Renesas Synergy™ Platform

NetX™ and NetX Duo™ Auto IP Module Guide

Introduction

This module guide will enable you to effectively use a module with 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 advanced uses of the module are available on the Renesas Synergy Knowledge Base (see the References section at end of the document), it is a valuable resources for creating more complex designs.

The Auto IP Protocol is designed to dynamically configure IPv4 addresses on a local network without requiring a server; unlike the Dynamic Host Configuration Protocol (DHCP). Auto IP uses address resolution protocol (ARP) for automatic IP address assignment and allocates addresses in the range of 169.254.1.0 through 169.254.254.255.

Note: Except for internal processing, the NetX Duo™ Auto IP is identical in the application, set-up, and running of an Auto IP session as the NetX™ Auto IP.

Contents

1. NetX and NetX Duo Auto IP Module Features

- Compliant with RFC3927 and related RFCs
- Uses ARP probes to check for address conflicts
- Uses the collision handler notification in NetX to detect an address already in use
- Registers a valid Auto IP address with the IP instance
- Provides high-level APIs for:
  - Creating and deleting an Auto IP instance
  - Starting and stopping the Auto IP thread task
  - Specifying the network interface on which to run Auto IP

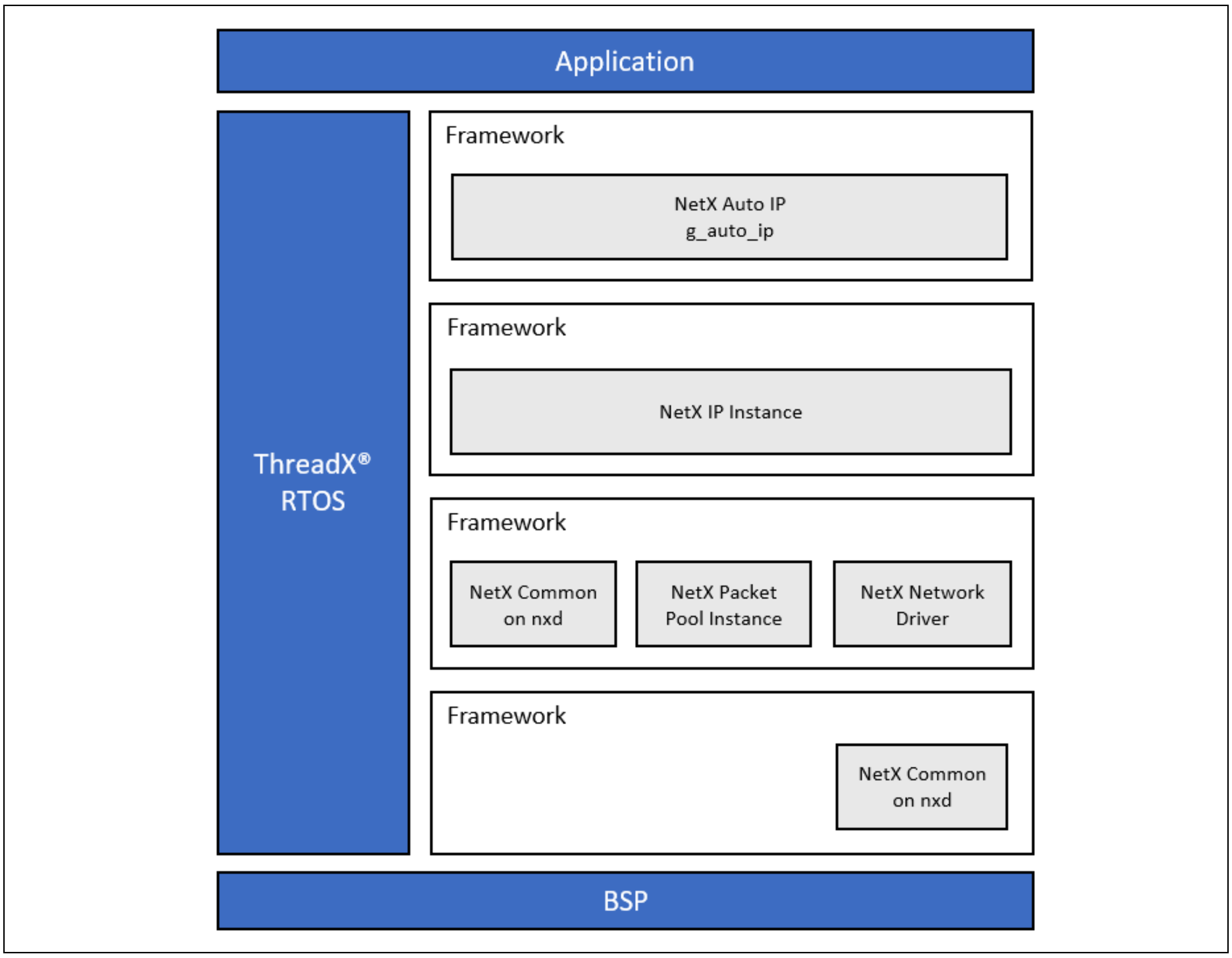


Figure 1. NetX and NetX Duo Auto IP Module Block Diagram

2. NetX and NetX Duo Auto IP Module APIs Overview

The NetX Auto IP defines APIs for creating, deleting, getting, and setting addresses. 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 and NetX Duo Auto IP Module API Summary

Function Name	Example API Call and Description
nx_auto_ip_create	`nx_auto_ip_create(&g_auto_ip0, "AutoIP 0", &g_ip0, stack_pointer, stack_size, priority);` Create an Auto IP instance

Function Name	Example API Call and Description
nx_auto_ip_delete	`nx_auto_ip_delete(&g_auto_ip_0);` Delete Auto IP instance.
nx_auto_ip_get_address	`nx_auto_ip_get_address(&g_auto_ip_0, &local_address);` Get current Auto IP address.
nx_auto_ip_set_interface	`nx_auto_ip_set_interface(&g_auto_ip_0, interface_index);` Set network interface needing an Auto IP address.
nx_auto_ip_start	`nx_auto_ip_start(&g_auto_ip_0, IP_ADDRESS(0,0,0,0));` Start Auto IP processing. If the address input is NULL. NetX Auto IP randomly assigns an address in the Auto IP address range.
nx_auto_ip_stop	`nx_auto_ip_stop(&g_auto_ip_0);` Stop Auto IP processing.
nx_dhcp_server_stop	`nx_dhcp_server_stop(&dhcp_server);` Stop DHCP server processing.

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

Table 2. Status Return Values

Name	Description
NX_SUCCESS	Successful AutoIP function
NX_AUTO_IP_ERROR	Error creating components of Auto IP instance
NX_PTR_ERROR*	Invalid pointer input
NX_CALLER_ERROR*	Invalid caller of this service
NX_AUTO_IP_NO_LOCAL	No Auto IP address registered with the NetX IP instance.
NX_AUTO_IP_BAD_INTERFACE_INDEX	Invalid network interface

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 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 and NetX Duo Auto IP Module Operational Overview

The NetX Auto IP protocol first selects a random address within the Auto IP IPv4 address range of 169.254.1.0 through 169.254.254.255. Alternatively, the application may force a starting IP address by providing it to the `nx_auto_ip_start` service; this is useful in situations where an Auto IP address has been used previously.

Once an auto IP address is selected, the NetX Auto IP sends out a series of ARP probes for the selected address. An ARP probe consists of an ARP request message with the sender address set to 0.0.0.0 and the target address set to the desired Auto IP address. A series of these ARP probes are sent (the actual number is set by the **ARP probes to send** property of the NetX Auto IP instance); if another network node responds to this probe or sends an identical probe for the same address, a new auto IP address is randomly selected within the auto IP IPv4 address range and the probe processing repeats.

If **ARP probes to send** and probes are sent without any responses, the NetX Auto IP issues many ARP announcements (set by the **Number of ARP announces** property) for the selected address. An ARP announcement consists of an ARP request message with both the sender and target address in the ARP message set to the selected auto IP address. If another network node responds to an announced message or sends an identical announcement for the same address, a new auto IP address is randomly selected within the auto IP IPv4 address range, and the probe processing starts over. When the probe, and the announcement, completes without any detected conflicts, the selected auto IP address is considered valid and the address is registered with the IP instance.

The NetX Auto IP registers the auto IP-generated IP address with the NetX IP instance successful probe and announcement processing. The Auto IP application can be notified of address changes using the `nx_ip_address_change_notify` callback in NetX, or it can use the `nx_ip_status_check` to determine when a valid IP address is assigned. Once a valid address is assigned, the application should stop the auto IP task using the `nx_auto_ip_stop` service. The address change callback notifies the application of address changes after the auto IP thread task is suspended. Possible reasons for an address changing without explicitly being done with an auto IP may be due to auto IP-address conflicts with other nodes, or a DHCP address resolution to replace the auto IP address.

3.1 NetX and NetX Duo Auto IP Module Important Operational Notes and Limitations

3.1.1 NetX and NetX Duo Auto IP Module Operational Notes

- The NetX DHCP Client and NetX Auto IP can both be used to ensure a host has a valid IP address. Typically, the DHCP Client attempts to contact a server. If none of the servers respond to the DHCP Client, the client is suspended, and the auto IP task is started. Auto IP generally guarantees a local address even if no DHCP Server is available. The DHCP Client can try later to broadcast requests to a DHCP Server; this process, if successful, automatically overwrites the auto IP local address.
- When the IP address changes, the application is responsible for closing out existing socket connections.

3.1.2 NetX and NetX Duo Auto IP Module Limitations

- If the NetX DHCP is used with the auto IP, the DHCP thread created must have a higher priority than the auto IP thread.
- The NetX Auto IP does not provide a mechanism to retain previously used IP address.

Refer to the latest *SSP Release Notes* for any additional operational limitations for this module.

4. Including the NetX and NetX Duo Auto IP Module in an Application

This section describes how to include either or both NetX and NetX Duo Auto IP module in an application using the SSP configurator.

Note: It is assumed 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 and NetX Duo Auto IP 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 Auto IP module is `g_auto_ip0`. This name can be changed in the associated **Properties** window.)

Table 3. NetX and NetX Duo Auto IP Module Selection Sequence

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

As shown in the following figure, when the NetX Auto IP module is added to the thread stack, the configurator automatically adds any needed lower-level modules. Any modules needing additional configuration information have the 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; they need only be added once and can be used by multiple stacks. Modules with a Pink band can require the selection of lower-level modules; these are either optional or recommended. (This is indicated in the block with the inclusion of this text.) If the addition of lower-level modules is required, the module description include **Add** in the text.

Clicking on any Pink banded modules brings up the **New** icon and displays possible choices.

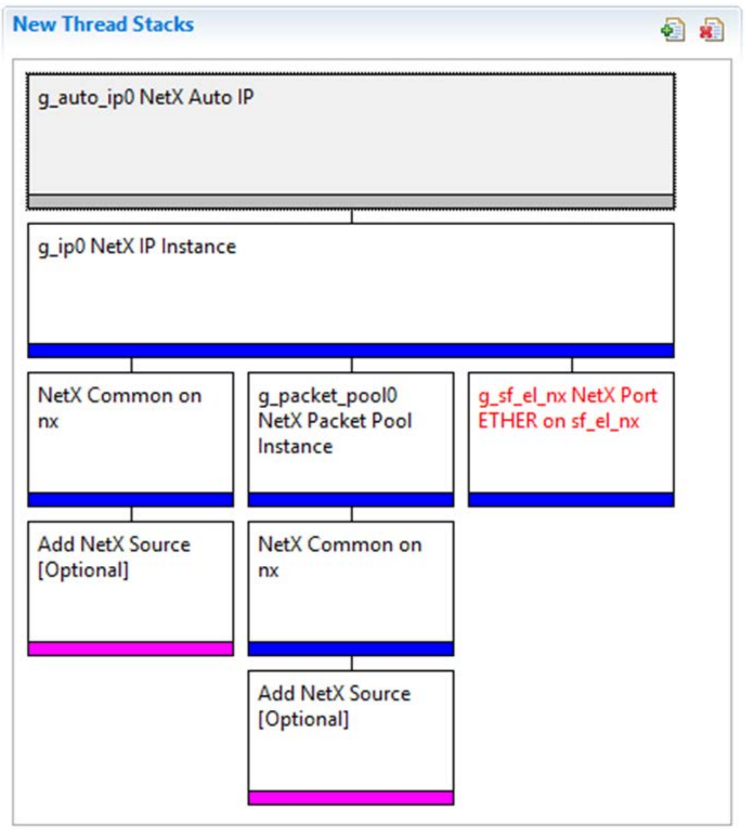

New Thread Stacks
g_auto_ip0 NetX Auto IP
g_ip0 NetX IP Instance
NetX Common on nx
g_packet_pool0 NetX Packet Pool Instance
g_sf_el_nx NetX Port ETHER on sf_el_nx
Add NetX Source [Optional]
NetX Common on nx
Add NetX Source [Optional]

Figure 2. NetX and NetX Duo Auto IP Module Stack

5. Configuring the NetX and NetX Duo Auto IP Module

The NetX Auto IP module must be configured by the user for the desired operation. The SSP configuration window automatically identifies (by highlighting the block in red) any required configuration selections, such as interrupts or operating modes, which must be configured for lower-level modules for successful operation. Only 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 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 indicates the validity of the setting based on the targeted MCU (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 looking over the following configuration table values. This helps to orient you and can be a useful **hands-on** approach to learning the ins and outs of developing with SSP.

Table 4. Configuration Settings for the NetX and NetX Duo Auto IP Module

Parameter	Value	Description
Wait before sending first probe (seconds)	1	Wait before sending first probe selection

Parameter	Value	Description
ARP probes to send	3	ARP probes to send selection
Minimum wait between probes (seconds)	1	Minimum wait between probes selection. (The Auto IP recommends a random delay between probes.)
Maximum wait between probes (seconds)	2	Maximum wait between probes selection
Maximum conflicts before increasing processing delay	10	Maximum conflicts before increasing processing delay before Auto IP tries another local IP address.
Wait extend after maximum conflicts (seconds)	60	Interval that Auto IP waits before trying another local address after the maximum conflicts is reached.
Wait before announcement (seconds)	2	Wait before announcement if the local IP address being probed results in zero conflicts.
Number of ARP announces	2	Number of ARP announcements sent selection
Wait between announces (seconds)	2	Wait between announces selection
Wait between defense announces (seconds)	10	Wait between defense announces selection
Name	g_auto_ip0	Module name
Internal thread stack size (bytes)	2048	Internal thread stack size selection
Internal thread priority	3	Internal thread priority 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 stack modules can be desirable. For example, it might be useful to select different addresses for the Ethernet port. 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 and NetX Duo Auto IP Lower-Level Modules

Only a small number of settings must be modified from the default for the IP layer and lower-level drivers 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 are locked to prevent user modification. The following table identifies all the settings within the properties section for the module.

Table 5. Configuration Settings for the NetX 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
**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. When All zeros use the 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	520	ARP Cache Size in Bytes selection

ISDE Property	Value	Description
Reverse ARP	Enable, Disable Default: Disable	Reverse ARP selection
TCP	Enable, Disable Default: Enable	TCP selection
UDP	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 that are only available in NetX Duo.

Table 6. Configuration Settings for the NetX and NetX Duo Common Instance

ISDE Property		Value	Description
No configurable parameters			

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 and NetX Duo 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 NetX Port ETHER

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

ISDE Property	Value	Description
	ThreadX, CM0+: invalid) Default: Disabled	
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.

5.2 NetX and NetX Duo Auto IP Module Clock Configuration

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

5.3 NetX and NetX Duo Auto IP Module Pin Configuration

The ETHERC peripheral module uses pins on the MCU device to communicate to external devices. I/O pins must be selected and configured by the external device as required. 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^{2}C pins.

Note: The selected operation mode determines the peripheral signals available and the MCU pins required.

Table 9. 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 10. 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: Example settings 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 and NetX Duo Auto IP Module in an Application

In a typical application, it is assumed that an IP instance has been created and an ARP is enabled. Once this IP instance is accomplished, the typical steps in using the NetX Auto IP in an application are:

1. Allow time for the IP thread task and the network driver to get initialized (2-3 seconds) using the `tx_thread_sleep()` API
2. Set the address change notification with the `nx_ip_address_change_notify` API [Optional]
3. Start the Auto IP instance with the `nx_auto_ip_start` API.

4. Check for a valid address for the IP instance using either the `nx_ip_status_check` or `nx_auto_ip_get_address` API.
   The `nx_ip_status_check` API defaults to the primary address. If running Auto IP on a secondary interface, use the `nx_ip_interface_status_check`. Note that nx_auto_ip_get_address API works for Auto IP on either primary or secondary addresses.
5. If a valid local IP address is assigned, stop the auto IP thread task using the `nx_auto_ip_stop` API.

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

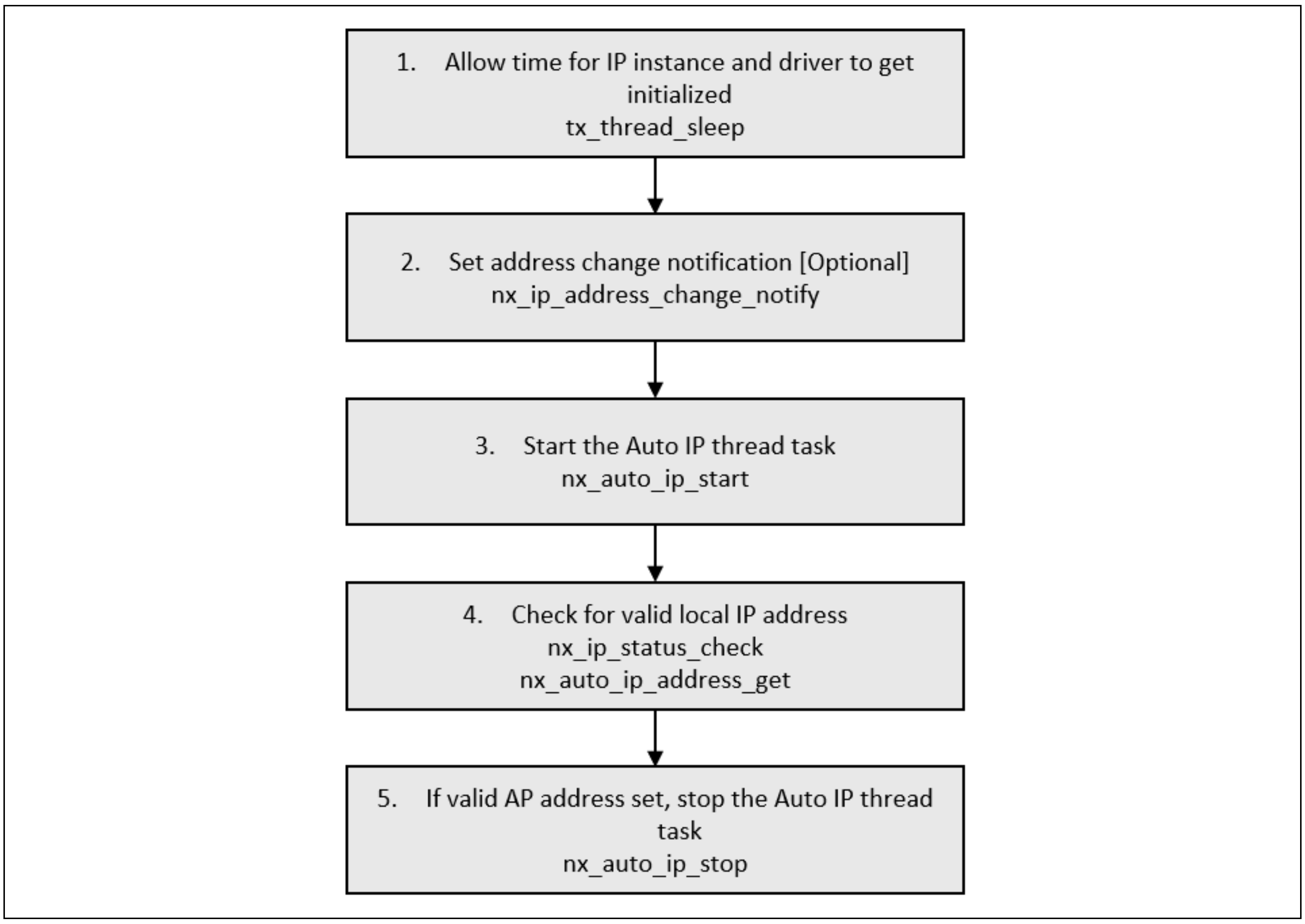

1. Allow time for IP instance and driver to get initialized
tx_thread_sleep
2. Set address change notification [Optional]
nx_ip_address_change_notify
3. Start the Auto IP thread task
nx_auto_ip_start
4. Check for valid local IP address
nx_ip_status_check
nx_auto_ip_address_get
5. If valid AP address set, stop the Auto IP thread task
nx_auto_ip_stop

Figure 3. Flow Diagram of a Typical NetX and NetX Duo Auto IP Module Application

7. The NetX and NetX Duo Auto IP Module Application Project

The application project associated with this module guide demonstrates the steps needed 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 and NetX Duo Auto IP module. You can also read over the code in `autoip_thread_entry.c` which illustrates the NetX and NetX Duo Auto IP module APIs in a complete design.

The application project demonstrates the typical use of the NetX and NetX Duo Auto IP module APIs. The auto IP protocol is responsible for selecting a valid IP address; this is a helpful method in non-DHCP cases, for example. Appropriate NetX functions are used to monitor the network connection.

The following table identifies the target versions for the associated software and hardware used by the Application Project:

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

Resource	Revision	Description
e^2 studio	7.3.0 or later	Integrated Solution Development Environment

Resource	Revision	Description
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 v3.3	Starter Kit

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

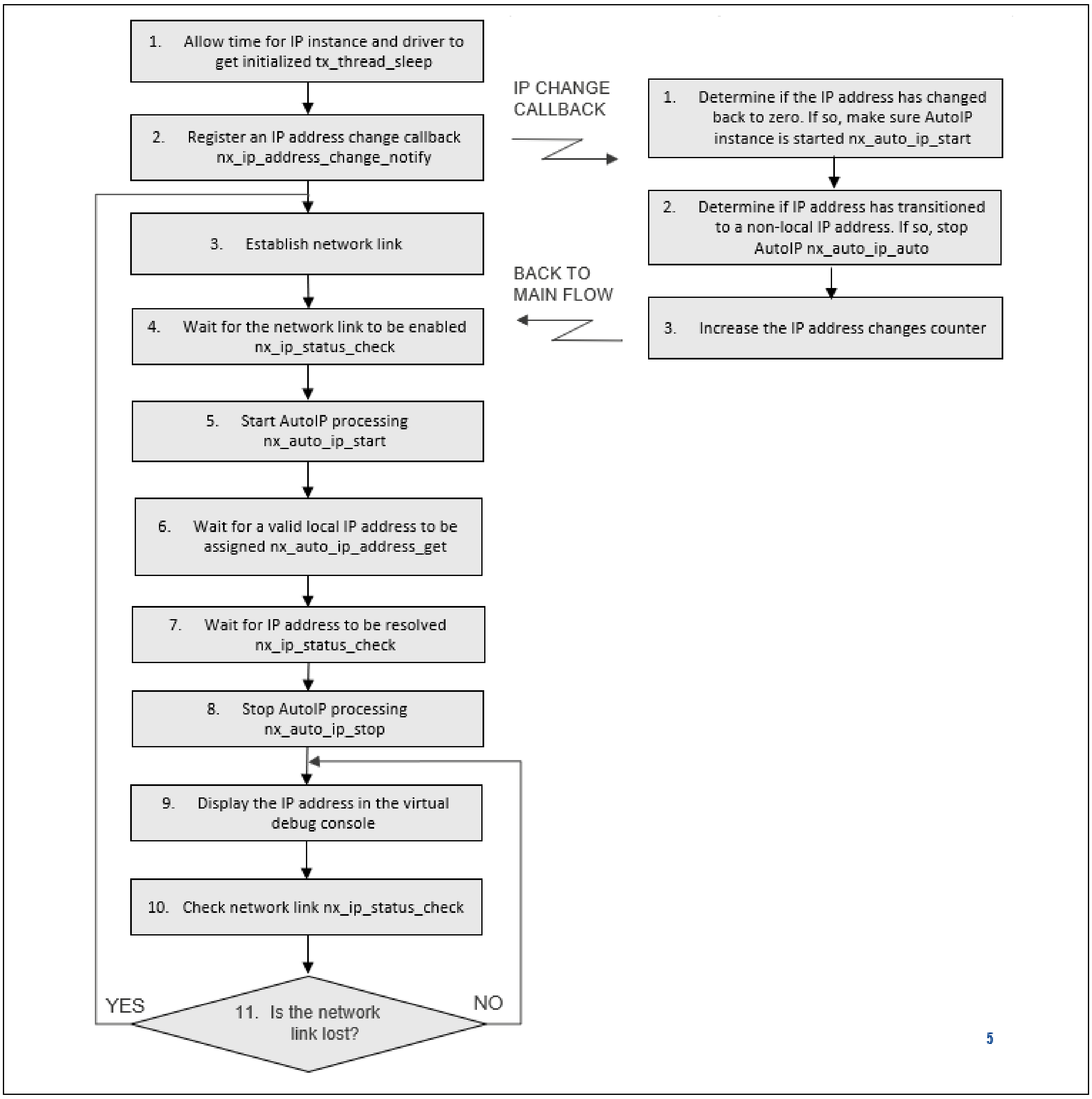

1. Allow time for IP instance and driver to get initialized tx_thread_sleep
2. Register an IP address change callback nx_ip_address_change_notify
3. Establish network link
4. Wait for the network link to be enabled nx_ip_status_check
5. Start AutoIP processing nx_auto_ip_start
6. Wait for a valid local IP address to be assigned nx_auto_ip_address_get
7. Wait for IP address to be resolved nx_ip_status_check
8. Stop AutoIP processing nx_auto_ip_stop
9. Display the IP address in the virtual debug console
10. Check network link nx_ip_status_check
11. Is the network link lost?
YES
NO
IP CHANGE CALLBACK
1. Determine if the IP address has changed back to zero. If so, make sure AutoIP instance is started nx_auto_ip_start
2. Determine if IP address has transitioned to a non-local IP address. If so, stop AutoIP nx_auto_ip_auto
3. Increase the IP address changes counter
BACK TO MAIN FLOW
5

Figure 4. NetX and NetX Duo Auto IP Module Application Project Flow Diagram

The `autoip_thread_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 follow along with the description provided to help identify key uses of APIs.

The first section of the `autoip_thread_entry.c` has the header files which reference the NetX and NetX Duo Auto IP instance and integer types. The next section contains macro definitions, that created to improve code readability and included for your convenience. Next, a variable is defined; it holds the number of IP address changes.

The second section of the file contains prototypes of functions defined and called in the third section of the same file.

The third section of the file contains definitions of functions. It starts with an auxiliary function `ip_octet`, which extracts a selected octet from an IP address. It is helpful for printing an IP address (stored in a single variable) in readable 4-octet format. Next is a function `ip_address_changed`; it is a callback function called every time the IP address has changed. It might be used to detect the IP address change occurrence and to perform appropriate action for such an event. Here it performs the start of an auto IP protocol and the `g_address_changes` variable incrementation. The next function, `establish_network_link`, is responsible for establishing the network link. The `nx_ip_status_check` function is then used to check if a network link is enabled. If so, an auto IP protocol is started by calling the `nx_auto_ip_start` function. An auto IP protocol tries to assign a valid IP address selected from a range of 169.254.1.0 through 169.254.254.255. Function `nx_auto_ip_get_address` checks whether an assigned IP address is valid. The function `nx_ip_status_check` is then used again; this time, however, it checks whether an IP address has been resolved. If so, the network link has been established correctly and an auto IP protocol may be stopped using the `nx_auto_ip_stop` function.

The last section of the `autoip_thread_entry.c` file contains the `autoip_thread_entry` function definition. This is the core part of the application project; functions described earlier are called from this function. At its beginning, there are definitions of necessary variables and a constant. The `tx_thread_sleep` function after that assures a delay for the IP thread task and the network driver initialization. Information about the LEDs available on the board is then read using the `R_BSP_LedsGet` function. An IP address change callback function is registered by the `nx_ip_address_change_notify` function. From now on, the `ip_address_changed` function is called every time an IP address has changed. The network link is established by the `establish_network_link` function. After that, there is an infinite loop. Inside of it, the LED is toggled. The currently established IP address is retrieved with the `nx_ip_address_get` function and displayed in the virtual debug console. There is a network link check performed. In case of a network-link loss, the application starts waiting for a network link to be established again by calling the `establish_network_link` function. The infinite loop repeats after a delay caused by the `tx_thread_sleep` function.

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

Table 12. NetX and NetX Duo Auto IP Module Configuration Settings for the Application Project

ISDE Property	Value Set
Wait before sending first probe (seconds)	1
ARP probes to send	3
Minimum wait between probes (seconds)	1
Maximum wait between probes (seconds)	2
Maximum conflicts before increasing processing delay	10
Wait extend after maximum conflicts (seconds)	60
Wait before announcement (seconds)	2
Number of ARP announces	2
Wait between announces (seconds)	2
Wait between defense announces (seconds)	10
Name	g_auto_ip0
Internal thread stack size (bytes)	2048
Internal thread priority	3

Table 13. NetX and NetX Duo IP Instance Configuration Settings for the Application Project

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
**IPv6 Global Address (use commas for separation)	0x2001, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x1
**IPv6 Link Local Address (use commas for separation, all zeros mean use MAC address)	0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0
IP Helper Thread Stack Size (bytes)	2048
IP Helper Thread Priority	3
ARP	Enable
ARP Cache Size in Bytes	520
Reverse ARP	Disable
TCP	Enable
UDP	Enable
ICMP	Enable

** Indicates properties that are only available in NetX Duo.

Table 14. NetX and NetX Duo Common Configuration Settings for the Application Project

ISDE Property	Value Set
No configurable parameters	

Table 15. NetX Packet Pool Instance Configuration Settings for the Application Project

ISDE Property	Value Set
Name	g_packet_pool0
Packet Size in Bytes	1500
Number of Packets in Pool	32

Table 16. NetX Port ETHER Configuration Settings for the Application Project

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	0x0A0076BA
Number of Receive Buffer Descriptors	8
Number of Transmit Buffer Descriptors	32
Ethernet Interrupt Priority	Priority 8 (CM4: valid, CM0+: invalid)
Name	g_sf_el_nx
Channel	1
Callback	NULL

8. Customizing the NetX and NetX Duo Auto IP Module for a Target Application

Some configuration settings can be changed by the developer from those shown in the application project. For example, you can easily change the MAC address of the SK-S7G2 Ethernet interface. To make the change, modify the properties, **Channel 1 MAC Address High Bits** and the **Channel 1 MAC Address Low Bits**, of the NetX Port ETHER configuration settings. You can also change ARP behavior by modifying **NetX and NetX Duo Auto IP** module configuration settings. For example, the delay for sending the first ARP probe may be adjusted with the **Wait before sending the first probe (seconds)** property. The number of ARP probes to be sent can be altered with the **ARP probes to send** property.

9. Running the NetX and NetX Duo Auto IP Module Application Project

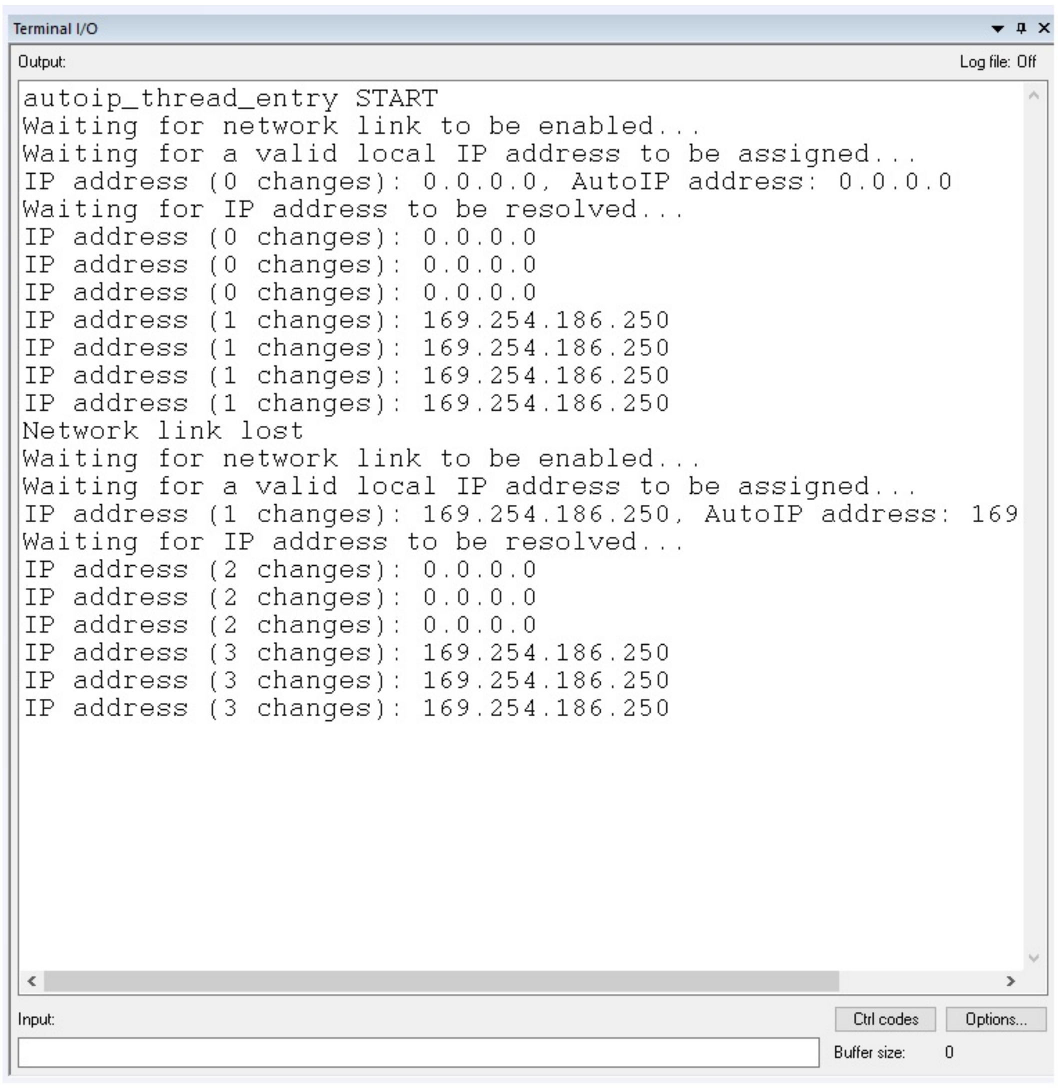
To run the NetX and NetX Duo Auto IP Module application project and to see it executed on a target kit, you can simply import it into your ISDE, compile, and run debug.

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

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 or the IAR EW for Synergy ISDE and building/running the application.
2. Connect to the host PC via a micro USB cable to J19 on SK-S7G2 Kit.
3. Start to debug the application. While in Debug mode in e^2 studio, click **Run** > **Resume** or click on the **Play** icon twice.
4. Connect to the host PC via an Ethernet cable to J11 on SK-S7G2 Synergy MCU Kit.
5. The SK-S7G2 Synergy MCU Kit gets a valid IP address.

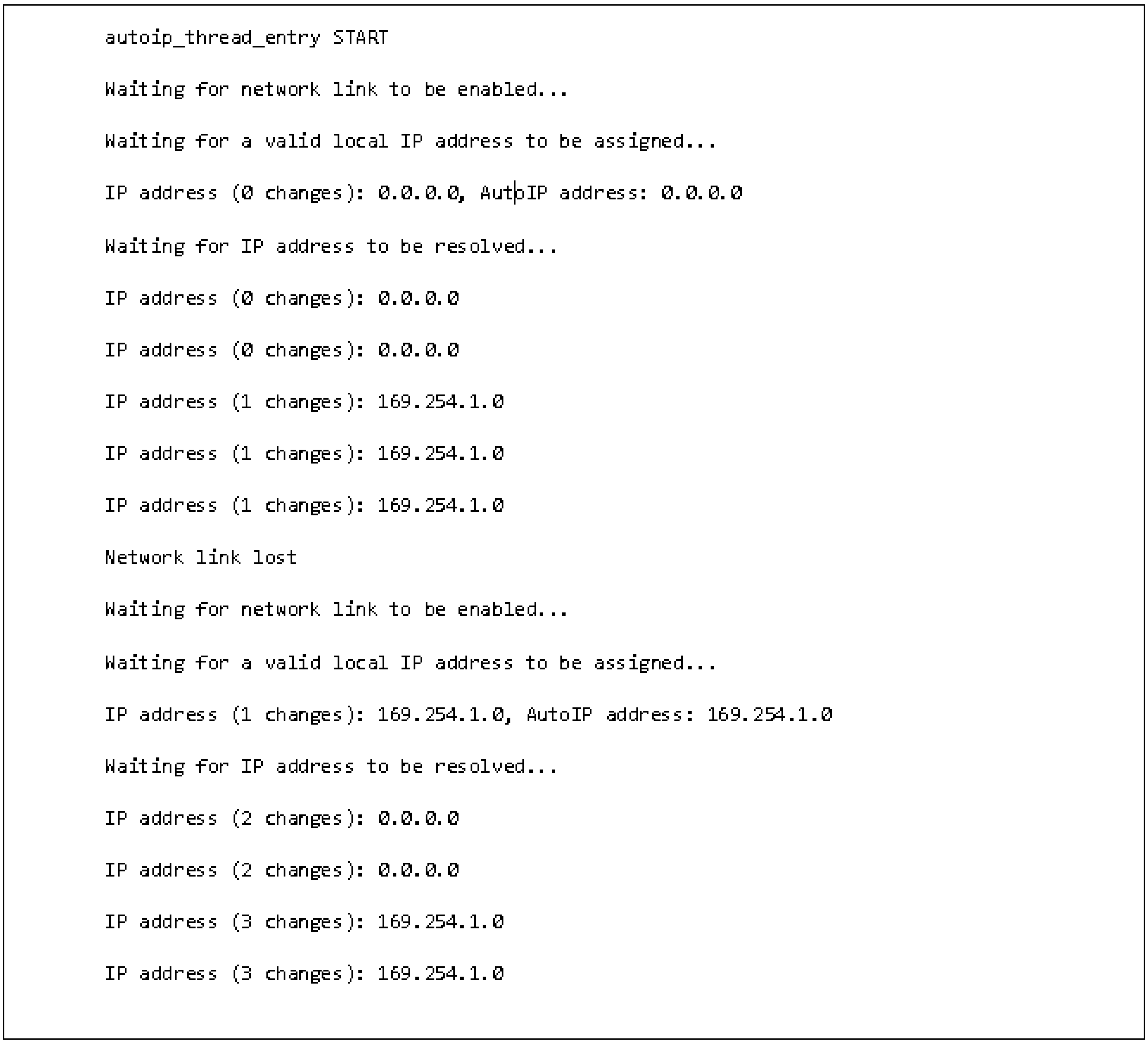
   Its output text messages can be viewed in the virtual debug console(In case of e2 Studio, Renesas Debug Virtual Console. In case of IAR , Terminal I/O)
6. Disconnect Ethernet cable from J11 on SK-S7G2 Kit for a few seconds, then connect the cable again.
7. The SK-S7G2 Kit gets a valid IP address.

   Its output text messages can be viewed in the virtual debug console as shown in the Figure 5 and 6.



```
autoip_thread_entry START
Waiting for network link to be enabled...
Waiting for a valid local IP address to be assigned...
IP address (0 changes): 0.0.0.0, AutoIP address: 0.0.0.0
Waiting for IP address to be resolved...
IP address (0 changes): 0.0.0.0
IP address (0 changes): 0.0.0.0
IP address (0 changes): 0.0.0.0
IP address (1 changes): 169.254.186.250
IP address (1 changes): 169.254.186.250
IP address (1 changes): 169.254.186.250
IP address (1 changes): 169.254.186.250
Network link lost
Waiting for network link to be enabled...
Waiting for a valid local IP address to be assigned...
IP address (1 changes): 169.254.186.250, AutoIP address: 169
Waiting for IP address to be resolved...
IP address (2 changes): 0.0.0.0
IP address (2 changes): 0.0.0.0
IP address (2 changes): 0.0.0.0
IP address (3 changes): 169.254.186.250
IP address (3 changes): 169.254.186.250
IP address (3 changes): 169.254.186.250
```

Figure 5. Example Output from NetX and NetX Duo Auto IP Module Application Project on IAR



```
autoip_thread_entry START

Waiting for network link to be enabled...

Waiting for a valid local IP address to be assigned...

IP address (0 changes): 0.0.0.0, AutoIP address: 0.0.0.0

Waiting for IP address to be resolved...

IP address (0 changes): 0.0.0.0

IP address (0 changes): 0.0.0.0

IP address (1 changes): 169.254.1.0

IP address (1 changes): 169.254.1.0

IP address (1 changes): 169.254.1.0

Network link lost

Waiting for network link to be enabled...

Waiting for a valid local IP address to be assigned...

IP address (1 changes): 169.254.1.0, AutoIP address: 169.254.1.0

Waiting for IP address to be resolved...

IP address (2 changes): 0.0.0.0

IP address (2 changes): 0.0.0.0

IP address (3 changes): 169.254.1.0

IP address (3 changes): 169.254.1.0
```

Figure 6. Example Output from NetX and NetX Duo Auto IP Module Application Project on E2 Studio

10. NetX and NetX Duo Auto IP Module Conclusion

This *Module Guide* has provided all the background information needed to select, add, configure, and use the module in an example project. Many of these steps were time consuming and error-prone activities in previous generations of embedded systems. The Renesas Synergy™ Platform makes these steps much less time consuming and removes the common errors like conflicting configuration settings or incorrect selection of 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 and NetX Duo Auto IP Module Next Steps

After you have mastered a simple NetX and NetX Duo Auto IP module project, you may want to review a more complex example. Because auto IP is only one of the methods which can configure the IP address, it is advisable to try out the NetX DHCP module as well as other networking modules available in the NetX and NetX Duo Interfaces. *User's Guides* for these modules can be easily found using the instructions provided in the References section at the end of this document.

12. NetX and NetX Duo Auto IP Module Reference Information

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

Links to all the most up-to-date NetX and NetX Duo Auto IP module reference materials and resources are available on the Synergy Knowledge Base: https://en-support.renesas.com/knowledgeBase/16977449.

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	Nov.01.17	—	Initial release
1.01	Jan.08.19	—	Updated configuration tables and settings.
1.02	May.02.19	—	Updated configuration file for SSP 1.6.0.

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/.