Renesas Synergy™ Platform

NetX™ and NetX Duo™ FTP Client Module Guide

Introduction

This Module Guide will enable you to effectively use this module in your own design. On completion of this guide you will be able to 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 the document and will be valuable resources for creating more complex designs.

The File Transfer Protocol (FTP) is a protocol designed for file transfers. FTP utilizes reliable Transmission Control Protocol (TCP) services to perform its file transfer function. Because of this, FTP is a highly reliable file transfer protocol with high-performance. The actual FTP file transfer is performed on a dedicated FTP connection.

This module guide refers to the NetX Duo FTP Client. Except where noted, the information applies equally for the NetX FTP Client.

NetX Duo™ FTP Client accommodates both IPv4 and IPv6 networks whereas NetX™ FTP Client supports IPv4 only. While IPv6 does not directly change the FTP protocol, some changes in the original NetX FTP API are necessary to accommodate IPv6 and are described in this document.

Key elements related to the NetX Duo FTP Client implementation on the Renesas Synergy™ Platform are reviewed, especially adding and configuring the NetX Duo FTP Client module to a Renesas Synergy™ Platform project. For details on module operation, consult, *NetX Duo File Transfer Protocol (FTP) Client User's Guide* for the Renesas Synergy Platform. It is one of the X-Ware™ and NetX Component Documents for Renesas Synergy that is included in a zip file which is available from the Renesas Synergy Gallery (www.renesas.com/synergy/ssp).

The module guide's sections can be read independently by experienced Synergy developers or in series by developers new to the Renesas Synergy Platform.

Contents

1. NetX Duo FTP Client Features

- Multi-thread support
- APIs for connecting to and disconnecting from the FTP server
- APIs file read, write, rename and delete operations
- APIs for directory listing and default directory set operations
- Support for passive FTP Client mode

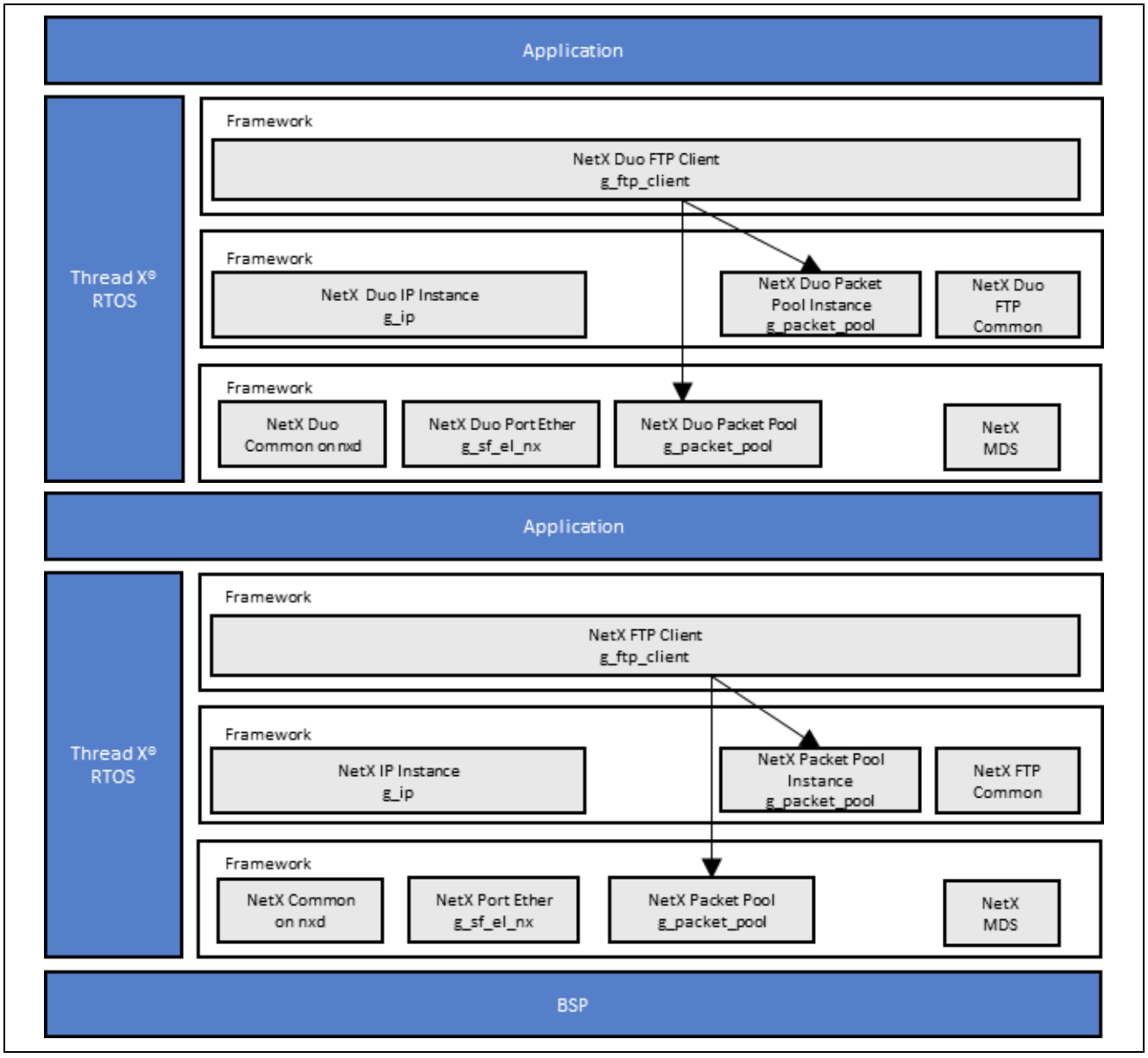


Figure 1. NetX and NetX Duo FTP Client Organization, Options and Stack Implementations

2. NetX Duo FTP Client APIs Overview

The NetX Duo FTP Client define APIs for client creation or deletion. It also defines APIs to connect to or disconnect from the server, directory operations, and file operations. The next table lists the available APIs, an example API call, and a short description of each API. A table of return status values follows.

Table 1. NetX Duo FTP Client API Summary

Function Name	Example API Call and Description
.nx_ftp_client_create	`nx_ftp_client_create(&my_client, “My Client”, &client_ip, 2000, &client_pool);` Delete an FTP Client instance.
.nx_ftp_client_delete	`nx_ftp_client_delete(&my_client);` Delete an FTP Client instance.
.nx_ftp_client_connect	`nx_ftp_client_connect(&my_client, server_ip_addr, NULL, NULL, 100);` Connect to an FTP Server IPv4 support.
.nxd_ftp_client_connect*	`nxd_ftp_client_connect(&my_client, server_ip_addr, NULL, NULL, 100);` Connect to an FTP Server with IPv6 and IPv4 support.
.nx_ftp_client_disconnect	`nx_ftp_client_disconnect(&my_client, 200);` Disconnect from the FTP Server.
.nx_ftp_client_create	`nx_ftp_client_directory_create(&my_client, “my_dir”, 200);` Create a directory on the server.
.nx_ftp_client_directory_set	`nx_ftp_client_directory_listing_set(&my_client, “my_dir”, &my_packet, 200);` Get a directory listing from the server.
.nx_ftp_client_directory_delete	`nx_ftp_client_directory_delete(&my_client, “my_dir”, 200);` Delete a directory on the server.
.nx_ftp_client_directory_listing_get	`nx_ftp_client_directory_listing_get(&my_client, “my_dir”, &my_packet, 200);` Get a directory listing from the server.
.nx_ftp_client_directory_listing_continue	`nx_ftp_client_directory_listing_continue(&my_client, &my_packet, 200);` Continue a directory listing from the server.
.nx_ftp_client_file_open	`nx_ftp_client_file_open(&my_client, “my_file.txt”, NX_FTP_OPEN_FOR_READ, 200);` Open a file on the server.
.nx_ftp_client_file_close	`nx_ftp_client_file_close(&my_client, 200);` Close the currently open file on the server.
.nx_ftp_client_file_read	`nx_ftp_client_file_read(&my_client, &my_packet, 200);` Read from a file already open on the server.
.nx_ftp_client_file_write	`nx_ftp_client_file_write(&my_client, my_packet, 200);` Write to an already open file on the server.
.nx_ftp_client_file_rename	`nx_ftp_client_file_rename(&my_client, “my_file.txt”, “new_file.txt”, 200);` Rename a file on the server.
.nx_ftp_client_file_delete	`nx_ftp_client_file_delete(&my_client, “my_file.txt”, 200);` Delete a file on the server.

Note: The nxd_ftp_client_connect function is only available in the NetX Duo FTP Client. For details on function data structures, typedefs, defines, API data, API structures and function variables, review the *SSP User’s Manual* listed in References at the end of this document.

Table 2. Error Status Return Values

Name	Description
NX_SUCCESS	Successful FTP function.
NX_FTP_NOT_DISCONNECTED	FTP client is already connected.
NX_FTP_200_CODE_NOT_RECEIVED	Server rejects FTP connection.
NX_FTP_300_CODE_NOT_RECEIVED	Server rejects user/password.
NX_PTR_ERROR	Invalid FTP, username, or password pointer.
NX_CALLER_ERROR	Invalid caller of this service.
NX_IP_ADDRESS_ERROR	Invalid IP address.
NX_FTP_NO_2XX_RESPONSE_XXD	FTP server error response.
NX_FTP_NO_2XX_RESPONSE_PORT	FTP server response to PORT.
NX_FTP_NO_1XX_RESPONSE	FTP server response to NLST.
NX_FTP_END_OF_LISTING	No more entries in directory.
NX_FTP_NO_2XX_RESPONSE_DISCONNECT	FTP server response to disconnect.

Note: Lower level drivers may return Common Error Codes. Refer to the *SSP User's Manual* listed in the References at the end of this document for a definition of all relevant Error codes.

3. NetX Duo FTP Client Operational Overview

FTP Client Requirements

To function properly, the NetX Duo FTP Client package requires NetX Duo. The NetX FTP Client package requires NetX. You cannot mix and match NetX/Duo libraries with FTP Client. The IP instance, packet pool(s) and FTP instance are generated automatically. All the application has to do is wait for the network link to become enabled (using the nx_ip_status_check service).

By default, IPv6 is enabled on NetX Duo. In the auto-generated code, IPv6 and IMCPv6 is enabled on NetX Duo, and the IPv6 global address and link local address are registered with the IP instance. By the time the thread entry function starts running, the FTP Client application can start using NetX Duo services. However, most IPv6 applications observe the Duplicate Address Detection protocol, where NetX Duo sends out three Neighbor Solicitation packets to verify that no other hosts on the network are using the same global or link local addresses. This takes usually 3-4 seconds. At this time, the FTP Client host is ready to start communicating over IPv6.

The FTP Client can use the FileX® embedded file system but it is not required. The application project simply uses local memory buffers to create file buffers for the FTP Client to upload to the FTP server. Further, a developer uses their own file system, along the guidelines suggested in `filex_stub.h`, to define each of the services listed in that file.

FTP File Names

FTP file names should be in the format of the target file system, such as FileX, if the application chooses to use one. FTP file names should be NULL-terminated ASCII strings, with full path information if necessary. There is no specified limit for the size of FTP file names in the NetX Duo FTP implementation. The packet pool payload size should be able to accommodate the maximum path and file name.

FTP Client Commands

The FTP has a simple mechanism for opening connections and performing file and directory operations. Basically, there is a set of standard FTP commands issued by the Client after a connection has been successfully established on TCP port 21. The following shows some of the basic FTP commands. Note that when FTP runs over IPv6 the only difference is the PORT command is replaced with the EPRT command, but this detail is handled internally by the FTP Client thread task:

Table 3. FTP Client Commands

FTP Command	Meaning
CWD path	Change working directory
DELE filename	Delete specified file name
EPRT ip_address, port	Provide IPv6 address and Client data port
LIST directory	Get directory listing
MKD directory	Make new directory
NLST directory	Get directory listing
NOOP	No operation, returns success
PASS password	Provide password for login
PORT ip_address,port	Provide IP address and Client data port
PWD path	Pickup current directory path
QUIT	Terminate Client connection
RETR filename	Read specified file
RMD directory	Delete specified directory
RNFR oldfilename	Specify file to rename
RNTO newfilename	Rename file to supplied file name
STOR filename	Write specified file
TYPE I	Select binary file image
USER username	Provide username for login

Note: These ASCII commands are used internally by the NetX Duo FTP Client software to perform FTP operations with the FTP Server.

FTP Server Responses

Once the FTP Server processes the Client request, it returns a 3-digit coded response in ASCII followed by optional ASCII text. The numeric response is used by the FTP Client software to determine whether the operation succeeded or failed. The following list shows various FTP Server responses to Client requests:

Table 4. First Numeric Field

First Numeric Field	Meaning
1xx	Positive preliminary status – another reply coming.
2xx	Positive completion status.
3xx	Positive preliminary status – another command must be sent.
4xx	Temporary error condition.
5xx	Error condition.

Table 5. Second Numeric Field

Second Numeric Field	Meaning
0xx	Syntax error in command.
1xx	Positive preliminary status – another reply coming.
2xx	Positive completion status.
3xx	Positive preliminary status – another command must be sent.
4xx	Temporary error condition.
5xx	Error condition.

FTP Write Requests:

1. Client issues TCP connect to Server port 21.
2. Server sends **220** response to signal **success**.
3. Client sends **USER** message with **username**.
4. Server sends **331** response to signal **success**.
5. Client sends **PASS** message with **password**.
6. Server sends **230** response to signal **success**.

7. Client sends **TYPE I** message for **binary transfer**.
8. Server sends **200** response to signal **success**.
9. IPv6 applications: Client sends **EPRT** message with IP address and port. IPv4 applications: Client sends **PORT** message with IP address and port.
10. Server sends **200** response to signal **success**.
11. Client sends **STOR** message with **file name to write**.
12. Server creates data socket and connects with client data port specified in the previous **EPRT** or **PORT** command.
13. Server sends **125** response to signal **file write has started**.
14. Client sends contents of file through the data connection. This process continues until file is completely transferred.
15. When finished, Client disconnects data connection.
16. Server sends **250** response to signal file write is successful.
17. Clients sends **QUIT** to terminate FTP connection.
18. Server sends **221** response to signal **disconnect is successful**.
19. Server disconnects FTP connection.

FTP Authentication

Whenever the file transfer protocol connection takes place, the Client must provide the Server with a username and password. Some FTP sites allow what is called Anonymous FTP, which allows FTP access without a specific username and password. For this type of connection, **anonymous** should be supplied for username and the password should be a complete email address. When the NetX Duo FTP Client connects to the FTP Server using the nxd_ftp_client_connect (nx_ftp_client_connect for NetX FTP Client), the username and password are two of the input arguments. These can be left blank or NX_NULL for the anonymous login.

Note that the difference between nx_ftp_client_connect and nxd_ftp_client_connect is the former takes a ULONG for the server IP address, while the latter takes an address type NXD_ADDRESS. This data type can hold either an IPv4 or IPv6 address. That fact that it supports both makes it a "**duo"** service as designated by the prefix "nx**d**_". NetX Duo FTP Client supports nx_ftp_client_connect, but you are encouraged to use **duo** services over legacy services whenever possible.

3.1 Important Operational Notes and Limitations

3.1.1 Operational Notes

FTP Multi-Thread Support

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

RFCs

NetX Duo FTP is compliant with RFC 959, RFC 2428, and related RFCs.

3.1.2 Limitations

NetX Duo FTP Client Constraints

The FTP standard has many options for file data representation. NetX Duo FTP does not implement switch options like ls – al. The NetX Duo FTP Server expect to receive requests and their arguments in a single packet rather than consecutive packets.

Like UNIX implementations, NetX Duo FTP assumes the following file format constraints:

- File Type: Binary
- File Format: Nonprint Only
- File Structure: File Structure Only

4. Including the NetX Duo FTP Client in an Application

This section describes how to include the NetX Duo FTP Client 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 *Getting Started Guide for SSP* listed in the Reference Section at the end of this document to learn how to manage each of these important steps in creating SSP based applications.

To add the NetX Duo FTP Client to an application, simply add it to the `ftp_client_thread` using the Stacks Selection Sequence in the following table.

Table 6. Thread Selection Sequence

Resource	ISDE Tab	Stacks Selection Sequence
FTP Client Thread	Threads	Threads> New Thread
Symbol	ftp_client_thread	
Name	FTP Client	
Stack size	2048 (default is 1024 which is marginal)	
Priority	10 (default is 1 but only the IP thread task should be priority 1	
Auto Start	Enabled	
Time Slicing	1	

The default name for the NetX Duo FTP Client is `g_ftp_client0` and is shown in the following tables. This name can be changed in the associated Properties window. When the driver is added, it shows up in the thread, as shown in Figure 2.

Table 7. NetX Duo FTP Client Selection Sequence

Resource	ISDE Tab	Stacks Selection Sequence
g_ftp_client0 NetX Duo FTP Client	Threads	New Stack> X-Ware> NetX Duo> Protocols> NetX Duo FTP Client

Table 8. NetX FTP Client Selection Sequence

Resource	ISDE Tab	Stacks Selection Sequence
g_ftp_client0 NetX FTP Client	Threads	New Stack> X-Ware> NetX> Protocols> NetX FTP Client

When the NetX Duo FTP Client is added to the Thread Stack, as shown in Figure 2, the configurator automatically adds the lower level drivers needed. Any drivers requiring additional configuration information are box text highlighted in red. Modules with a gray band are individual, standalone modules. Modules with a blue band are shared or common and only need to be added once, since they can be used by multiple stacks. Modules with a pink band can require selecting lower level drivers. Sometimes these drivers are optional or recommended and the block indicates this with the inclusion of this text. If lower level drivers are required, the module description will include “Add” in the text. Clicking any pink banded modules brings up the “New” icon and shows possible choices.

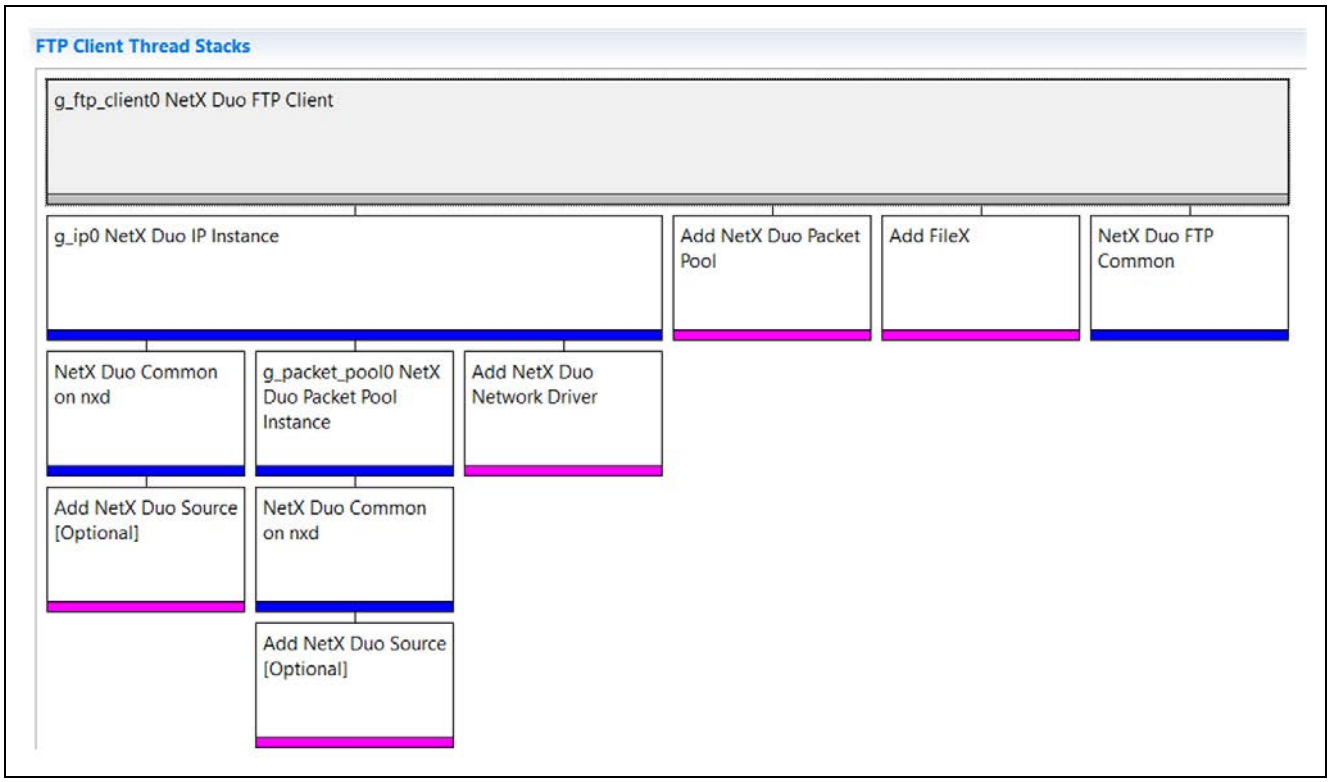


Figure 2. NetX Duo FTP Client Stack

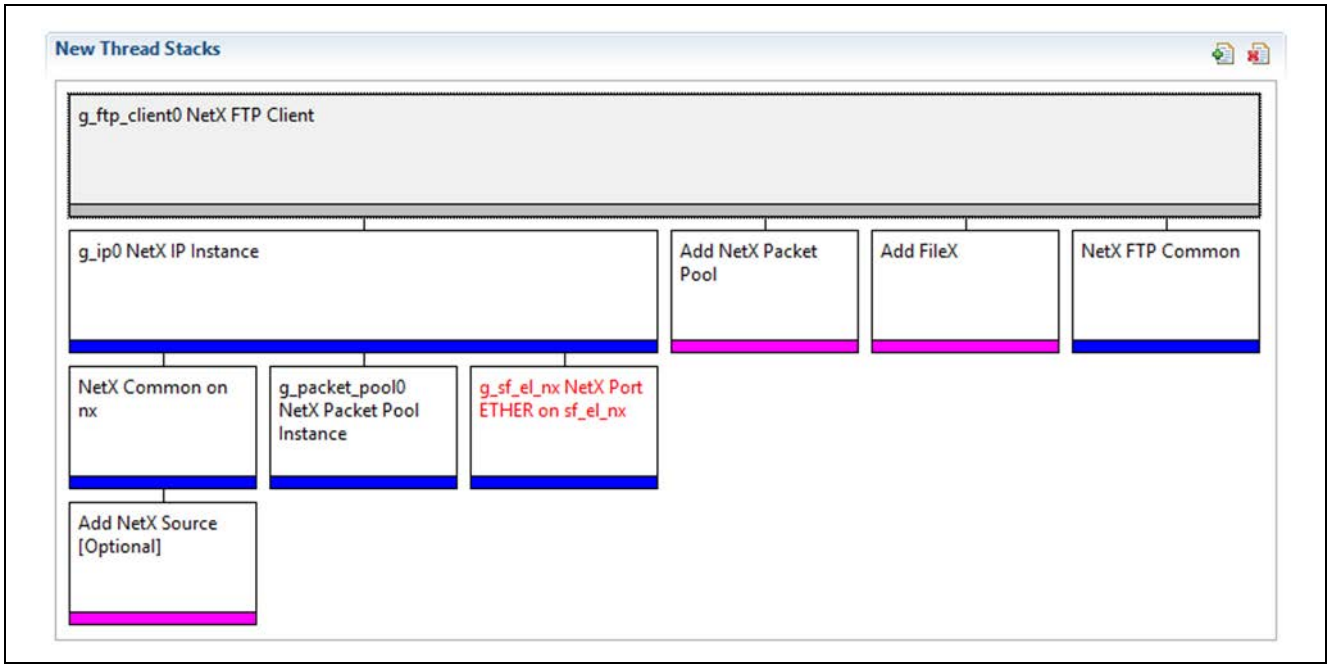


Figure 3. NetX FTP Client Stack

5. Configuring the NetX Duo FTP Client Module

The user must configure NetX Duo FTP Client modules for the desired operation. The SSP configuration window automatically identifies, by highlighting the block in red, any configuration, such as interrupts or operating modes, required for lower level modules to achieve successful operation. Only properties that can be changed without causing conflicts are available for modification. Other properties are 'locked' and unavailable for modification. They are identified with a lock icon for the 'locked' property in the ISDE Property window. 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. These settings 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 within the ISDE when configuring Interrupt Priority levels.

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

Table 9. Configuration Settings for NetX Duo FTP Client

ISDE Property	Setting	Description
Name	g_ftp_client0	Module name.
TCP socket window size (bytes)	2048	TCP socket window size selection.

Note: The setting examples and defaults in the table are for a project using the Synergy MCU Family. 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 pins for the Ethernet peripheral. 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 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 and NetX Duo FTP Client Low Level Drivers

Only a small number of settings must be modified from the default for lower level drivers and these settings are indicated in 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 tables list all the settings within the properties section for the module.

Notes on property settings:

- Properties highlighted in yellow should be modified to suit the developer's environment, in particular the *IPv4 Address.*
- It is recommended to set the *Packet Size In Bytes* to 1568 to eliminate the need for packet chaining. Packet chaining logic is fixed in subsequent SSP releases to 1.2.0
- The *IP Helper Thread Priority* should be higher (lower in number) than the FTP Client application thread for optimal performance.
- *Number of Packets in Pool* may need to be increased depending on the activity in the developer environment.

Table 10. Configuration for the NetX IP Instance

ISDE Property	Setting	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
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

Table 11. Configuration for the NetX Packet Pool Instance on g_packet_pool0

ISDE Property	Setting	Description
Name	g_packet_pool1	Module name
Packet Size in Bytes	1568	Packet size selection
Number of Packets in Pool	16	Number of packets in pool selection

Table 12. Configuration for the FileX on Block Media (not used in this module guide project)

ISDE Property	Setting	Description
Name	g_fx_media0	Module name.
Format media during initialization.	Enabled, Disabled (Default: Disabled)	Format media during initialization selection.
File System is on SDMMC	True, False (Default: True)	File System initialization selection.
Formatting Options		Formatting options selection.
Volume Name	Volume 1	Volume name selection.
Number of FATs	1	Number of FATs selection.
Directory Entries	256	Directory entries selection.
Hidden Sectors	0	Hidden sectors selection.
Total Sectors	3751936	Total sectors selection.
Bytes per Sector	512	Bytes per Sector selection.
Sectors per Cluster	1	Sectors per Cluster selection.
Working media memory size	512	Working media memory size selection.

Table 13. Configuration for NetX Duo FTP Common

ISDE Property	Setting	Description
FileX support	Enabled, Disabled (Default: Enabled)	FileX support selection
Control Type of Service	Normal, Minimum delay, Maximum data, Maximum reliability, Minimum cost (Default: Normal)	Control type of service selection
Data Type of Service	Normal, Minimum delay, Maximum data, Maximum reliability, Minimum cost (Default: Normal)	Data type of service selection
Fragmentation option	Don't fragment, Fragment okay (Default: Don't fragment)	Fragment option selection
Time to live	128	Time to live selection
Packet Queue depth	60	Packet queue depth selection

Note: FileX support is disabled, because FileX is not used in this project. It can easily be added as described in section 8, Customizing the NetX Duo FTP Client for a Target Application. If the developer chooses to, the FileX support property may be set to Enabled.

Table 14. Configuration for NetX Common on nx

ISDE Property	Setting	Description
No configurable properties		

Note: For the driver to work correctly, the properties in Table 15 highlighted in aqua *must* be modified to the settings indicated.

Table 15. Configuration for NetX Port Ether

ISDE Property	Setting	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	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) Must not be left Disabled!	Ethernet interrupt priority selection
Name	g_sf_el_nx	Module name
Channel	1	Channel selection
Callback	NULL	Callback selection

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

5.2 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 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. Table 16 lists the method used to select pins from within the SSP configuration window and Table 17 is an example showing pin selection.

Note: The Operation Mode selection mode determines what peripheral signals are available and thus what MCU pins are required.

Table 16. Pin Selection Sequence for ETHERC Module

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

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

Table 17. Pin Configuration Settings for ETHERC1

Pin Configuration Property	Settings	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 above are for a project using the S7G2 Synergy MCU and the SK-S7G2 Kit. Other Synergy MCUs and other Synergy kits may have different available pin configuration settings.

6. Using the NetX Duo FTP Client in an Application

The following example assumes a local FTP Server is already running, NetX services are available, and the FTP Client instance is created. The typical steps in running an FTP session for the NetX Duo FTP Client are:

1. Connect to the FTP Server using the nx_ftp_client_connect API. For NetX Duo, there is also nxd_ftp_client_connect which supports both IPv4 and IPv6 IP addresses.
2. Open a file for either writing or reading, using the nx_ftp_client_open API.
3. Write to a file using the nx_ftp_client_write API
4. Read from a file using the nx_ftp_client_read API
5. Close a file using the nx_ftp_client_close API

The following diagram shows these common steps in a typical operational flow.

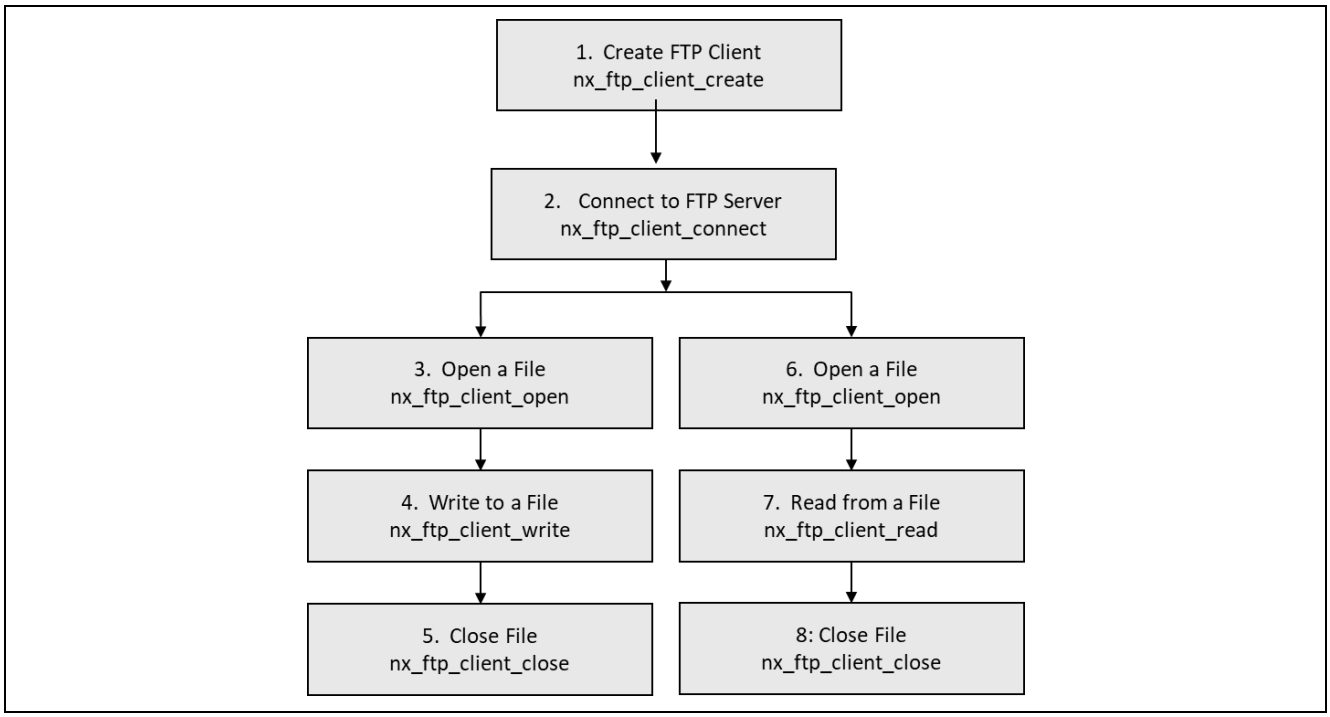


Figure 4. Flow Diagram of a Typical NetX Duo FTP Client Application

7. The NetX Duo FTP Client Application Project

The application project associated with this Module Guide demonstrates the preceding 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 Duo FTP Client Module. The code in the application thread entry function, ftp_client_thread_entry.c, uses the run_ftp_client_session service defined in NetX_Duo_FTP_Client_ MG_AP.c and NetX_Duo_FTP_Client_ MG_AP.h. These files work in either a NetX or NetX Duo environment. The application project connects with an FTP server and conducts a simple FTP session. FileZilla is a freely downloadable and very easy to use FTP server for Windows environment.

The Application Project demonstrates the typical use of the NetX Duo FTP Client APIs. The autogenerated code creates the FTP instance, IP instance, packet pool, and the thread for the FTP Client application. It initializes the network; at this point the FTP application is ready to run. The module guide project for the FTP Client thread entry waits for the network link to become enabled; in this case, the Ethernet link is using the nx_ip_status_check service. The FTP Client application then proceeds to connect to the FTP server, upload (put) a file, and retrieve (get) the same file back before disconnecting from the Server.

The following table lists the associated software and hardware target versions the Application Project uses.

Table 18. 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
SK-S7G2	v3.0 to v3.3	Development Kit
IAR EW for Renesas Synergy	8.23.3	IAR Embedded Workbench® for Renesas Synergy™
SSC	7.3.0	Synergy Standalone Configurator

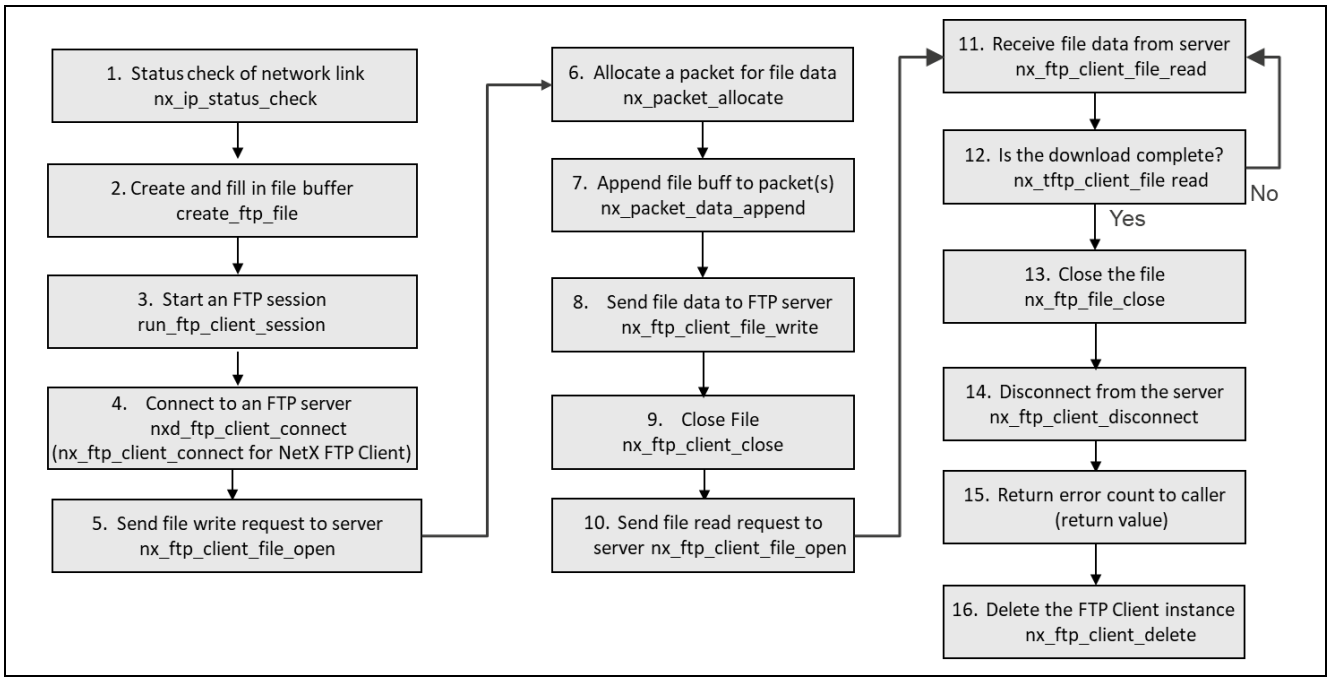


Figure 5. NetX Duo FTP Client module guide project steps

The complete module guide project, which can be downloaded using the link provided in the Reference Section at the end of this document, contains the source code files `ftp_client_thread_entry.c`, NetX_Duo_FTP_Client_`MG_AP.c` and NetX_Duo_FTP_Client_`MG_AP.h`. These files are located in the src folder in the project, once the project has been imported into the ISDE. Within the ISDE, you can open the file, and follow along while reading the following descriptions to help identify key uses of APIs.

The `ftp_client_thread_entry.c` file defines the parameters of the FTP session, including the server IP address, file name and file data, username and password, and timeout value for FTP operations. It defines the FTP server IP address, determines if IPv4 or IPv6 is used, fills in the file buffer, and defines the username and password. Before starting these steps, the FTP server should be running on the same local network as the FTP Client.

1. When the application thread starts, it verifies the network is enabled by calling nx_ip_status_check with the NX_IP_LINK_ENABLED option.
2. Create and fill the file buffer with data.
3. The thread uses this information to call the run_ftp_client_session function defined in NetX_Duo_FTP_Client_`MG_AP.c`.
   That function takes over the application and performs the following steps:
   A. Connects with the FTP server, for which it requires the username and password, using the nxd_ftp_client_connect.
      NetX FTP Client must use nx_ftp_client_connect that supports only IPv4. NetX Duo can also use nx_ftp_client_connect, but only with IPv4 server addresses
   B. Sends a request to write (upload) a file using the `nx_ftp_client_file_open` API with the `NX_FTP_OPEN_FOR_WRITE` option, specifying the filename of the file to write to (or, if the file did not previously exist, to create).
   C. Allocates a packet using nx_packet_allocate to transfer file data to the Server.
   D. Copies the file data into the packet using the `nx_packet_data_append` API (which will allocate additional packets as needed).
   E. Sends the file data to the FTP Server using the `nx_ftp_client_file_write` API. This service takes a file pointer that should point to the file allocated in step #A.
   F. Closes the file using the `nx_ftp_client_file_close` API.
   G. Sends a second request to the server, this time to read (download) a file, using the `nx_ftp_client_file_open` API with the `NX_FTP_OPEN_FOR_READ` option.

H. Receives file data packets from the FTP server using the `nx_ftp_client_file_read` API until a non-zero success status is received.
I. Verifies the file download completed if the error status is NX_FTP_END_OF_FILE (0xD7).
J. Closes the file using the `nx_ftp_client_file_close` API.
K. Disconnects from the FTP server using the `nx_ftp_client_disconnect` API.
L. Returns the number of errors during the session to the caller.
M. At this point the ftp_client_thread_entry function takes over.
N. Deletes the FTP Client using the `nx_ftp_client_delete` API.

The application ends by printing a message about the number of errors that occurred if SEMI_HOSTING is defined.

8. Customizing the NetX Duo FTP Client for a Target Application

Some configuration settings can normally be changed by the developer from those shown in the Application Project. For example, the user can easily change the configuration settings for the FTP Client in the configurator Properties window.

The FTP Client can use any supported NetX Duo interface. Consult the WiFi Framework documentation to see how to configure a NetX or NetX Duo stack to work over WiFi.

The FTP Client can use any FX_MEDIA, either physical or virtual. FileX can be configured to use different underlying media or even a RAM disk.

9. Running the NetX Duo FTP Client Application Project

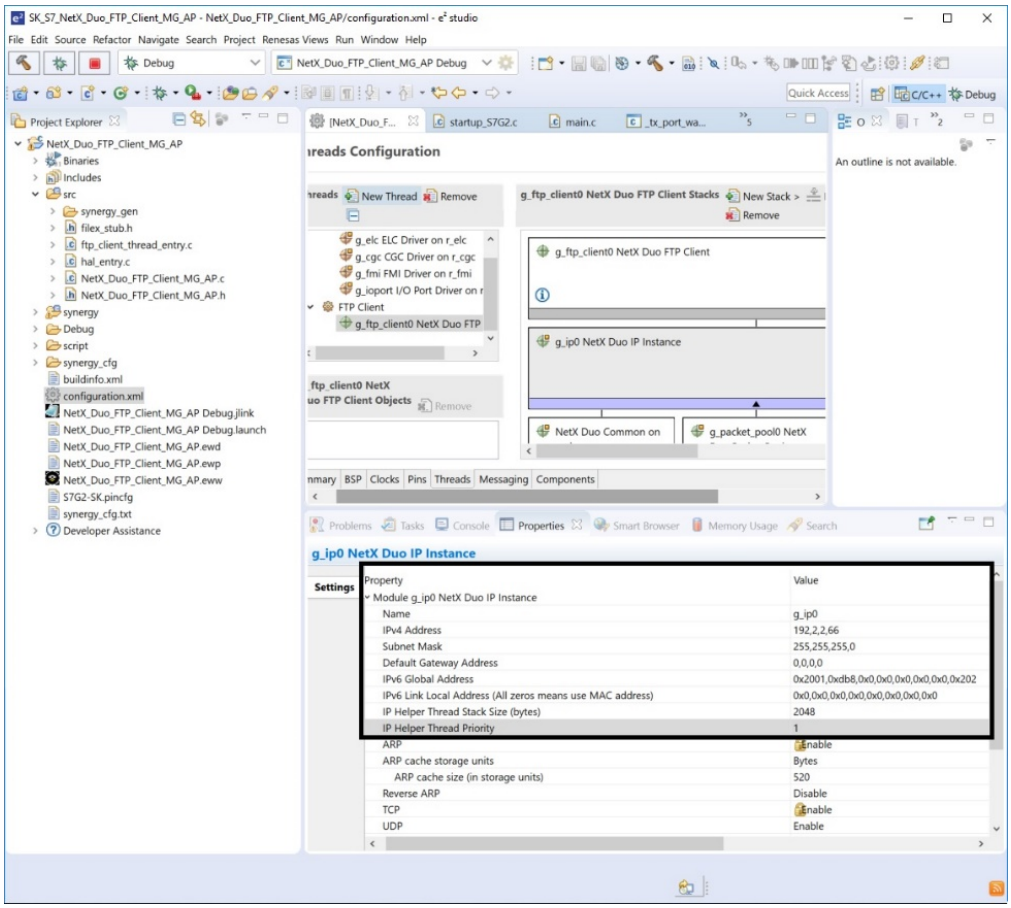
To run the NetX Duo FTP Client Application project and to see it executing on a target kit, you can simply import it into your ISDE, compile, and run debug.

To implement the NetX Duo FTP Client application in a new project, first start the FTP server on the same network as the FTP Client, and then use the following steps.

Note: The steps are in sufficient detail for someone experienced with the basic flow through the Synergy development process. If these steps are unfamiliar, see "*Getting Started with SSP*" in the *SSP User's Manual* listed in the Reference Section at the end of this document.

1. 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 ISDE or IAR EW for Synergy, and building/running the application.
2. Go to the configurator and make the following changes to the imported client project:

IPv4 Address	: 192.2.2.66
Subnet Mask	: 255.255.255.0
Default Gateway Address	: 0.0.0.0
IPv6 Global Address	: 0x2001,0xdb8,0x0,0x0,0x0,0x0,0x0,0x0,0x202
IPv6 Link Local Address	: 0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0 (All Zeros means use MAC address)

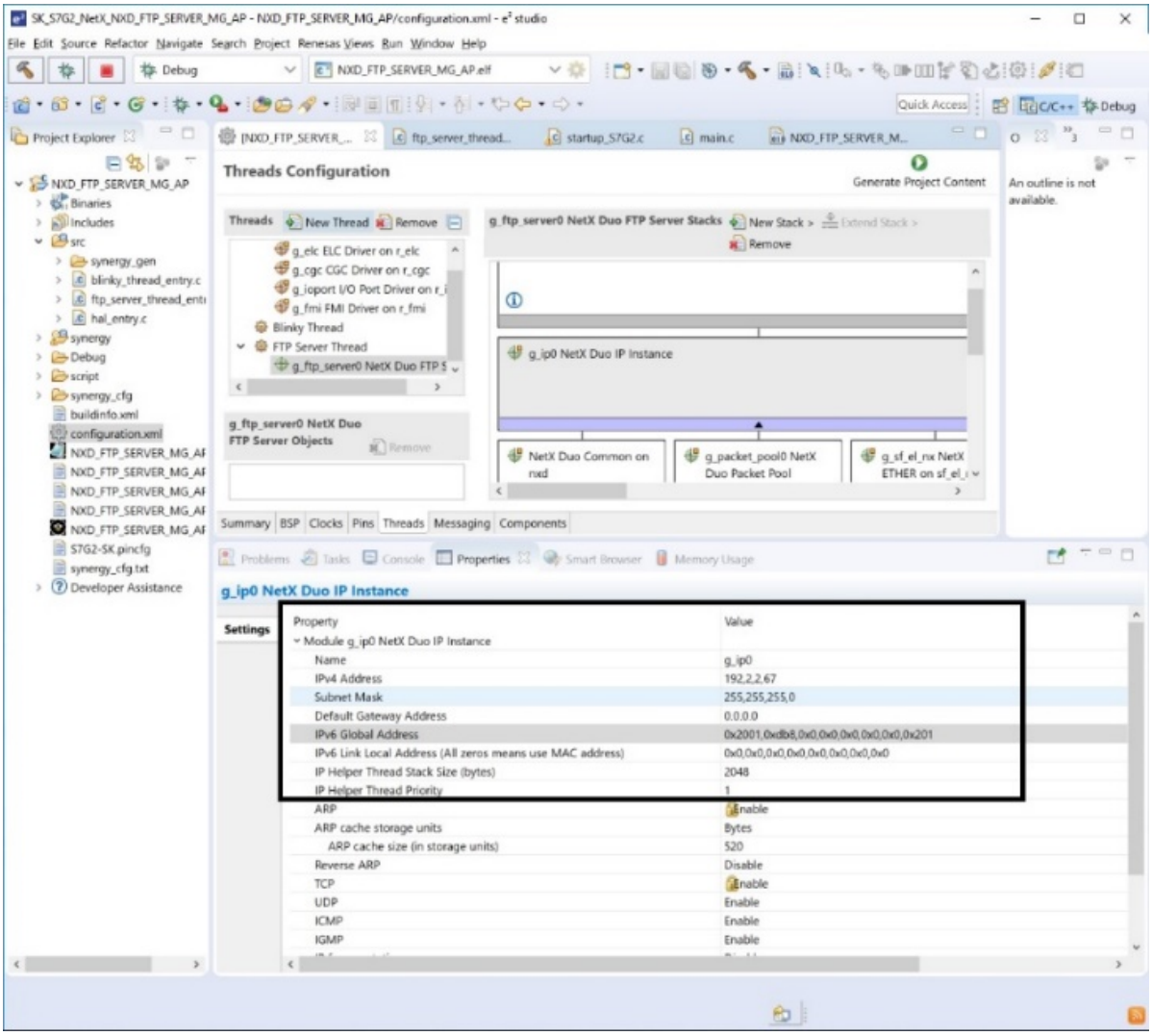


3. Set the properties listed in Section 5 for the IP instance, the IP default packet pool (shared by the FTP Client) `g_packet_pool0`, and the NetX Port Ether driver.
4. Click the **Generate Project Content** button.
5. Ensure that SEMI_HOSTING near the top of `ftp_client_thread_entry.c` has the debug output defined as enabled.
6. If using NetX Duo FTP Client, you can run the FTP Client over IPv6. To do so, define USE_IPV6 near the top of the `ftp_client_thread_entry.c` file.
7. As needed to match your environment, modify the following symbols from their default values defined in `ftp_client_thread_entry.c`.
   Note that the first symbol, USE_IPV6, is only applicable in NetX Duo FTP Client:

   - #define USE_IPV6 [by default this is not defined; define it to use FTP over IPv6]
   - #define FTP_TIMEOUT 3 [seconds]
   - #define FILE_SIZE 1400 [bytes]
   - #define FILE_NAME "test_file.txt"
   - #define USERNAME "username"
   - #define PASSWORD "password"
   - #define SERVER_ADDRESS IP_ADDRESS(192,2,2,67)
   - If using IPv6 (and only if using NetX Duo FTP Client):
   - #define SERVER_ADDRESS_V6_0 0x20010db8;
   - #define SERVER_ADDRESS_V6_1 0x0
   - #define SERVER_ADDRESS_V6_2 0x0
   - #define SERVER_ADDRESS_V6_3 0x201

8. Connect to the host PC via a micro-USB cable to J19 on SK-S7G2 device.
9. Connect the NetX device to the local network via the J11 (RJ45) connector.

10. Start the FTP server on a PC listening on port 21 in your local network (if available on your PC or in your network). Alternatively, the FTP server can run on another Synergy board (such as the SK-S7G2) and you can connect that Synergy board to the local network.
11. When you are running the FTP server application on the Synergy board, you need the following software configuration completed to test the client application. For the FTP server application, see the *NetX Duo™ FTP Server Module Guide* - Application Project.



- IPv4 Address : 192.2.2.67
- Subnet Mask : 255.255.255.0
- Default Gateway Address : 0.0.0.0
- IPv6 Global Address : 0x2001,0xdb8,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x201
- IPv6 Link Local Address : 0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0 (All zeros means use the MAC address)

The Server application kit must have USB memory stick with “test_file.txt” created and plugged into J6 (in case of SK-S7G2) connector USB Host port.

12. Check that the file to be uploaded does not already exist in the FTP server host. Some FTP servers will overwrite the file, while others will not accept the write request.

13. Note these considerations when running the application

— Same Subnet IPv4 but different IP address (client, server).
— Same Subnet Mask.
— Same Default Gateway Address.

— Same Subnet IPv6 but different IP address (client, server).
— IPv6 Link Local address should be all zeros.
— Must run Server application ready before execute client application.

14. Start to debug the application.
15. The output can be viewed in the Renesas Debug Console.

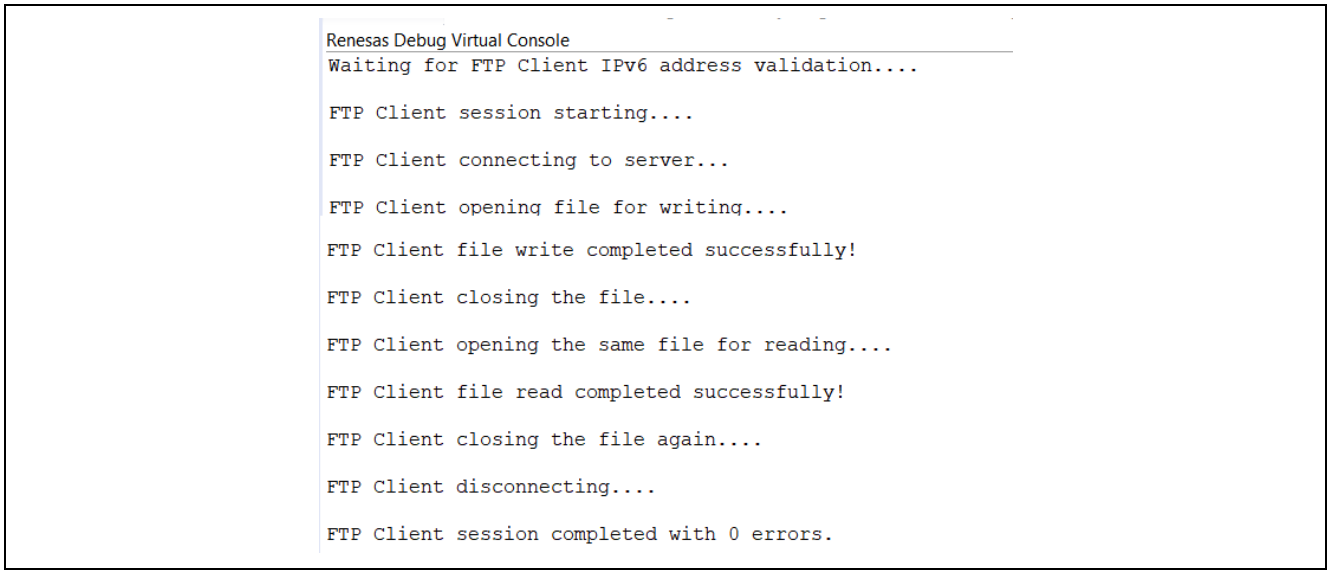


Figure 6. NetX Duo FTP client debug output from e² studio ISDE

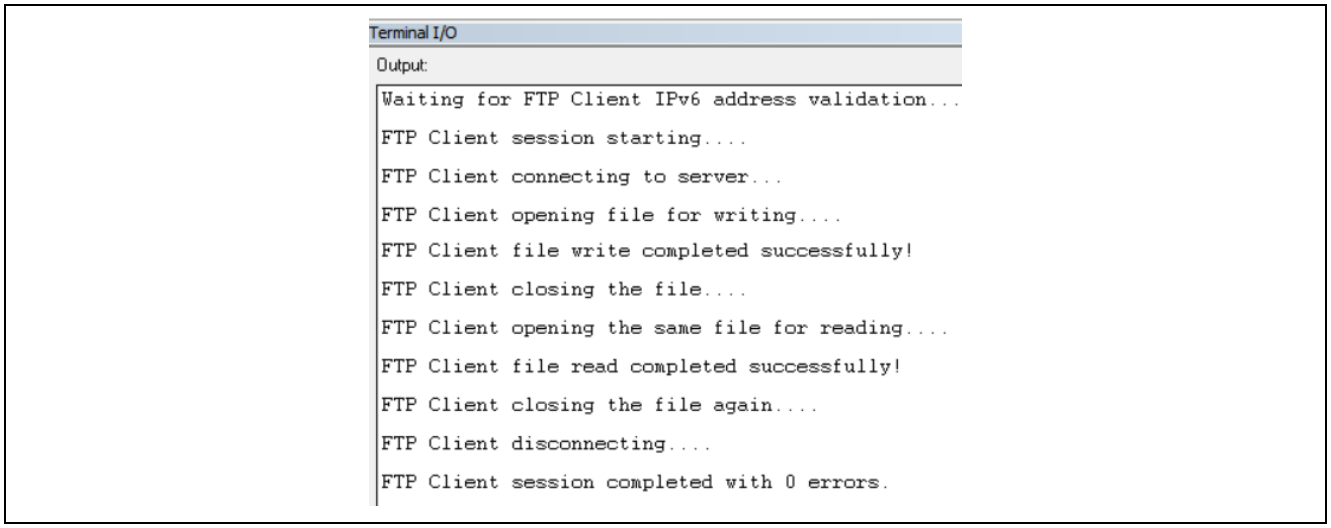


Figure 7. NetX Duo FTP Client debug output from IAR ISDE

10. 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 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, show how additional development time savings can be achieved 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.

An FTP Client allows you to upload and retrieve files from FTP Servers. The File Transfer Protocol is standardized and widely supported. It is a convenient light way of gaining network access to files. As seen in this project, the use of the FTP Client module requires only a few steps and provides full capabilities.

11. Next Steps

After you run the simple FTP Client project successfully, consider adding features and depth to the project.

To get around the problem of a firewall blocking file transfer, NetX Duo FTP Client supports passive FTP mode. In passive mode, the client establishes *both* channels, not just the control channel.

The module guide uses packets sent out over Ethernet to transfer files. However, a USB mass storage device can be also be used as a location for files to be up-or downloaded. Adding a USB mass storage device also requires adding the FileX stack for file handling.

Other Application Projects and Application Notes that demonstrate FTP Client use can be found in the Reference section at the end of this document.

12. Reference Information

1. Start Developing with the Renesas Synergy™ Platform
   Provides an overview of each of the elements of the Renesas Synergy Platform
2. SSP User's Manual
   A detailed reference manual for all SSP modules, APIs, structures and error codes.
3. *NetX™ and NetX Duo™ FTP Client Module Guide*
   The application project associated with this Module Guide
4. How do I Use Printf() with the Debug Console in the Synergy Software Package
   Provides step by step process to add printf() with the debug console in the SSP.

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

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Jan.10.18	—	Initial version
1.01	Jun.04.19	—	Updated to 1.6.0. Added step for Synergy FTP server MG.

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/.