Renesas Synergy™ Platform

NetX™ HTTP Client 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 Hypertext Transfer Protocol (HTTP) is a protocol designed for transferring content on the web. HTTP is a simple protocol that utilizes reliable Transmission Control Protocol (TCP) services to perform its content-transfer function; HTTP is a highly reliable content-transfer protocol, as well as being one of the most frequently used application protocols. All operations on the web utilize the HTTP protocol. The NetX Duo™ HTTP Client accommodates both IPv4 and IPv6 networks while the NetX™ HTTP Client only supports IPv4 communications. IPv6 does not directly affect the HTTP protocol. Some differences with the NetX HTTP Client are necessary to accommodate IPv6, which are covered later in this document.

This overview covers elements related to the NetX HTTP Client module implementation on the Renesas Synergy™ Platform. The primary focus is adding and configuring the NetX HTTP module to a project. For more details on the operation of this module, consult the *NetX Hyper Text Transfer (HTTP) Client User's Guide* for the Renesas Synergy™ Platform and *NetX Duo Hyper Text Transfer (HTTP) Client User's Guide* for the Renesas Synergy Platform documents. They are part of *X-Ware™ Component Documents for Renesas Synergy™* zip file available from the Synergy Gallery (https://www.renesas.com/synergy/software).

Note: Both NetX and NetX Duo-based modules are covered. In almost all cases, NetX Duo is a superset of NetX, so any module capabilities that are not supported by NetX will be clearly identified with a note.

Contents

1. NetX HTTP Client Module Features

- High-level APIs to:
  - Create and delete an HTTP client instance
  - Send Get and Put requests to HTTP servers
- The NetX HTTP is compliant with RFC1945, Hypertext Transfer Protocol/1.0, RFC 2581, TCP Congestion Control, RFC 1122, Requirements for Internet Hosts and related RFCs.

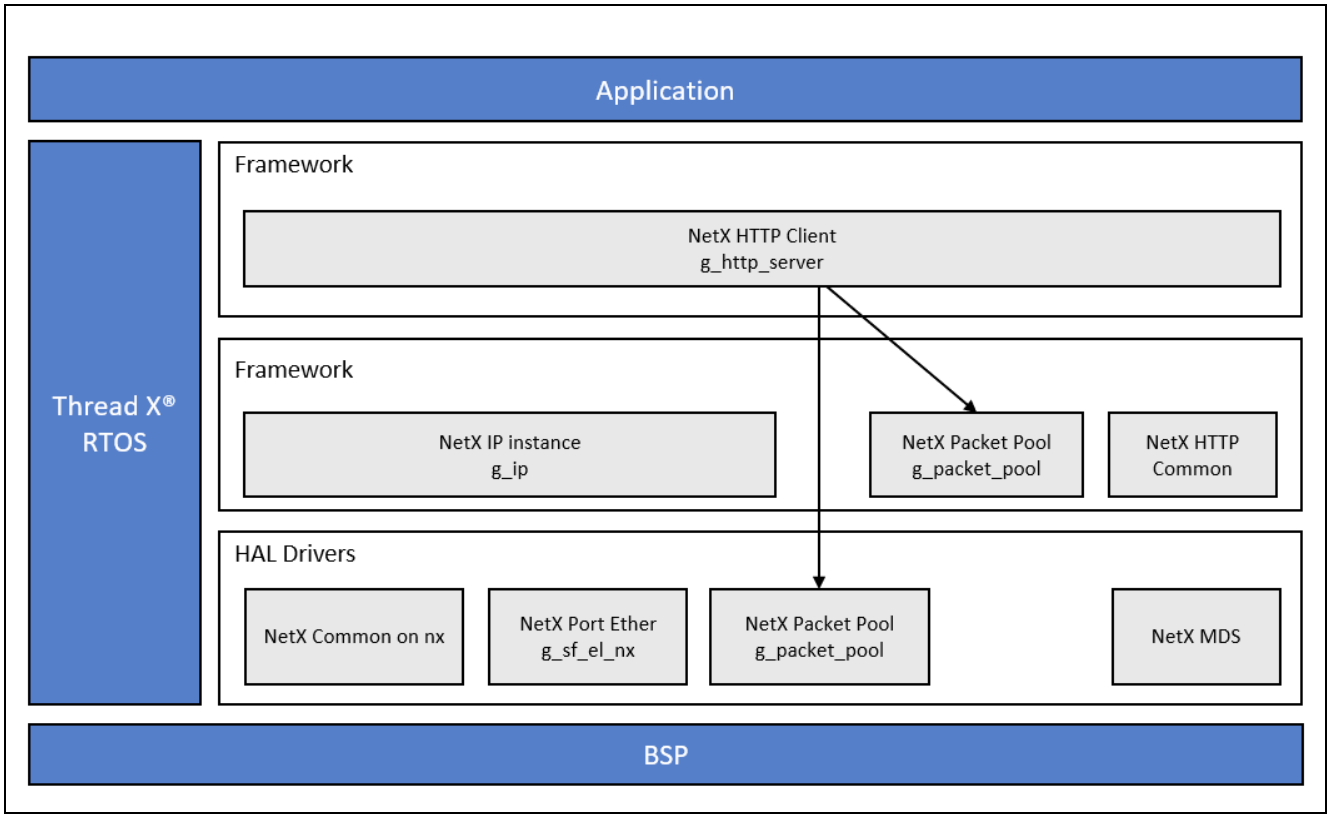


Figure 1. NetX HTTP Client Module Block Diagram

2. NetX HTTP Client Module APIs Overview

The NetX HTTP Client module defines APIs for creating, deleting, getting, and putting. 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 HTTP Client Module API Summary

Function Name	Example API Call and Description
nx_http_client_create	`nx_http_client_create(&my_client, “my client”, &ip_0, &pool_0, 100);` Create an HTTP Client Instance.
nx_http_client_delete	`nx_http_client_delete(&my_client);` Delete an HTTP Client instance.
nx_http_client_get_start	`nx_http_client_get_start(&my_client, IP_ADDRESS(1,2,3,5), “/TEST.HTM”, NX_NULL, 0, “myname”, “mypassword”, 1000);` Start an HTTP GET request (IPv4 only).
nxd_http_client_get_start**	`nxd_http_client_get_start(&my_client, &server_ip_address, “/TEST.HTM”, NX_NULL, 0, “myname”, “mypassword”, 1000);` Start an HTTP GET request (IPv4 or IPv6)

Function Name	Example API Call and Description
nx_http_client_get_packet	`nx_http_client_get_packet(&my_client, &next_packet, 1000);` Get next resource data packet.
nx_http_client_put_start	`nx_http_client_put_start(&my_client, IP_ADDRESS(1, 2, 3, 5), “/TEST.HTM”, “myname”, “mypassword”, 20, NX_WAIT_FOREVER);` Start an HTTP PUT request (IPv4 only).
nxd_http_client_put_start**	`nxd_http_client_put_start(&my_client, &server_ip_address, "/client_test.htm", "name", "password", 103, 50);` Start an HTTP PUT request (IPv4 or IPv6)
nx_http_client_put_packet	`nx_http_client_put_packet(&my_client, packet_ptr, NX_WAIT_FOREVER);` Send next resource data packet.
nx_http_client_set_connect_port**	`nx_http_client_set_connect_port(&g_http_client0, 81);` Connect to the HTTP server port on the specified port. Intended for situations where the Client must use another port beside 80.

Note: For details on operation and definitions for the function data structures, typedefs, defines, API data, API structures, and function variables, review the *SSP User's Manual* **API References** for the associated module.

**Only available in NetX Duo HTTP Client.

Table 2. Status Return Values

Name	Description
NX_SUCCESS	Successful HTTP function
NX_CALLER_ERROR**	Invalid caller of the service
NX_PTR_ERROR**	Invalid HTTP, `ip_ptr`, or packet pool pointer
NX_INVALID_PORT**	Invalid port input
NX_HTTP_POOL_ERROR	Invalid payload size in packet pool
NX_HTTP_NOT_READY	HTTP Client not in ready state
NX_HTTP_PASSWORD_TOO_LONG	Password exceeded expected length
NX_HTTP_AUTHENTICATION_ERROR	Invalid name and/or password
NX_HTTP_FAILED	HTTP client error communicating with the HTTP server
NX_HTTP_GET_DONE	HTTP client get packet operation is complete
NX_HTTP_BAD_PACKET_LENGTH	Invalid packet received - length incorrect
NX_HTTP_INCOMPLETE_PUT_ERROR	Server responds before PUT is complete
NX_HTTP_REQUEST_UNSUCCESSFUL_CODE	Received an error code instead of 2xx from server
NX_HTTP_PASSWORD_TOO_LONG	Password exceeded expected length
NX_HTTP_USERNAME_TOO_LONG	Username exceeded expected length
NX_SIZE_ERROR	Invalid total size of resource in PUT request
NX_INVALID_PACKET	Invalid TCP packet; not enough room for packet header

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 are error codes which are only returned if error checking is enabled. Refer to the *NetX User's Guide* for the Renesas Synergy Platform or *NetX Duo User's Guide* for the Renesas Synergy Platform for more details on error-checking services in NetX and NetX Duo, respectively.

3. NetX HTTP Client Module Operational Overview

The NetX HTTP Client module creates an IP instance which carries out NetX operations and enables TCP services in the NetX library. It creates the HTTP client instance and a TCP socket for sending and receiving HTTP messages to the server listening on port 80. The HTTP client requires a packet pool; the module can supply one by sharing the IP default packet pool (`g_packet_pool0`), or by creating a new one. The minimum packet payload is set by the **Minimum packet size** property of the HTTP client instance. This packet pool is used by the HTTP client only to transmit packets, so the packet-pool size and payload can be optimized based on the expected size and the number of HTTP client packets sent out.

The NetX Duo HTTP Client supports both IPv4 and IPv6 connections; if the HTTP client needs to use IPv6 to connect to a server. By default it is. However, is using the **NetX Duo Source** element instead of the prebuilt NetX Duo library, make sure the **NetX Duo IPv6 Support** property is set to **enabled** in the **NetX Duo Source** list of properties. By default, the ICMPv6 checksum computation for incoming and outgoing packets are performed by default. But if the project uses the NetX Duo Source element, verify that the **Checksum computation support on transmitted ICMPv6 packets** and **Checksum computation support on received ICMPv6 packets** properties of the **NetX Duo source** element set to **enabled**. Make sure the **IPv6 Global Address of the Client** host is set in the IP instance element; NetX Duo will do the necessary processing to enable IPv6 and ICMPv6 services required for IPv6 underlying protocols.

Once the HTTP client has a valid IP address, it can make PUT and GET requests. To upload packets, use the `nx_http_client_put_start**` service. This service has a server IP address input, so the HTTP client can connect to the server. If the data to upload exceeds more than one packet, the application uses the `nx_http_client_put_packet` service until all the data is uploaded. To download data from the server, use the `nx_http_client_get_start` service; this requires the server IP address, so the HTTP client can connect to the server. If the data to download exceeds more than one packet, the application uses the `nx_http_client_get_packet` service until all the data is downloaded; this is indicated by getting the `NX_HTTP_GET_DONE` status return.

In the NetX Duo HTTP Client module, the application can use the `nxd_http_client_put_start` for IPv4 connections, and the `nxd_http_client_get_start` service for either IPv4 or IPv6 connections; `nx_http_client_put_start` and `nx_http_client_get_start` services are also available in the Net Duo HTTP Client. The `nx_http_client_put_packet` and `nx_http_client_get_packet` services do not require an HTTP Server IP address, so there is no Duo-equivalent APIs for these services.

In NetX Duo HTTP Client, the `nx_http_client_set_connect_port` service is available for those circumstances where the HTTP client needs to connect to the HTTP server on a port other than the default of Port 80.

HTTP Server Responses

Once the HTTP server processes the client command, it returns an ASCII response string that includes a 3-digit numeric-status code listed in the following table. The numeric response is used by the HTTP client software to determine whether the operation succeeded or failed.

Table 3. Various HTTP server responses to client commands

Numeric Field	Meaning
200	Request was successful
400	Request was not formed properly
401	Unauthorized request, client needs to send authentication
404	Specified resource in request was not found
500	Internal HTTP server error
501	Request not implemented by HTTP server
502	Service is not available

For example, a successful client request to PUT the file **test.htm** is responded to with the message **HTTP/1.0 200 OK**.

3.1 NetX HTTP Client Module Important Operational Notes and Limitations

3.1.1 NetX HTTP Client Module Operational Notes

- The HTTP client packet pool must be large enough to hold the complete HTTP header.
- The wait option for disconnecting from the server before deleting the client TCP socket is set by the Operation Timeout property; the wait option for all other HTTP client services is set in the API.
- Both GET and PUT start services require a resource, username, and password as an input. The maximum size of each is set by the **Maximum resource name length**, **Maximum username length**, and **Maximum password length** properties in the **NetX HTTP Common** element. When the GET and PUT operation is completed, the HTTP client disconnects from the server.
- The HTTP client TCP socket receive window is set by the **TCP socket window size** property. This is used in the TCP protocol for one peer to let the other know not to send more data pending-acknowledgment packets for data already received.

3.1.2 NetX HTTP Client Module Limitations

- The HTTP protocol in NetX and NetX Duo implements the HTTP 1.0 standard; it does not support 1.1. The constraints are as follows:
  - Persistent connections are not supported
  - Request pipelining is not supported
  - Content compression is not supported
  - TRACE, OPTIONS, and CONNECT requests are not supported
- Refer to the most recent *SSP Release Notes* for any additional operational limitations for this module.

4. Including the NetX HTTP Client Module in an Application

This section describes how to include the NetX HTTP Client module in an application using the SSP configurator.

Note: It is assumed that 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 or NetX Duo HTTP Client module to an application, simply add it to a thread using the stacks selection sequence given in the following table. (The default name for the NetX and NetX Duo HTTP Client module is `g_http_client0`. This name can be changed in the associated **Properties** window.)

Table 4. NetX HTTP Client Module Selection Sequence

Resource	ISDE Tab	Stacks Selection Sequence
`g_http_client0` NetX HTTP Client Module	Threads	New Stack> X-Ware> NetX> Protocols> NetX HTTP Client
`g_http_client0` NetX Duo HTTP Client Module	Threads	New Stack> X-Ware> NetX Duo> Protocols> NetX Duo HTTP Client

When the NetX HTTP Client 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 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 as 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.

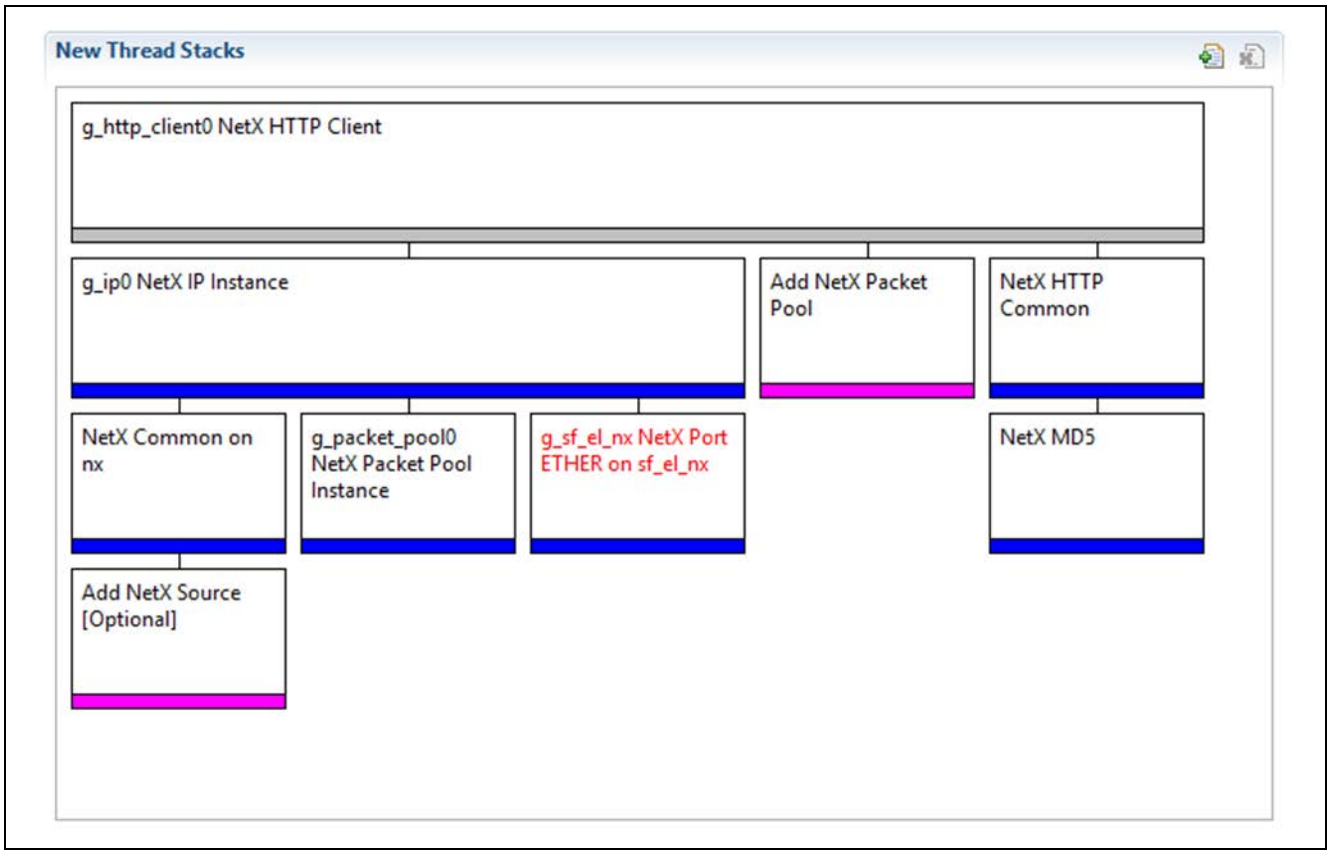


Figure 2. NetX HTTP Client Module Stack

5. Configuring the NetX HTTP Client Module

The NetX HTTP Client 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, that 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.

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. Note that the interrupt priorities listed in the **Properties** window in the ISDE includes 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 HTTP Client module 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 5. Configuration Settings for the NetX HTTP Client Module

Parameter	Value	Description
Minimum packet size (bytes)	300	Minimum packet size selection
Operation timeout (seconds)	10	Operation timeout selection
Maximum password length (bytes) (**Not included in NetX Duo)	20	Maximum password length selection

Parameter	Value	Description
Maximum username length (bytes) (**Not included in NetX Duo)	20	Maximum username length selection
Name	g_http_client0	Module name
TCP socket window size (bytes)	1024	TCP socket window size selection

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

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

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

5.1 Configuration Settings for the NetX SNTP Client Lower-Level Modules

Typically, 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 6. 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 Global Address (use commas for separation)	0x2001, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x1	IPv6 global address selection
**IPv6 Link Local Address (use commas for separation, all zeros mean use MAC address)	0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0	IPv6 link local address selection
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: 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

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

**Indicates properties only included with NetX Duo.

Table 7. Configuration Settings for the IP Default Packet Pool Instance

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

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

Table 8. Configuration Settings for the optional HTTP Client Packet Pool Instance

ISDE Property	Value	Description
Name	g_packet_pool1	Module name
Packet Size in Bytes	640	Packet size selection
Number of Packets in Pool	16	Number of packets in pool selection

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

Table 7. Configuration Settings for the NetX Common Instance

ISDE Property	Value	Description
No configurable settings		

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

Table 8 Configuration Settings for the NetX HTTP Common Instance

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
Time to live	128	Time to live selection
MD5 Support	Enable, Disable Default: Disable	MD5 support selection
Packet Queue depth	40	Packet queue depth selection

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

Table 9. 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

ISDE Property	Value	Description
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), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX®), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid) Default: Disabled	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 Group. Other MCUs may have different default values and available configuration settings.

Table 10. Configuration Settings for the NetX MD5

ISDE Property	Value	Description
No configurable settings		

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

5.2 NetX HTTP 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 HTTP 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 illustrates the method for selecting the pins within the SSP configuration window and the subsequent table illustrates 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 11. Pin Selection for the ETHERC Module

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

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

Table 12. Pin Configuration Settings for the 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
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 Group. Other MCUs may have different default values and available configuration settings.

6. Using the NetX HTTP Client Module in an Application

The following example assumes an IP Instance for the HTTP server with ARP has already been created and TCP has been enabled. The typical steps in using the NetX HTTP Client module (and the associated server functions) in an application are:

1. Wait for valid IP address and network driver initialization using the nx_ip_status_check API.
2. Upload data to the HTTP server with the `nx_http_client_put_start` API. For IPv6 connections, use the `nxd_http_client_put_start` API in the NetX Duo HTTP Client (which can also be used for IPv4 connections).
3. Download data from the HTTP server with the `nx_http_client_get_start` API. For IPv6 connections, use the `nxd_http_client_get_start` API in the NetX Duo HTTP Client (which can also be used for IPv4 connections).
4. Delete the HTTP client with the `nx_http_client_delete` API. (Note that the packet pool can also be deleted if it's not used elsewhere, such as by the IP instance in the application.)

The following diagram illustrates the common steps in a typical operational flow:

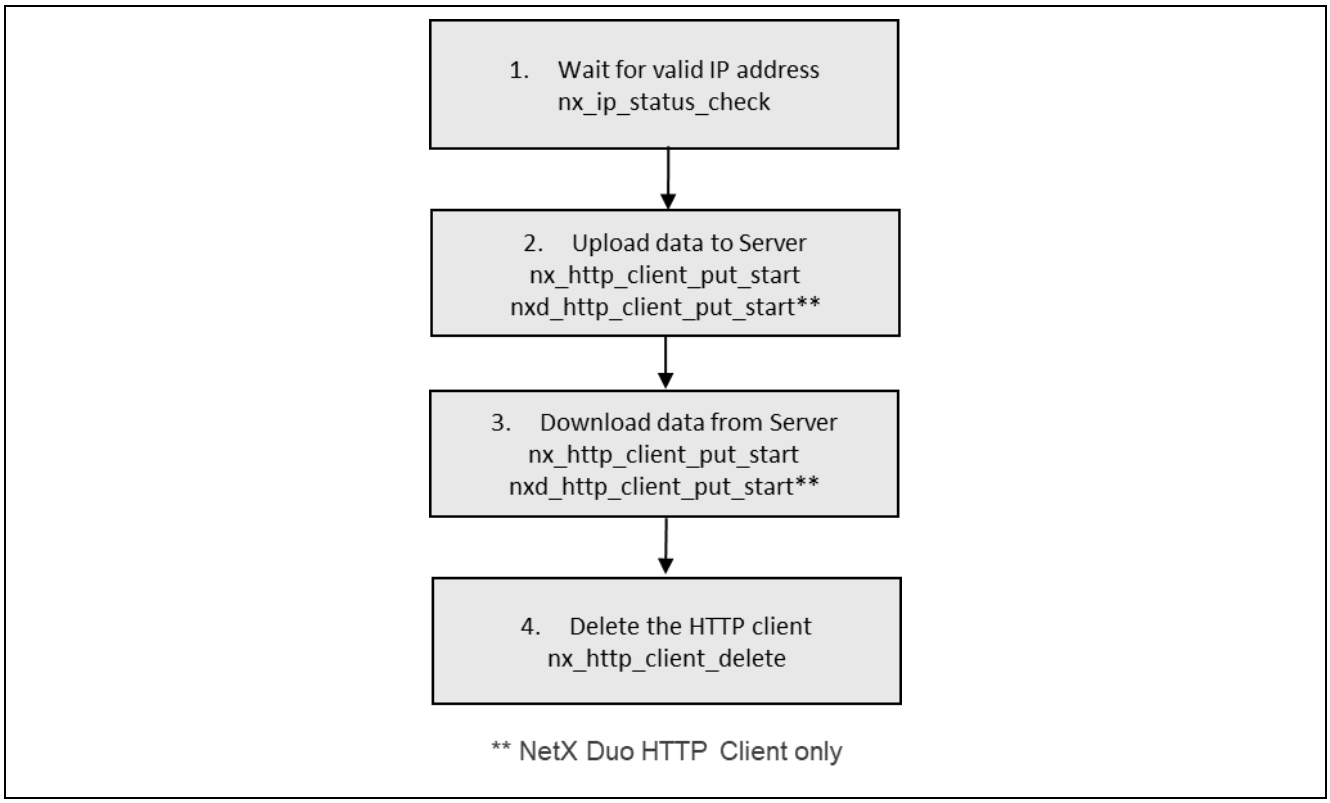


Figure 3. Flow Diagram of a Typical NetX HTTP Client Module Application

7. The NetX HTTP Client 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 HTTP Client module. You can also read over the code (`http_thread_entry.c`) used to illustrate the NetX HTTP Client module APIs in a complete design.

The application project demonstrates the typical use of the NetX HTTP Client module APIs. It connects to a HTTP server running on its network and gets a remote resource from that server specified by macro constants. The contents of the resource are printed to the debugger console via semi-hosting.

Table 13. Software and Hardware Resources Used by the Application Project

Resource	Revision	Description
e² studio	7.3.0 or later	Integrated Solution Development Environment
SSP	1.6.0 or later	Synergy Software Platform
IAR EW for Synergy	8.23.3 or later	IAR Embedded Workbench® for Renesas Synergy™
SSC	7.3.0 or later	Synergy Standalone Configurator
SK-S7G2	v3.0 to v3.1	Starter Kit

The following diagram shows the application project in a simple flow:

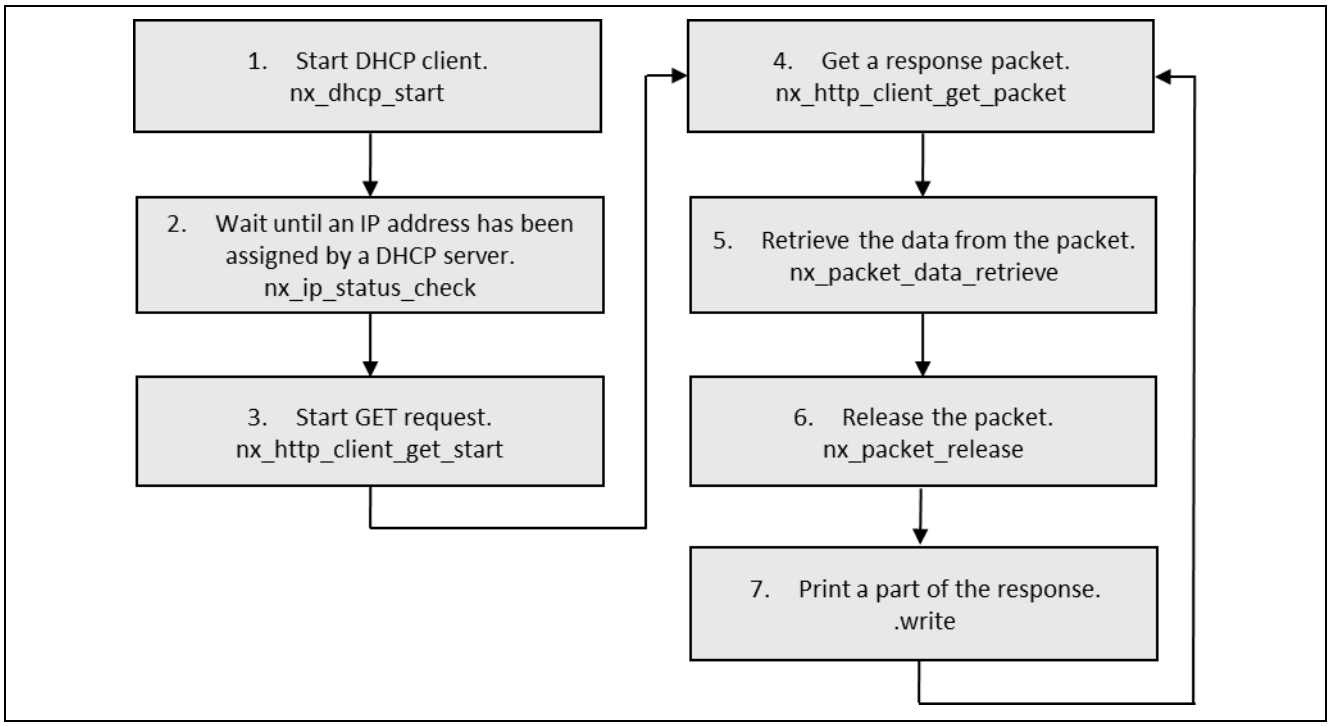


Figure 4. NetX HTTP Client Module Application Project Flow Diagram

The `http_thread_entry.c` file resides in the project once it has been imported into the ISDE. You can open this file, within the ISDE and follow along with the description provided to help identify key uses of APIs.

The first section of `http_thread_entry.c` has the header files which reference the NetX HTTP Client instance structure and a code section which defines macro constants used in the application. There are settings for a host IP address, a resource path, a wait option used in HTTP client operations, and a receive buffer. The next section is the entry function for the main program-control section. The HTTP client instance initializes a GET request using the HTTP Server IP address and the resource `HTTP_CLIENT_PATH`. The GET loop then starts, the response packet is received, and the data is extracted and copied to a receive buffer. Once done, the packet is released and the contents printed to the virtual debug console. The application repeats these functions until all the data is received.

Table 14. NetX HTTP Client Module Configuration Settings for the Application Project

ISDE Property	Value Set
Minimum packet size (bytes)	300
Operation timeout (seconds)	10
Maximum password length (bytes)	20
Maximum username length (bytes)	20
Name	g_http_client0
TCP socket window size (bytes)	1024

Table 15. Configuration Settings for the NetX IP Instance

ISDE Property	Value Set
Name	`g_ip0`
IPv4 Address (use commas for separation)	0,0,0,0
Subnet Mask (use commas for separation)	255,255,255,0
IP Helper Thread Stack Size (bytes)	2048
IP Helper Thread Priority	3
ARP	Enable
ARP Cache Size in Bytes	520
Reverse ARP	Disable

ISDE Property	Value Set
TCP	Enable
UDP	Enable
ICMP	Enable
IGMP	Enable
IP fragmentation	Disable

Table 16. Configuration Settings for the NetX HTTP Common Module

ISDE Property	Value Set
Type of Service	Normal
Fragmentation option	Don't fragment
Time to live	128
MD5 Support	Disable
Maximum resource name length (bytes)	40

Table 17. Configuration Settings for the NetX Packet Pool Instance

ISDE Property	Value Set
Name	`g_packet_pool0`
Packet Size in Bytes	1568 Recommended size to avoid packet chaining. This packet pool is shared between IP instance and HTTP Client. So, it will be used to receive packets as well.
Number of Packets in Pool	16

Table 18. Configuration Settings for the NetX Port ETHER

ISDE Property	Value Set
Parameter Checking	Default (BSP)
Channel 0 Phy Reset Pin	`IOPORT_PORT_09_PIN_03`
Channel 0 MAC Address High Bits	0x00002E09
Channel 0 MAC Address Low Bits	0x0A0076C7
Channel 1 Phy Reset Pin	`IOPORT_PORT_08_PIN_06`
Channel 1 MAC Address High Bits	0x00002E09
Channel 1 MAC Address Low Bits	0x0A0076C8
Number of Receive Buffer Descriptors	8
Number of Transmit Buffer Descriptors	32
Ethernet Interrupt Priority	Priority 2
Name	`g_sf_el_nx`
Channel	1
Callback	NULL

8. Customizing the NetX HTTP Client Module for a Target Application

Some configuration settings will normally be changed by the developer from those shown in the application project. For example, you can easily change the host IP address and the path to the resource in `http_thread_entry.c` file using macro definitions.

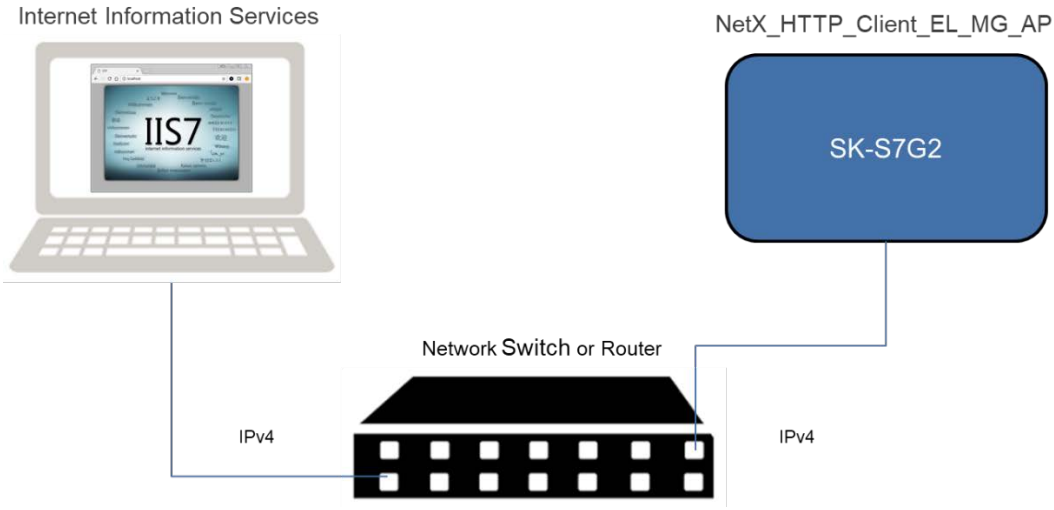
This application requires that you set your IP address statically. If you want to get an assigned IP address from a DHCP server on the network, you can include the DHCP Client in the main thread. The *NetX™ DHCP Client Module Guide* illustrates how to use it to obtain a dynamic provided IP address from a DHCP Server.

9. Running the NetX HTTP Client Module Application Project

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

Note: The following steps are detailed sufficiently 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* for a description of how to accomplish these steps.

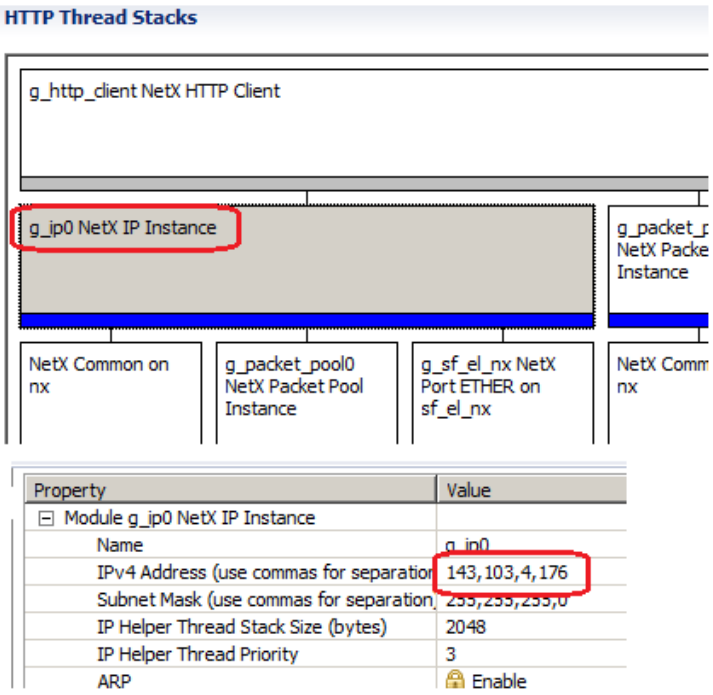
1. Connect your board and a test PC to a network switch or router.



   Run a HTTP server program on the test PC. In this example IIS (internet information services) was used because it is included on all versions of Windows starting with Windows 7. Be sure to configure the firewall on your Windows PC to allow outbound traffic over port 80.
2. Refer to the *Renesas Synergy™ Project Import Guide* (r11an0023eu0121-synergy-ssp-import-guide) included in this package) for instructions on importing the project into e^2 studio or IAR EW for Synergy and building/running the application.
3. Edit `http_thread_entry.c` by setting the constant HTTP_CLIENT_HOST_IP_ADDRESS to the IP address of the test PC running IIS (**Note: Do not include spaces between the numbers in the IP Address**).

```
38
39      /* IP address of the HTTP server */
40      #define HTTP_CLIENT_HOST_IP_ADDRESS IP_ADDRESS(143, 103, 4, 172)
41
42      /* Path to resource */
```

4. Set the property of IPv4 Address to an IP address for the board that is on the same network as the test PC.



5. Build the project and run the application.
6. The output can be viewed in the Renesas Debug Virtual Console.

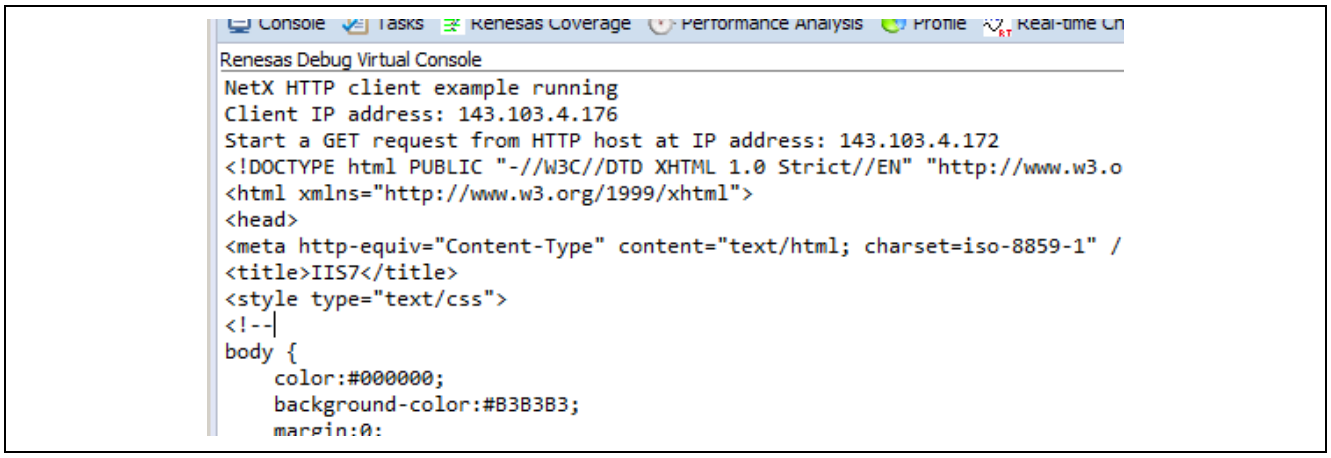


Figure 5. Example Output from NetX HTTP Client Module Application Project

10. NetX HTTP 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 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.

11. NetX HTTP Client Module Next Steps

After you have mastered a simple NetX HTTP Client project, you may want to review a more complex example. You may find that the HTTP Client with the DNS Client is a better fit for your target application. The DNS Client allows you to resolve IP addresses from host names. If you would like to run an HTTP server, then the NetX HTTP Server might be used. Guides for these modules can be found using instructions described in the following References section.

12. NetX HTTP Client Module Reference Information

SSP User Manual: Available in HTML format in the SSP distribution package and as a pdf from the Synergy Gallery (www.renesas.com/synergy/software).

Links to all the most up-to-date NetX or NetX Duo HTTP Client module reference materials and resources are available on the Synergy Knowledge Base:

- For NetX HTTP Client Module Guide Resources, visit: https://en-support.renesas.com/knowledgeBase/16977459
- For NetX Duo HTTP Client Module Guide Resources, visit: https://en-support.renesas.com/knowledgeBase/17913006

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.22.17	—	Initial Release
1.01	Jan.04.18	—	Minor edits for grammar and usage
1.02	Jan.10.19	—	Minor updates to configuration tables and settings
1.03	May.01.19	—	Updated for 1.6.0

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/.