Renesas Synergy™ Platform

NetX Port Ether 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 efficient starting point. References to more detailed API descriptions and suggestions of other application projects that illustrate more advanced uses of the module are available on the Renesas Synergy Knowledge Base (as described in the References section at the end of this document) and should be valuable resources for creating more complex designs.

The Synergy NetX Port Ether module (`sf_el_nx`) for NetX and NetX Duo is integrated into the SSP. Its function is to interface the generic NetX and NetX Duo software with the hardware. This module includes the MAC driver, the PHY driver, additional glue logic and utility functions.

Note: Unless otherwise stated there is no difference is how this module works in NetX or NetX Duo projects.

This document is divided into sections which can be read independently by a more experienced Synergy developer or in series by developers new to the Synergy Platform.

Contents

1. NetX Port Ether Module Features

- NetX services are implemented as a library, so only code that is needed is added to the project
  - For most applications, this results in an instruction image from between 5 Kbytes and 30 Kbytes. With IPV6 and ICMPv6 enabled, the size is from 30 Kbytes to 45 Kbytes
- NetX supports a variety of RFCs including the following:
  - RFC 1112 Host Extensions for IP Multicasting (IGMPv1)
  - RFC 1122 Requirements for Internet Hosts - Communication Layers
  - RFC 2236 Internet Group Management Protocol, Version 2
  - RFC 768 User Datagram Protocol (UDP)
  - RFC 791 Internet Protocol (IP)
  - RFC 792 Internet Control Message Protocol (ICMP)
  - RFC 793 Transmission Control Protocol (TCP)
  - RFC 826 Ethernet Address Resolution Protocol (ARP)
  - RFC 903 Reverse Address Resolution Protocol (RARP)
  - RFC 2460 Internet Protocol v6 (IPv6) Specification (NetX Duo only)
  - RFC 4443 Internet Control Message Protocol (ICMPV6) (NetX Duo only)
  - RFC 4861 Neighbor Discovery for IPv6 (NetX Duo only)
  - RFC 4862 IPv6 Stateless Address Auto Configuration (NetX Duo only)
- Packet-based zero-copy implementation of TCP/IP (buffers are not copied inside NetX as they travel across the stack or from the stack to the user application, freeing up memory and processing cycles for example, significantly improves transmission rates)
- Fast UDP processing

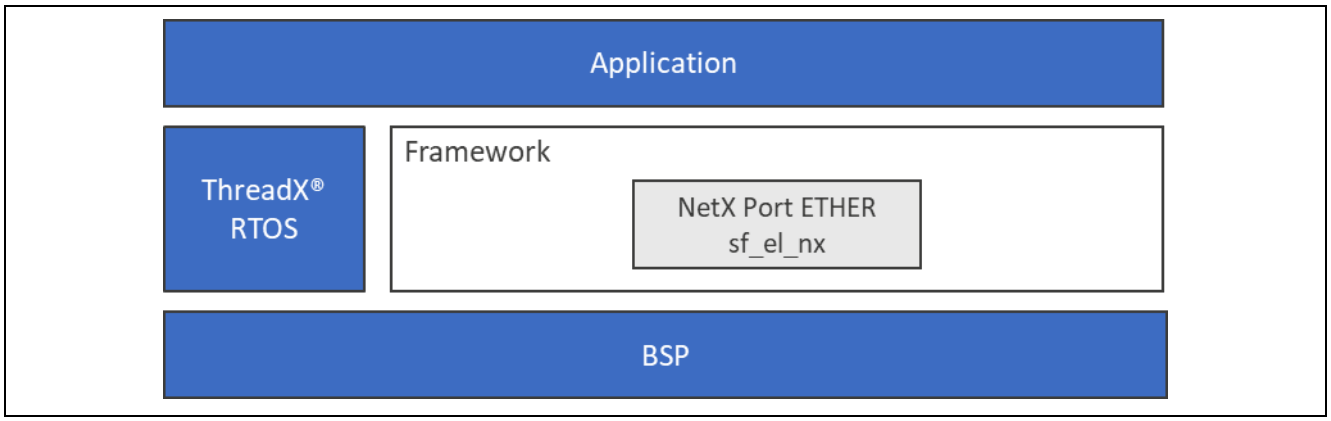

Figure 1. NetX Port Ether Module Block Diagram

2. NetX Port Ether Module APIs Overview

The NetX Port Ether module has a narrow API, used internally by NetX and by the module itself. It includes the Ethernet driver entry point (`nx_ether_driver_eth0`, `nx_ether_driver_eth1`), the Ethernet interrupt handler and other functions used internally by the module but externally visible.

3. NetX Port Ether Module Operational Overview

The NetX Port Ether module is a high-performance real-time implementation of the NetX Ethernet driver for the Renesas Synergy software and Synergy Ethernet IP.

Note: NetX assumes the existence of ThreadX and depends on its thread execution, suspension, periodic timers, and mutual exclusion facilities.

Network Driver

Each IP instance in NetX has a primary interface which is identified by its device driver specified in the nx_ip_create service. The network driver is responsible for handling various NetX requests, including packet transmission, packet reception, and requests for status and control.

For a multi-home system, the IP instance can be configured for multiple interfaces, each with an associated network driver that performs these tasks for the respective interface. The network driver must also handle asynchronous events occurring on the media. Asynchronous events from the media include packet reception, packet transmission completion, and status changes. NetX provides the network driver with several access functions to handle various events. These functions are designed to be called from the interrupt service routine portion of the network driver. For IP networks, the network driver should forward all ARP packets received to the `_nx_arp_packet_deferred_receive` internal function. All RARP packets should be forwarded to the `_nx_rarp_packet_deferred_receive` internal function.

There are two options for IP packets:

- If fast dispatch of IP packets is required, incoming IP packets are forwarded to `_nx_ip_packet_receive` for immediate processing. This greatly improves NetX performance in handling IP packets.
- IP packets are forwarded to `_nx_ip_packet_deferred_receive`. This service places the IP packet in the deferred processing queue where it is then handled by the internal IP thread, which results in the least amount of ISR processing time.

The network driver can also defer interrupt processing to run out of the context of the IP thread. In this mode, the ISR shall save the necessary information, call the internal function `_nx_ip_driver_deferred_processing`, and acknowledge the interrupt controller. This service notifies IP thread to schedule a callback to the device driver to complete the process of the event that causes the interrupt.

Some network controllers are capable of performing TCP/UDP/IP header checksum computation and validation in hardware without taking up valuable CPU resources. To take advantage of the hardware capability feature, NetX provides options to enable or disable various software checksum computation at compilation time, as well as turning on/off checksum computation at run time. See the “NetX Network Drivers” section in the *NetX User Guide* for more detailed information on writing NetX network drivers.

3.1 NetX Port Ether Module Important Operational Notes and Limitations

3.1.1 NetX Port Ether Module Operational Notes

A summary of some key operational notes is given as follows. Refer to the *NetX User Guide for the Renesas Synergy™ Platform* and *NetX Duo User Guide for the Renesas Synergy™ Platform* for details on each of these topics. The source code symbol is provided in addition to the NetX Source property for each feature that follows.

For example, to change the number of physical network interfaces, you can either set the Maximum Physical Interfaces property in the NetX Source or NetX Duo Source element, or you can define the source code symbol `NX_MAX_PHSYCIAL_INTERFACES` directly. In either case, it is still necessary to include the NetX and NetX Duo Source component, generate the project files and to rebuild the NetX library.

NetX Source Level settings

The following options are in the NetX and NetX Duo Source components:

- **Multiple Network Interface Support**

NetX supports systems connected to multiple physical devices using a single IP instance. Each physical interface is assigned to an interface control block in the IP instance. Applications wishing to use a multi-home system must define the value for the **Maximum Physical Interfaces** property (or `NX_MAX_PHSYCIAL_INTERFACES`) to the number of physical devices attached to the system, and then rebuild the NetX library. If the library is not rebuilt, this has no effect. The default value is 1.

- **Loopback Interface**

The loopback interface is a special network interface without a physical link attached to. The loopback interface allows applications to communicate using the IP loopback address 127.0.0.1 To use a logical loopback interface, either set the **Loopback** property to enabled (the default setting) or ensure the symbol `NX_DISABLE_LOOPBACK_INTERFACE` is not defined.

- **Enabling IPv6 (NetX Duo only)**

By default, IPv6 is enabled on the IP instance in NetX Duo. IPv6 can be disabled or enabled for the IP instance by setting the **NetX Duo IPV6 Support** property in the NetX Duo Source component. Other IPv6 related properties may be enabled (checksum on ICMPv6 packets, Duplicate Address Detection, and so on). See the *NetX Duo User Guide for the Renesas Synergy™ Platform* for details on IPv6 networking.

Interface Control Blocks: The number of interface control blocks in the IP instance is the number of physical interfaces (defined by `NX_MAX_PHYSICAL_INTERFACES`) plus the loopback interface if it is enabled. The total number of interfaces is defined in the symbol `NX_MAX_IP_INTERFACES`. This value should **not** be modified directly. NetX will define this symbol internally based on the `NX_MAX_PHSYCIAL_INTERFACES` and `NX_DISABLE_LOOPBACK_INTERFACE` settings.

NetX Port Ether Settings

- **Channel**

Channel determines which interface the Ethernet driver is applied to. It may need to be modified from the default value of zero. See Section 5 “Configuring the NetX Port ETHER”.

- **Channel 0/1 Phy Reset Pin**

The Channel 0/1 Phy Reset Pin property depends on the value of the Channel property above. It may need to be modified from the default value. See section 5, Configuring the NetX Port Ether Module.

- **Channel 0/1 MAC Address High Bits**
- **Channel 0/1 MAC Address Low Bits**

These properties set the device MAC address on the respective channel interface at compile time. To set the MAC address at run time, see the description of the Callback property below.

- **Callback**

The Callback property defines the user defined callback function which sets MAC address at runtime (network link initialization). The default value is NULL and the MAC address is assigned using the Channel MAC Address High/Low Bits settings.

- **Name**

The Name property names the instance of the NetX Port ETHER driver. For an IP instance configured with multiple network interfaces, the application must add an instance of the driver and give it a unique name (or compile errors will result). The default name is `g_sf_el_nx`.

- **Ethernet Interrupt Priority**

This property sets the driver interrupt priority. The default value is priority 4. The dropdown list indicates valid and invalid priorities depending on the MCU target.

- **Number of Receive Buffer Descriptors**
- **Number of Transmit Buffer Descriptors**

These properties set the number of buffer descriptors (BDs) for receiving and transmitting packets. When the Receive BDs are initialized, the driver allocates a packet for each BD. Therefore, the number of receive BDs should not deplete the IP instance packet pool from which it allocates packets.

The NetX Port ETHER driver supports packet chaining for both receiving and transmitting packets (where packets are chained in the application layer). If a packet is received on the network that exceeds the size of the IP default packet pool payload, the driver can allocate additional packets and process the incoming packet as a packet chain.

- **Parameter Checking**

Parameter Checking provides low-level error checking for each component of the project hardware layer. By default, it is set to Default (BSP) which means this property inherits the same property in the BSP.

3.1.2 NetX Port Ether Module Limitations

Refer to the most recent SSP Release Notes for any additional operational limitations for this module.

4. Including the NetX Port Ether Module in an Application

This section describes how to include the NetX Port Ether 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 Port Ether 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 FTP Server is `g_sf_el_nx`. This name can be changed in the associated **Properties** window).

Table 1. NetX Port Ether Module Selection Sequence

Resource	ISDE Tab	Stacks Selection Sequence
g_sf_el_nx NetX Port ETHER on sf_el_nx	Threads	New Stack> Framework> Networking> NetX Port ETHER on sf_el_nx

When the NetX Port Ether 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 and this is indicated in the block with the inclusion of this text. If the addition of lower level drivers is required, the module description will include "Add" in the text. Clicking on any pink banded modules will bring up the "New" icon and then will show the possible choices.

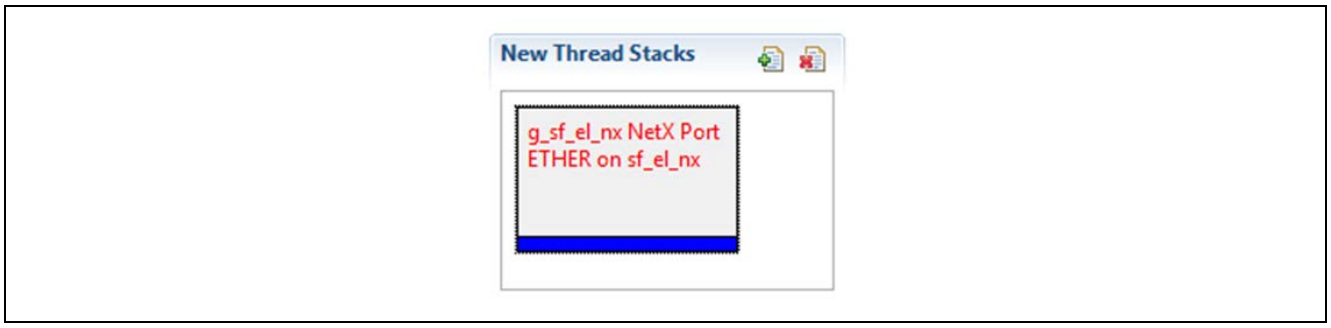

Figure 2. NetX Port Ether Module Stack

As configured below, the NetX and NetX Duo Auto IP automatically uses the IP instance packet pool `g_packet_pool0`. The driver is set to the NetX Port ETHER driver, `g_sf_el_nx`.

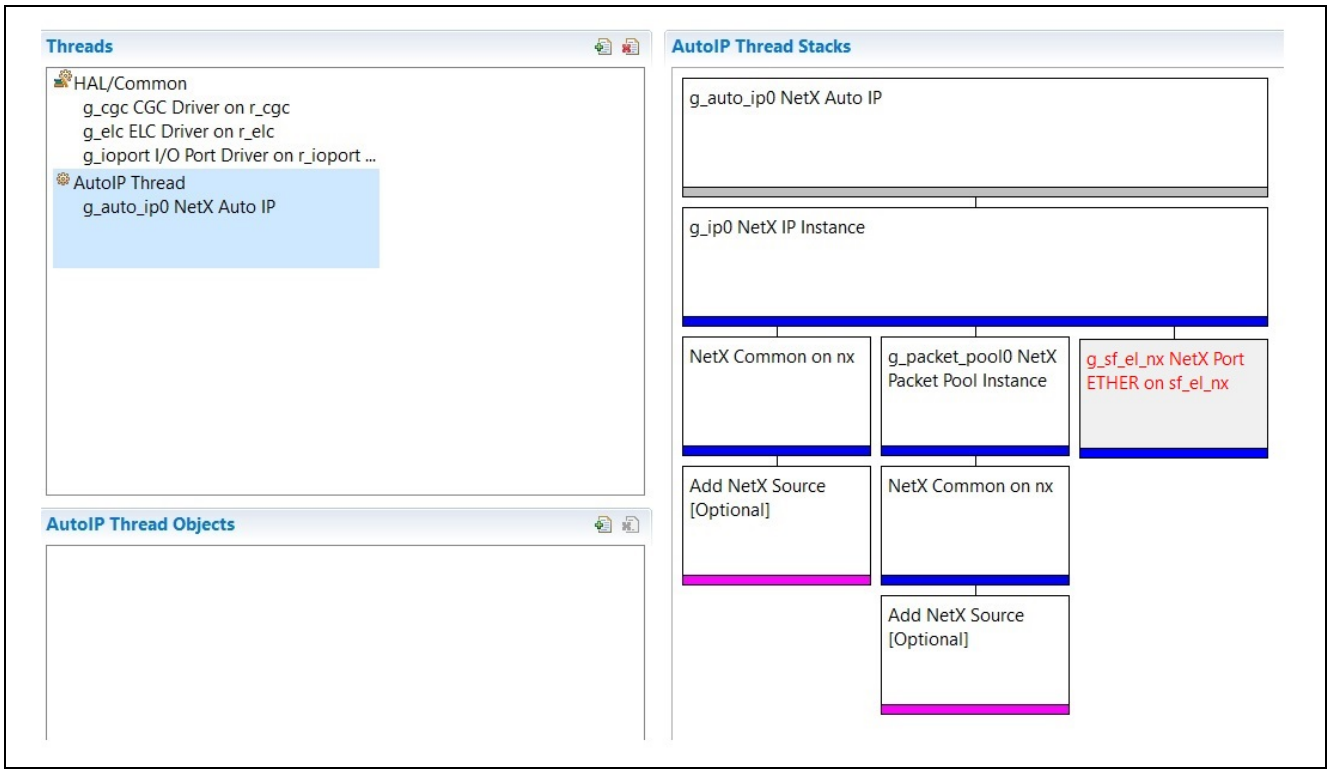

Figure 3. NetX Port Ether Application using NetX Auto IP

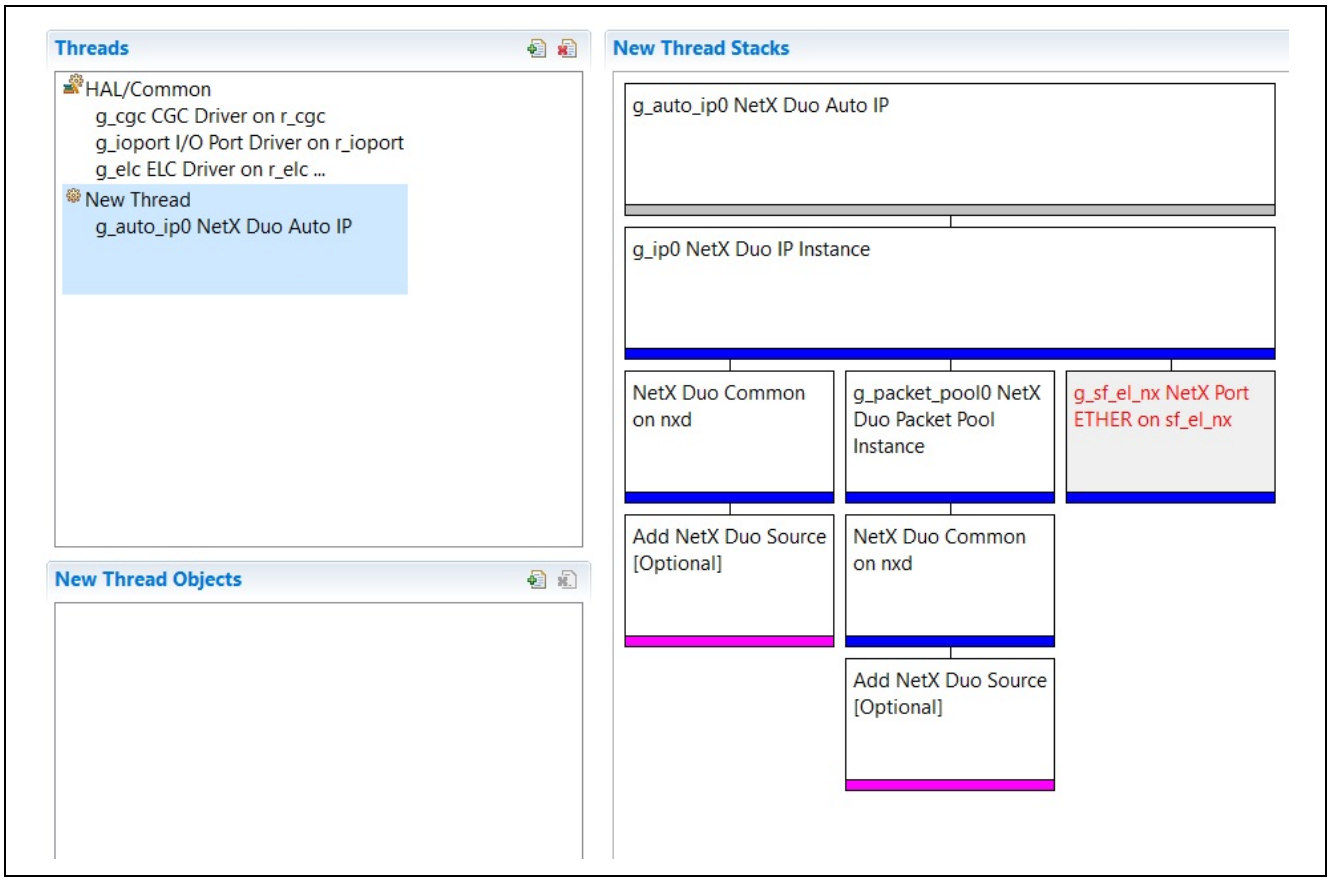

Figure 4. NetX Port Ether Application using NetX Duo Auto IP

5. Configuring the NetX Port Ether Module

The NetX Port Ether module must be configured for the desired operation. The SSP configuration window will automatically identify (by highlighting the block in red) any required configuration selections, such as interrupts or operating modes, which must be configured for lower-level modules for successful operation. Only those properties that can be changed without causing conflicts are available for modification. Other properties are 'locked' and unavailable for changes and these 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 within the **Properties** window of the associated module. Simply select the indicated module and then view the **Properties** window. The interrupt settings are often toward the bottom of the properties list, so scroll down until they become available. Also, note that the interrupt priorities listed in the **Properties** window in the ISDE will include an indication as to the validity of the setting based on the MCU targeted (CM4 or CM0+). This level of detail is not included in the following configuration properties tables but is easily visible with the ISDE when configuring interrupt-priority levels.

Note: You may want to open your ISDE, create the module, and explore the property settings in parallel with reviewing 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 SSP.

Table 2. Configuration Settings for the NetX Port Ether Module

Parameter	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 1 Phy Reset Pin	IOPORT_PORT_07_PIN_06	Channel 1 Phy reset pin selection
Channel 1 MAC Address High Bits	0x00002E09	Channel 1 MAC address high bits selection
Channel 1 MAC Address Low Bits	0x0A0076C8	Channel 1 MAC address low bits selection
Number of Receive Buffer Descriptors	8	Number of receive buffer descriptors selection
Number of Transmit Buffer Descriptors	32	Number of transmit buffer descriptors selection
Ethernet Interrupt Priority	Priority 0(highest)-15(lowest), Disabled Default: Priority 5	Ethernet interrupt priority selection
Name	g_sf_el_nx	Module name
Channel	0	Channel selection
Callback	NULL	Callback selection
Internal thread stack size (bytes)	4096	Internal thread stack size
Name of Login Function	ftp_login	Name of Login selection
Name of Logout Function	ftp_logout	Name of Logout selection

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

6. Using the NetX or NetX Port Ether Module in an Application

The NetX Port Ether Module should be used in combination with NetX. The steps performed by the Synergy automatically in a NetX application support are:

1. Initialize the system with `nx_system_initialize`.
2. Create a packet pool with `nx_packet_pool_create`. This creates the IP default packet pool, to be used by the IP instance and the Ethernet driver.
3. Create an IP Instance with `nx_ip_create`.
4. Enable ARP with `nx_arp_enable`.
5. Create an Auto IP instance with `nx_auto_ip_create`.
6. The following steps are performed directly by the application to set up and run the Auto IP thread task.
7. Check if the network link is enabled with the `nx_ip_interface_status_check` API.
8. Verify the device does not have an IP address by calling the `nx_ip_interface_address_get` API.
9. Set the IP address notification callback by calling the `nx_ip_address_change_notify` API.
10. Start the Auto IP instance with the `nx_auto_ip_start` API.
11. Wait for the IP address change callback to set the flag that the IP instance has an IP address.
12. Stop the Auto IP task by calling the `nx_auto_ip_stop` API.
13. Verify the IP instance has a non-zero IP address by calling the `nx_ip_interface_address_get` API again.

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

Figure 5. Flow Diagram of a Typical NetX Port Ether Module Application

7. NetX Port Ether Module Application Project

The application project associated with this module guide demonstrates the aforementioned 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 Auto IP module. You can also read over the code in `_thread0_entry.c` which is used to illustrate the NetX Auto IP module APIs in a complete design.

The application project demonstrates the typical use of the NetX Auto IP module APIs. The application project main thread entry starts the Auto IP thread task, obtains the IP address, and stops the Auto IP thread as it is no longer needed. The contents of the resource are printed using the semi-hosting utility and can be viewed on the host machine.

Table 3. Software and Hardware Resources Used by the Application Project

Resource	Revision	Description
e² studio	v7.5.1 or later	Integrated Solution Development Environment
SSP	v1.7.0 or later	Synergy Software Platform
IAR EW for Renesas Synergy	v8.23.3 or later	IAR Embedded Workbench® for Renesas Synergy™
SSC	v7.5.1 or later	Synergy Standalone Configurator
SK-S7G2	v3.0 to v3.1	Starter Kit

The following diagram shows a simple operational flow of the application project:

Figure 6. NetX Port Ether Module Application Project Flow Diagram

The `auto_thread0_entry.c` file is located in the project once it has been imported into the ISDE. You can open this file within the ISDE and use the following description to help identify key uses of APIs.

The application checks if it has an IP address. If it does not, it prepares to use Auto IP; it sets an IP address change notification, so it knows if there is a change in the device IP address. Then, it starts the Auto IP thread task and waits to be notified. Once notified, it verifies it has a non-zero IP address and now no longer needs the Auto IP thread task to run, so it stops it (suspends the thread actually). It could delete the Auto IP instance, but it might wish to use it again at a later time.

The debug output uses the semi-hosting properties of the Renesas Virtual Debug Console. For debug output, `SEMI_HOSTING` is in the preprocessor list. Right-click on the project in **e² Studio -> Properties -> C/C++ Build -> Setting -> Cross ARM C Compiler -> Preprocessor**.

Note that all the APIs in this demo use the 'interface' API. For example, instead of using the `nx_ip_address_get` API, it uses the `nx_ip_interface_address_get` API with the second input as the interface index. Since this demo is using the primary interface, in all cases the interface index specified is zero.

Note: It is assumed that you are familiar with using `printf()` and the debug console in the Synergy Software Package. If you are unfamiliar with `printf()`, refer to the "How do I Use Printf() with the Debug Console in the Synergy Software Package" knowledge base article, available as described in the references section at the end of this document. Alternatively, the user can see results via the watch variables in the debug mode.

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

Table 4. Modified Configuration Settings for NetX Port Ether Module on sf_el_nx

ISDE Property	Value	Description
Parameter Checking	Enabled	Enable or disable the parameter checking
Channel 1 Phy Reset Pin	IOPORT_PORT_08_PIN_06	Channel 1 Phy reset pin selection
Ethernet Interrupt Priority	Default: Priority 5	Ethernet interrupt priority selection
Channel	1	Channel selection

Table 5. Creating a NetX and NetX Duo Auto IP Instance

Resource	ISDE Tab	Stacks Selection Sequence
g_auto_ip0 NetX Auto IP	Threads	New Stack> X-Ware> NetX> Protocols> NetX Auto IP
g_auto_ip0 NetX Duo Auto IP	Threads	New Stack> X-Ware> NetX Duo> Protocols> NetX Duo Auto IP

Table 6. Configuration Settings for NetX and NetX Duo Auto IP

ISDE Property	Value	Description
Name	g_auto_ip0	Module instance name

Table 7. Configuration for the NetX and NetX Duo IP Instance

ISDE Property	Value	Description
Name	g_ip0	Module name
IPv4 Address (use commas for separation)	*0,0,0,0*	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	520	ARP Cache Size in Bytes selection
Reverse ARP	Enable, Disable Default: Disable	Reverse ARP selection
TCP	Enable, Disable Default: Disable	TCP selection
UDP	*Enable*	UDP selection
ICMP	Enable, Disable Default: Disable	ICMP selection
IGMP	Enable, Disable Default: Disable	IGMP selection

8. Customizing the NetX Port Ether Module for a Target Application

Some configuration settings will normally be changed by the developer from those shown in the application project. For example, the user can modify the MAC address by setting the *Channel 0/1 MAC Address High Bits* and *Channel 0/1 MAC Address Low Bits properties*. To set the MAC address at runtime, modify the *Callback* property of the NetX Port Ether component to the name of the user defined function to set the MAC address.

```
void my_mac_address_callback(nx_mac_address_t *p_mac_config);
```

where `p_mac_config` is defined as:

```
/* MAC Address structure */
typedef struct st_nx_mac_address
{
    ULONG nx_mac_address_h;
    ULONG nx_mac_address_l;
} nx_mac_address_t;
```

During link initialization, this callback will be invoked.

Table 8. Set a user-defined callback by modifying the NetX Port ETHER properties as follows.

ISDE Property	Value	Description
Callback	my_mac_address_callback	User defined callback

For data throughput intensive applications, the developer can modify the number of buffer descriptors. Be aware that an increase in the *Number of Receive Buffer Descriptors* property requires additional packets from the packet pool used by the driver (the IP default packet pool, g_packet_pool0 most typically). During initialization, a packet is allocated for each RX buffer descriptor as part of the ‘zero copy’ feature of the NetX Port ETHER driver. Beyond a certain point, adding more RX buffers does not increase throughput. This number will depend on the hardware and network conditions and is determined empirically.

9. Running the NetX Port Ether Module Application Project

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

To implement the NetX Port Ether module application in a new project, follow the steps below for defining, configuring, auto-generating files, adding code, compiling, and debugging on the target kit. Following these steps is a hands-on approach that can help make the development process with SSP more practical.

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

To create and run the NetX Port Ether/Auto IP application project, simply follow these steps:

1. Create a new Renesas Synergy™ project for the S7G2-SK called NetX_Port_ETHER_ AP.
2. Select the **Threads** tab.
3. Click on the **+** icon in the threads panel.
4. Set the stack size to 2048 and set the thread instance to autoip_thread0.
5. In the thread stack pane, click on the + icon and choose X-Ware -> NetX -> Protocols -> NetX AutoIP (or X-Ware -> NetX Duo -> Protocols -> NetX Duo Auto IP for NetX Duo applications).
6. Set the properties for the NetX Auto IP instance and the **NetX Port ETHER** driver.
   Make sure to set the **Ethernet Interrupt Priority**. Its default value is Disabled. For the SK-S7G2 board, choose **Channel 1** and the Channel 1 Phy Reset Pin should be **IOPORT_PORT_08_PIN_06**. These properties default to **0** and **IOPORT_PORT_07_PIN_06**, respectively.
7. Click on the **Generate Project** Content button.
8. Add the code from the supplied project file "autoip_thread0_entry.c" into the project thread entry function making sure the thread entry function matches what the name of the application thread is.
   For example, Synergy will generate a my_auto_ip_thread_entry function automatically for a thread of the name "my_auto_ip_thread".
9. To step through code, modify the optimization to **None -O0** (the default to this value is -O2 for Optimize more):
   - Right-click the project and choose **Properties** -> **C/C++ Build** -> **Settings** -> **Optimization**. Set the Optimization Level to **None** (-O0).
   - For better performance and reduced code size, reset this to **-O2**.
10. Enable debug printf output (optional).
11. Right-click the project and choose **Properties** -> **C/C++ Build** -> **Settings** -> **Cross ARM C Compiler** -> **Preprocessor** and add a define in the Defined symbols pane by clicking on the **+** icon. Type in **SEMI_HOSTING**.
12. Right click the project and choose **Build Project**.
13. Connect to the host PC via a micro USB cable to J19 on SK-S7G2.
14. Connect the Ethernet cable to J11 on the SK-S7G2 board.
15. Right click the project and choose Debug as -> **Renesas GDB Hardware Debugging**.
16. Run the application.
17. The output, if there are no errors, can be viewed in the Renesas Debug Console as follows.

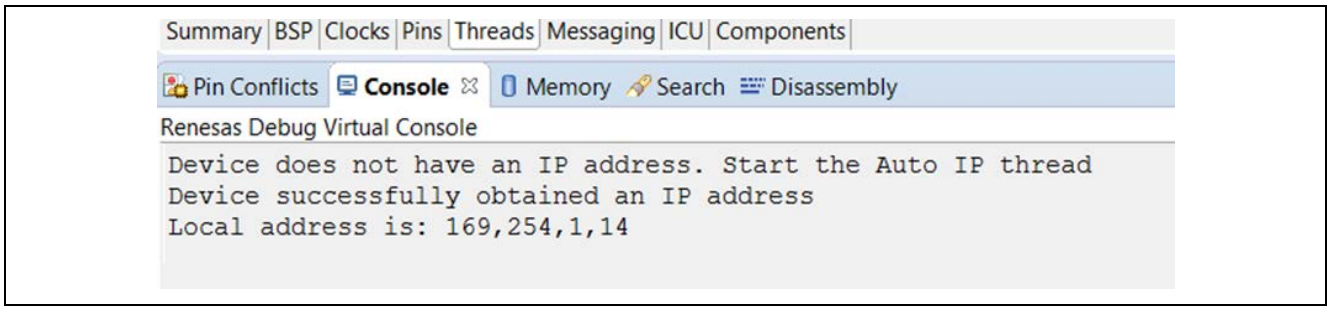

Figure 7. Example Output from NetX Port ETHER / Auto IP Application Project

10. NetX Port Ether 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 low-level modules. 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 Port Ether Module Next Steps

After you have mastered a simple NetX Port ETHER project you may want to review a more complex example. The same general set up for the Ethernet driver can be applied to any NetX socket application or NetX protocol, the only difference being the top-level modules added into the project.

Table 9. Setting a different channel for the NetX Port ETHER driver by modifying the Channel property

ISDE Property	Value	Description
Channel 1 Phy Reset Pin	IOPORT_PORT_07_PIN_06	Channel 1 Phy reset pin selection may need to be modified if using a different channel. For the DK-S7G2, if using Channel 0, the default setting is correct.
Channel	0	Channel selection. Note that this option is available, depending on which board kit the project uses. The DK-S7G2 has two Ethernet network ports on Channels 0 and 1. The SK-S7G2 board only has one Ethernet port on Channel 1

When adding another physical network interface, the driver operates on, a second driver instance must be added to the project and it must have a unique name. The NetX Port Ether Module has separate properties for primary and secondary interfaces. One set is for Channel 0 and the other set is for Channel 1. Each channel has a property setting for **Phy Reset Pin**, **MAC Address High Bits**, and **MAC Address Low Bits**. NetX and NetX Duo have API for adding network interfaces. For details, see the NetX and NetX Duo User Guide for the Renesas Synergy™ Platform.

Table 10. Run two Ethernet network interfaces by creating a second instance of NetX Port ETHER.

ISDE Property	Value	Description
Channel 0 Phy Reset Pin	IOPORT_PORT_07_PIN_06	Channel 1 Phy reset pin
Channel	0	Channel selection.
Name	g_sf_el_nx_1	Modify name if there is already a driver instance with the name g_sf_el_nx.

Note: In this case, the second instance of the Ethernet driver for the secondary interface runs on Channel 0 and works with the default Channel 0 Phy Reset setting. Not shown: the MAC address bits must be different from the MAC address for Channel 1.

12. NetX Port Ether Module Reference Information

SSP User Manual: Available at www.renesas.com/us/en/products/synergy/software/ssp.html as an SSP distribution package, and also as a pdf from the Synergy Gallery.

Links to sf_el_nx module reference materials and resources are available on the Synergy Knowledge Base: https://en-support.renesas.com/knowledgeBase/16977543.

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	May.15.17	-	Initial Release
1.01	Sep.13.17	-	Update to Hardware and Software Resources Table
1.02	Feb.08.19	-	Update to Hardware and Software Resources Table
1.10	Apr.29.19	-	Updated to SSP v1.6.0
1.11	Oct.11.19	-	Updated to SSP v1.7.0

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/.