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

Table 6. Configuration Settings for the NetX Telnet Common

ISDE Property	Value	Description
Type of Service for UDP requests	Normal, Minimum delay, Maximum data, Maximum reliability, Minimum cost (Default: Normal)	Type of service UDP requests selection
Fragmentation option	Don't fragment, Fragment okay (Default: Don't fragment)	Fragment option selection
Server TCP port number	23	Server TCP port number selection
Time to live	128	Time to live selection

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

Table 7. Configuration Settings for the NetX Common on nx

ISDE Property	Value	Description
No configurable settings		

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

Table 8. Configuration Settings 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_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: Priority 5)	Ethernet interrupt priority selection
Name	`g_sf_el_nx`	Module name
Channel	0	Channel selection
Callback	NULL	Callback selection

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

5.2 NetX Telnet Server 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 Server 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 lists the method for selecting the pins within the SSP configuration window and the subsequent table lists an example selection for the I^2C pins.

Note: The operation mode selection 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: The selection sequence assumes USBFS0 or USBHS0 are the desired hardware target for the driver.

Table 10. Pin Configuration Settings for ETHERC1

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

Property	Value	Description
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 and defaults are for a project using the S7G2 Synergy MCU. Other Synergy MCU Groups may have different default values and available configuration settings.

6. Using the NetX Telnet Server Module in an Application

The typical steps using the NetX Telnet Server along with a test client in an application are:

1. Use the `nx_ip_status_check` API to check that the IP instance can communicate.
2. Configure the Telnet Server to do option negotiation, optional.
3. If options negotiation is enabled, configure the Telnet Server to create its own packet pool (default) or set the Telnet Server pool from the application.
4. Start the Telnet Server using the `nx_telnet_server_start` API.
5. Process client requests with the callbacks specified at build time.
6. Delete the Telnet Server when done.
7. If the Telnet Server packet pool was created by the application, it can be deleted if not in use by other threads.

The NetX Telnet Server module application needs to execute response processing in the registered callback function. Typically, in callback processing, the content of the received packet is confirmed, and the data content is transmitted with the `nx_telnet_server_packet_send` API. When disconnection of communication is required from the server side, call the `nx_telnet_server_disconnect` API.

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

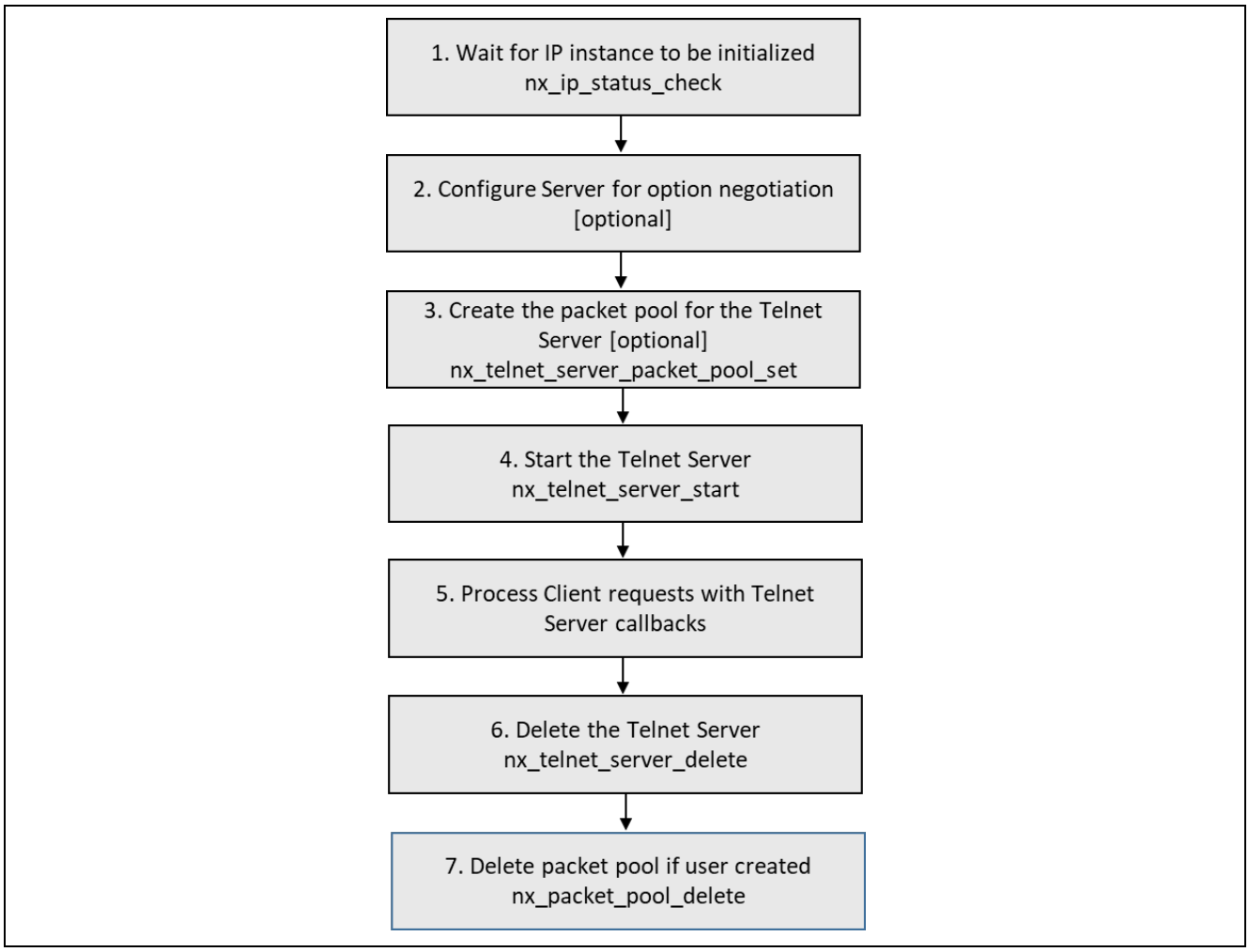

Figure 3. Flow Diagram for a Typical NetX Telnet Server Module Application

7. The NetX Telnet Server Module Application Project

The application project associated with this module guide demonstrates the steps in a full design. The project can be found using the link provided in the References section at the end of this document. You may want to import and open the application project within the ISDE and view the configuration settings for the NetX Telnet Server module. You can also read over the code (in `telnet_server_thread_entry.c`) which is used to illustrate the NetX Telnet Server module APIs in a complete design.

The application project demonstrates the typical use of the NetX Telnet Server module APIs. The application project, Telnet Server thread, creates the NetX IP instance as well as NetX Telnet Server instance; the Telnet Server is then started, and an infinite loop is initialized. Once a client connects, writes to the server, or disconnects from it, a callback function is executed.

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

Resource	Revision	Description
e^2 studio	6.2.1 or later	Integrated Solution Development Environment
SSP	1.5.0 or later	Synergy Software Platform
IAR EW for Synergy	8.23.1 or later	IAR Embedded Workbench® for Renesas Synergy™
SSC	6.2.1 or later	Synergy Standalone Configurator
SK-S7G2	v3.0 to v3.1	Starter Kit
Telnet Client	—	Can be a pc or another board running the module guide telnet_client

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

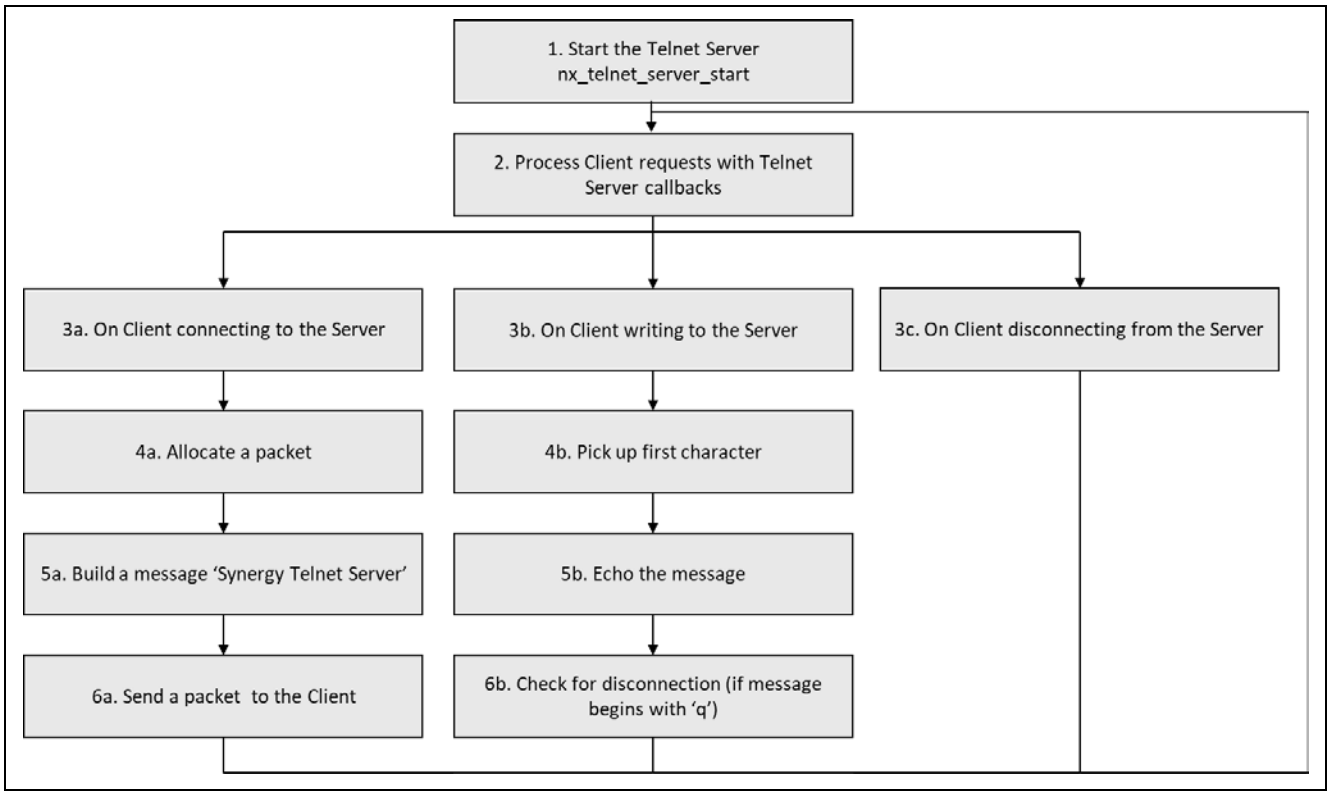

Figure 4. NetX Telnet Server Module Application Project Flow Diagram

The first section of `telnet_server_thread_entry.c` has the header file that references the telnet server instance and a code section that creates the server itself. The next section is the callback functions; the first function, `telnet_client_connect`, is run whenever a client connects to the server. Its purpose is to allocate a packet to a packet pool, create a greeting message **Synergy Telnet Server**, and send it to the client. The second function, `telnet_receive_data`, is run whenever the client sends any data to the server; it echoes back the message the server received. When a message begins with a letter **q**, the server disconnects. The last callback function, `telnet_client_disconnect`, runs when a client disconnects from the server; it doesn't perform any action. The next section contains the entry function which starts the Telnet Server, then it initializes the infinite loop.

A few key properties are configured in this application project to support the required operations and the physical properties of the target board and MCU. The properties with the values set for this specific project are listed in the following table. You can also open the application project and view these settings in the **Properties** window as a hands-on exercise.

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

ISDE Property	Value Set
Properties of g_ip0 NetX IP Instance	
IPv4 Address (use commas for separation)	Any static IPv4 address of your choice. (192,168,0,10)
Subnet Mask (use commas for separation)	Any subnet mask of your choice (255,255,255,0)
IPv6 Link Local Address (use commas for separation; All zeros indicates the user MAC address) – valid for NetX Duo only	Any static IPv6 address of your choice. (0xfe80, 0x0, 0x0, 0x0, 0x123, 0x4567, 0x89ab. 0xcdef)
Properties of g_sf_el_nx NetX Port ETHER on sf_el_nx	
Channel 1 Phy Reset Pin	IOPORT_PORT_08_PIN_06
Channel	1
Ethernet Interrupt Priority	Priority 8 (CM4 valid, CM0+: invalid)

Note: Also set a global address when running the client application on another board and make sure it matches that global address.

8. Customizing the NetX Telnet Server Module for a Target Application

Some configuration settings will normally be changed by you from those shown in the application project. For example, you can easily change the configuration settings for the NetX IP Instance. They can set a static IP of your choice or use a NetX DHCP Client to obtain an IP address provided by a DHCP server in the network. You can also change the body of callback functions to adjust their behavior to your own needs.

9. Running the NetX Telnet Server Module Application Project

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

Note: The following steps are described in sufficient detail for someone experienced with the basic flow through the Synergy development process. If these steps are unfamiliar, refer to the first few chapters of the *SSP User's Manual* to learn how to accomplish these steps.

To create and run the NetX Telnet Server module application project, simply follow these steps:

1. Create a new Renesas Synergy project for the S7G2-SK MCU called **NETX_TELNET_SERVER_MG_AP**.
2. Select the **Threads** tab.
3. Add a new thread to the threads, name it, **TELNET Server Thread**, and set the symbol `telnet_server_thread`.
4. In the **TELNET Server Thread Stacks**, add a **NetX Telnet Server**.
5. Click on the **Generate Project Content** button.
6. Add the code from the supplied project file `telnet_server_thread_entry.c` or copy over the generated `telnet_server_thread_entry.c` file.
7. Connect to the host PC via a micro USB cable to J19 on the SK-S7G2 MCU.
8. Connect the host PC via an Ethernet cable to J11 on the SK-S7G2 MCU.
9. Start to debug the application.
10. Open a terminal application (**PuTTY**, **Tera Term** or others) and connect to a **TELNET Server** using the chosen IP address. In case of a failure, make sure you have a statically defined IP address on your client machine and your firewall is disabled.
11. The result can be viewed in the terminal application window.

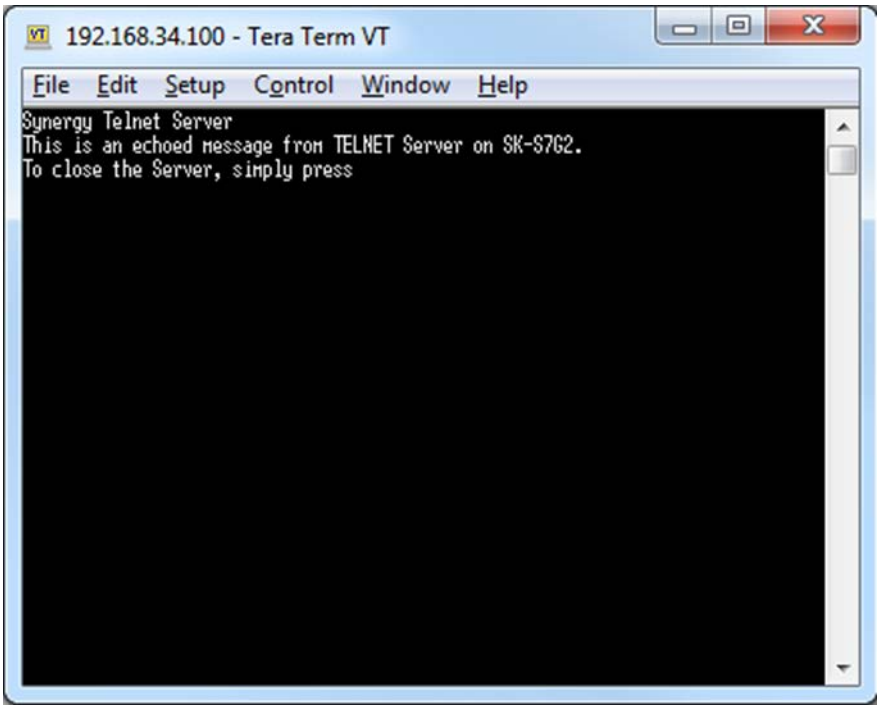

Figure 5. Example Output from NetX Telnet Server Application Project

10. NetX Telnet Server 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 less time-consuming and removes the common errors, such as conflicting configuration settings, or 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. You can setup a Telnet Server on your kit and provide callback functions for various activities of the Telnet Clients and connect to the server in just a few moments.

11. NetX Telnet Server Module Next Steps

After you have mastered a simple Telnet Server module project, you may want to review a more complex example. Other application projects and application notes that demonstrate NetX module use can be found as described in the following References section.

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

12. NetX Telnet Server Module Reference Information

SSP User Manual: Available in HTML format in the SSP distribution package and as a pdf from the Synergy Gallery.

Links to the latest NetX Telnet Server module reference materials and resources are available on the Renesas Knowledge Base: https://en-support.renesas.com/knowledgeBase/17913024.

Website and Support

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

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

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Jun.16.17	—	Initial Release
1.01	Jan.08.18	—	Edits for grammar and usage
1.02	Jan.30.19	12 & 13	Minor updates to Table 11 and Table 12

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/.