Renesas Synergy™ Platform

# NetX Duo TLS Module Guide

R11AN0341EU0100
Rev.1.00
Sep 29, 2018

## Introduction

The NetX Duo TLS 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 NetX Duo TLS module is based on ThreadX “NetX Duo Secure” which implements the Transport Layer Security (TLS) protocol as described in RFCs 2246 (version 1.0), 4346 (version 1.1), and 5246 (version 1.2). NetX Duo Secure also includes routines for basic X.509 (RFC 5280). NetX Duo Secure is intended for applications using the ThreadX RTOS.

## Features

The TLS/SSL protocol provides privacy and reliability between two communicating applications. It has the following basic properties:

- Encryption: The messages exchanged between communicating applications are encrypted to ensure that the connection is private. Symmetric cryptography mechanism such as AES (Advanced Encryption Standard) is used for data encryption.
- Authentication: A mechanism to check the peer’s identity using digital certificates or some other means to assure that the peer is the intended target to prevent fraud.
- Integrity: A mechanism to detect message tampering and forgery to ensure that connection is reliable. Message Authentication Codes (MACs) created using algorithms like the Secure Hash Algorithm (SHA) are used to ensure message integrity

## Contents

## 1. NetX Duo TLS Module Features

- Support for RFC 2246 The TLS Protocol Version 1.0
- Support for RFC 4346 The Transport Layer Security Protocol Version 1.1
- Support for RFC 5246 The Transport Layer Security (TLS) Protocol Version 1.2
- Support for RFC 5280 X.509 PKI Certificates (v3)
- Support for RFC 3268 Advanced Encryption Standard (AES) Cipher suites for Transport Layer Security (TLS)
- RFC 3447 Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1
- RFC 2104 HMAC: Keyed-Hashing for Message Authentication
- RFC 6234 US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)
- RFC 4279 Pre-Shared Key Cipher suites for TLS

Notes: 1. Please refer to *Express Logic NetX Duo Secure User Guide* for more and complete details.
2. For known issues and limitations of NetX Duo Secure module, please refer to the *SSP Release Notes*.

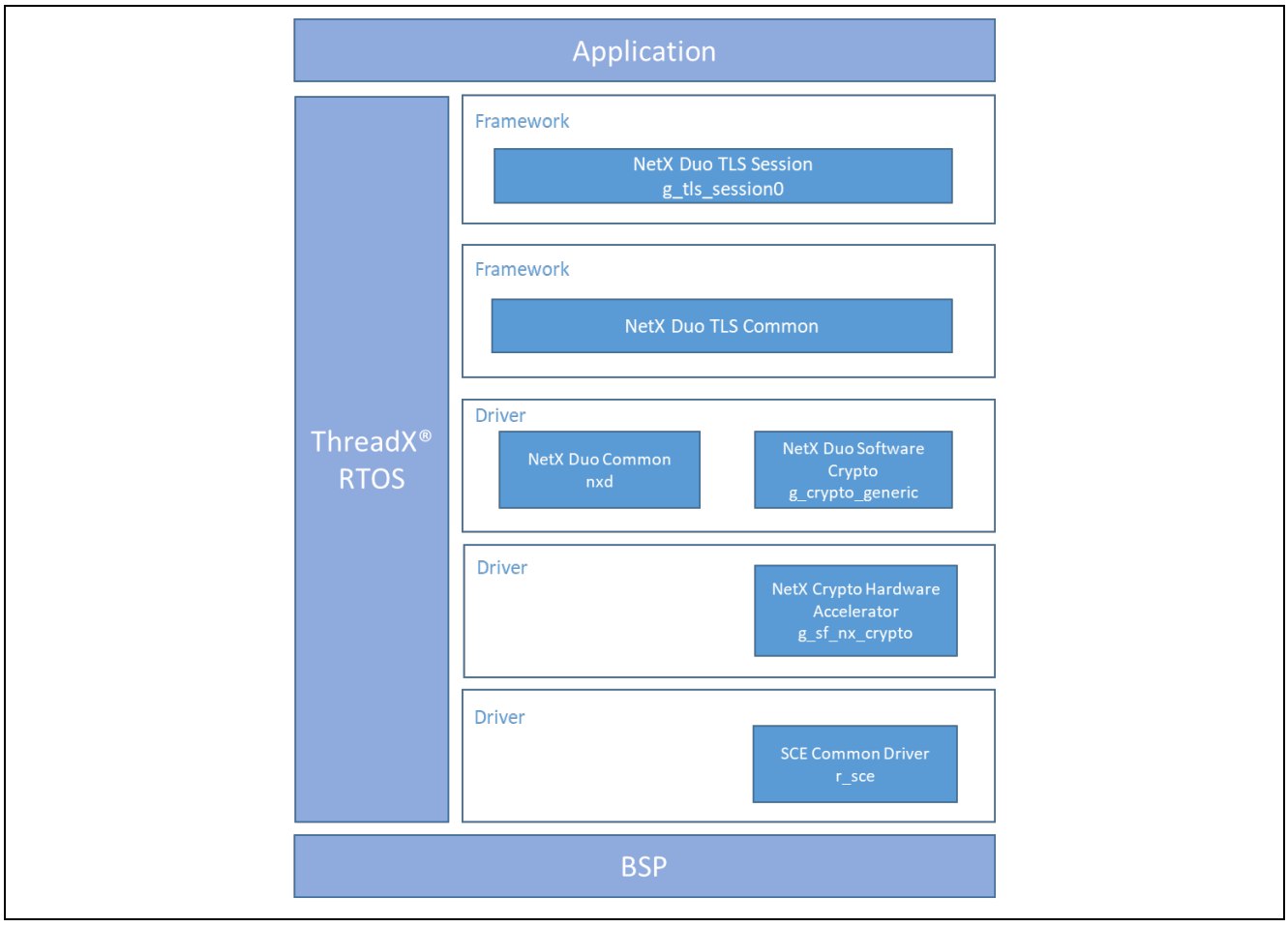


**Figure 1 NetX Duo TLS Module Block Diagram**

## 2. NetX Duo TLS Module APIs Overview

The NetX Duo TLS Support module defines APIs for creating the creating and setting up a TLS security session. A complete list of the available APIs, an example API call, and a short description of each can be found in the table below. A table of status return values follows.

**Table 1 Status Return Values**

| Function Name | Example API Call and Description |
|---|---|
| nx_secure_tls_local_ certificate_add | nx_secure_tls_local_certificate_add (tls_session, certificate);<br><br>Adds an initialized certificate to a TLS session for use as a local identification certificate - the TLS Server certificate for TLS servers, and the Client certificate for TLS clients. |
| nx_secure_tls_local_ certificate_remove | nx_secure_tls_local_certificate_remove(tls_session, common_name, common_name_length);<br><br>Removes a certificate instance from the local certificates list, keyed on the Common Name field. |
| nx_secure_tls_metadata_ size_calculate | nx_secure_tls_metadata_size_calculate(cipher_table, metadata_size);<br><br>Determines the size of the buffer needed by TLS for encryption metadata for a given ciphersuite table |
| nx_secure_tls_packet_ allocate | nx_secure_tls_packet_allocate(tls_session, pool_ptr, packet_ptr, wait_option);<br><br>Allocates a packet for a TLS application such that it allows additional room for the TLS header |
| nx_secure_tls_remote_ certificate_allocate | nx_secure_tls_remote_certificate_allocate( tls_session, certificate, raw_certificate_buffer, buffer_size);<br><br>Adds an uninitialized certificate instance to a TLS session for the purpose of allocating space for certificates provided by a remote host during a TLS session |
| nx_secure_tls_session_ certificate_callback_set | nx_secure_tls_session_certificate_callback_set( tls_session, session);<br><br>Sets up a function pointer that TLS will invoke when a certificate is received from a remote host, allowing the application to perform validation checks such as certificate revocation and certificate policy enforcement |
| nx_secure_tls_session_ create | nx_secure_tls_session_create(session_ptr, cipher_table, metadata_area, metadata_size);<br><br>Initializes a TLS session control block for later use in establishing a secure TLS session over a TCP socket or other lower-level networking protocol |
| nx_secure_tls_session_ client_verify_disable | nx_secure_tls_session_client_verify_ disable( tls_session);<br><br>Disables Client Certificate Verification for a particular TLS Session which previously had it enabled. |
| nx_secure_tls_session_ client_verify_enable | nx_secure_tls_session_client_verify_enable( tls_session);<br><br>Enables Client Certificate Verification for TLS Server instances. If enabled, the TLS Server will request and verify a remote TLS Client Certificate using all available crypto signature routines. |

| | |
|---|---|
| nx_secure_tls_session_delete | nx_secure_tls_session_delete(tls_session);<br><br>Deletes a TLS session object, returning any resources to the system |
| nx_secure_tls_session_end | nx_secure_tls_session_end(tls_session, wait_option);<br><br>Ends an active TLS session by sending the TLS CloseNotify alert to the remote host, then waiting for the response CloseNotify before returning. |
| nx_secure_tls_session_packet_buffer_set | nx_secure_tls_session_packet_buffer_set( session_ptr, buffer_ptr, buffer_size);<br><br>Sets the buffer TLS uses to reassemble incoming messages which may span multiple TCP packets. |
| nx_secure_tls_session_protocol_oversion_override | nx_secure_tls_session_protocol_version_override(tls_session, protocol_version);<br><br>Overrides the TLS protocol version to use for the TLS session. This allows for a different version of TLS to be utilized even if a newer version is enabled. |
| nx_secure_tls_session_receive | nx_secure_tls_session_receive(tls_session, packet_ptr_ptr, wait_option);<br><br>Receives data from an active TLS session, handling all decryption and verification before returning the data to the caller in the supplied NX_PACKET structure |
| nx_secure_tls_session_reset | nx_secure_tls_session_reset(session_ptr);<br><br>Resets a TLS session object, clearing out all data for initialization or re-use. |
| nx_secure_tls_session_send | nx_secure_tls_session_send(tls_session, packet_ptr, wait_option);<br><br>Sends data using an active TLS session, handling all encryption and hashing before sending data over the established TCP socket connection |
| nx_secure_tls_session_start | nx_secure_tls_session_start(tls_session, tcp_socket, wait_option);<br><br>Starts a TLS session given a TCP socket. The TCP connection must be established before calling this function or the TLS handshake will fail. |
| nx_secure_tls_session_time_function_set | nx_secure_tls_session_time_function_set(tls_session, time_func_ptr)<br><br>Sets up a function pointer that TLS will invoke when it needs to get the current time, which is used in various TLS handshake messages and for verification of certificates. |
| nx_secure_tls_trusted_certificate_add | nx_secure_tls_trusted_certificate_add( tls_session, certificate);<br><br>Adds an initialized certificate to a TLS session for use as a trusted Root Certificate |

| | |
|---|---|
| nx_secure_tls_trusted_certificate_remove | nx_secure_tls_trusted_certificate_remove( tls_session, common_name, common_name_length);<br><br>Removes a certificate instance from the trusted certificates store, keyed on the Common Name field. |
| nx_secure_tls_remote_certificate_free_all nx_secure_tls_remote_certificate_free_all | Release certificates previously registered with the TLS session. |
| **nx_secure_tls_psk_add | nx_secure_tls_psk_add(tls_session, pre_shared_key, psk_length, psk_identity, identity_length, hint, hint_length);<br><br>Adds a pre-shared key (PSK) to a TLS session for use with a PSK ciphersuite. The second parameter is the PSK identity used during the TLS handshake to select the proper key. |
| **nx_secure_tls_psk_find | nx_secure_tls_psk_find(tls_session, psk_data, psk_length,psk_identity, identity_length);<br><br>Finds a pre-shared key (PSK) in a TLS session for use with a PSK ciphersuite. The PSK is found using an "identity hint" that should match a field in the PSK structure in the TLS session. |
| **nx_secure_tls_client_psk_set | nx_secure_tls_client_psk_set(tls_session, pre_shared_key, psk_length, psk_identity, identity_length, hint, hint_length);<br><br>Sets the pre-shared key (PSK) for a TLS Client in a TLS session control block for use with a remote server that is sing a PSK ciphersuite. |
| nx_secure_x509_certificate_initialize | Initialize X.509 Certificate for NetX Secure TLS |
| nx_secure_x509_common_name_dns_check | nx_secure_x509_common_name_dns_check(&certificate, dns_tld, dns_tld_length)<br><br>Check DNS name against X.509 CertificateCheck DNS name against X.509 Certificate |
| nx_secure_x509_crl_revocation_check | nx_secure_x509_crl_revocation_check(&crl_data, crl_length, &cert_store, &certificate)<br><br>Check X.509 Certificate against a supplied Certificate Revocation List |

Note: ** Requires that the property PSK Cipher Suite of the TLS Common component be enabled.

## 3. NetX Duo TLS Module Operational Overview

TLS uses TCP and provides secure communications for application layer protocols such as HTTP and MQTT. TLS can also be used in ‘bare’ socket applications to send and receive data in a secure session to another peer. The module guide for this project uses this simplified application of TLS to demonstrate a TLS Client and TLS Server sockets exchanging data to a peer.

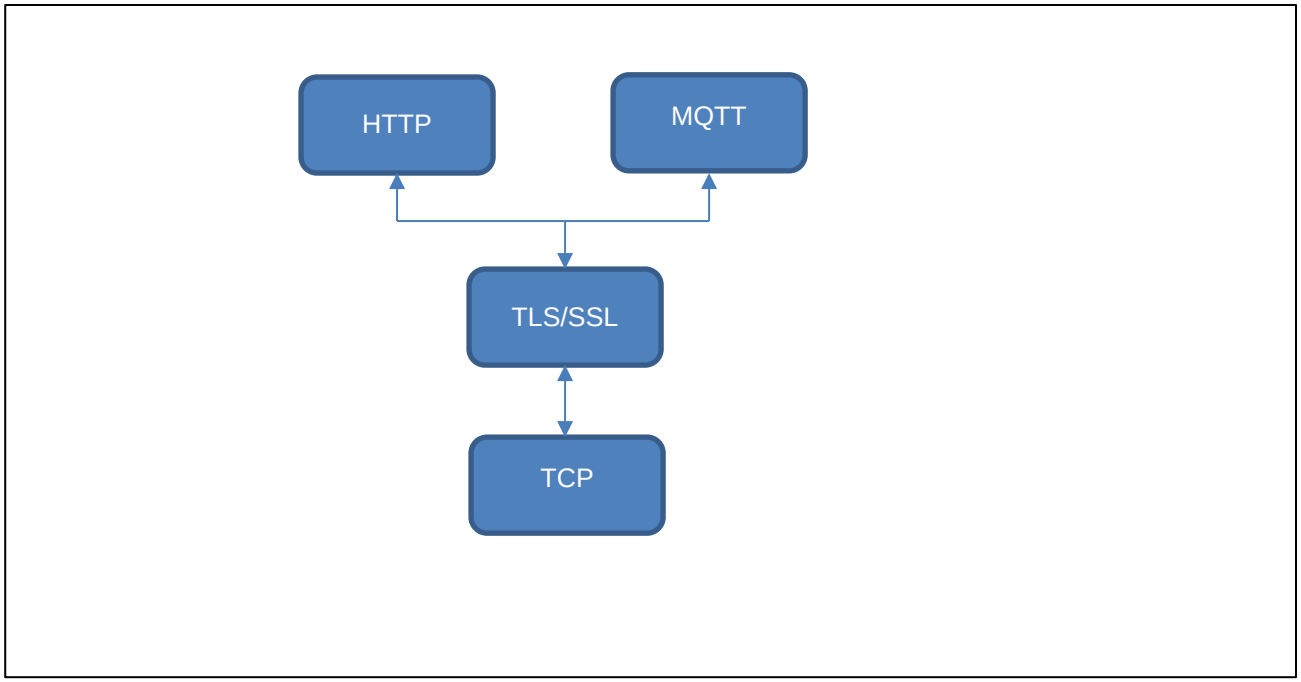


**Figure 2 TLS/SSL Layering**

TLS does not have a well-known port number. Instead it uses the designated port number of the secure variant of the higher layer protocol. For example, port number 443 for secure HTTP, port number 8883 for MQTT etc.

When a secure connection is established using TLS/SSL, for example using HTTPS, messages are exchanged between the client (which always initiates the connection) and a server. The first set of messages execute the TLS Handshake Protocol which sets up the secure TLS session. After the TLS handshake completes the client and server can securely send/receive data bidirectionally as shown in the following figure.

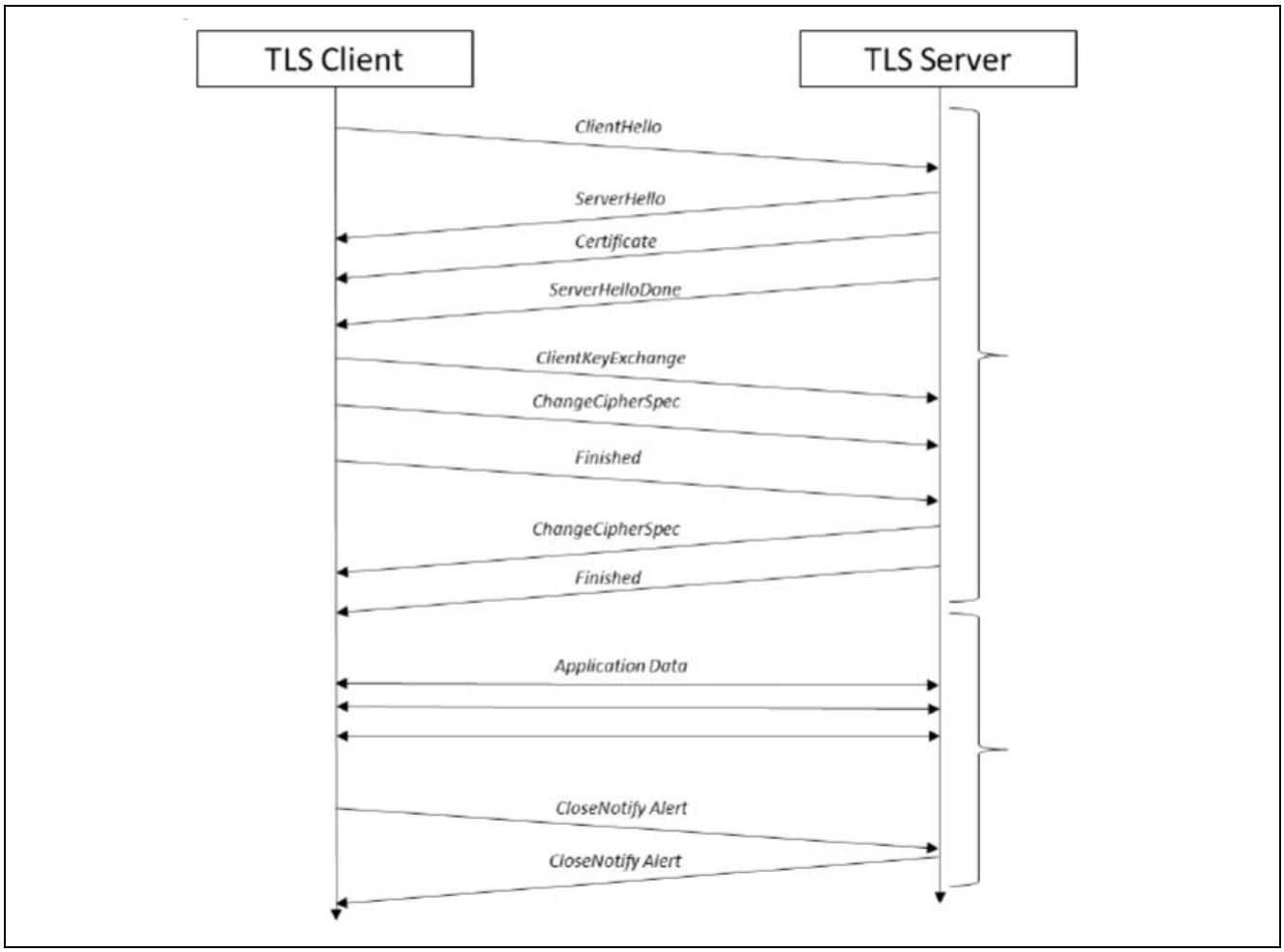


**Figure 3 TLS Protocol Sequence**

## 3.1 NetX Duo TLS Module Important Operational Notes and Limitations

### 3.1.1 NetX Duo TLS Module Operational Notes

- Before the TLS session is created, the cryptographic metadata buffer must be allocated. The size of the metadata buffer can be set using the properties pane. Also, the user can use NetX Duo Secure `nx_secure_metadata_size_calculate` API to calculate the required size of metadata buffer. The metadata size is specified in the "Meta data size" of the TLS session component, or in the `nx_secure_tls_session_create` call. The size of this buffer is dependent upon the cryptographic routines being used and **must** meet or exceed the size calculated by `nx_secure_metadata_size_calculate`.
- A packet reassembly buffer is also required for TLS sessions. To associate a packet buffer with a TLS session, use the `nx_secure_tls_session_packet_buffer_set` API. The reassembly buffer is used to store incoming TLS records which may span multiple TCP packets. If an incoming TLS record is larger than the supplied buffer, the TLS session will end with an error. A reasonable packet buffer size is 6-8 kbytes but note that TLS specifies that records may be as large as 16 kbytes.
- Also, before starting a TLS Client session the application must allocate memory for processing incoming server certificate data using the `nx_secure_tls_remote_certificate_allocate` call. A reasonable buffer size for most certificates is 2 kbytes and the TLS Client application should allow for 2-3 certificates from most servers. The `nx_secure_tls_remote_certificate_allocate` call should be called once for each expected server certificate with a new NX_SECURE_X509_CERT structure and buffer each time.
- For any incoming certificate, the NetX Duo Secure TLS will perform basic X.509 path validation. If there are no trusted certificates on the TLS Client, the TLS session with end with an error. A "trusted" certificate is added using the API `nx_secure_tls_trusted_certificate_add`.
- Before starting an TLS Server session, an identity certificate must be added using the `nx_secure_tls_local_certificate_add` API. The certificate must be initialized using the API nx_secure_x509_certificate_initialize.

- To verify an X.509 certificate, traverse a “certificate chain” from the received certificate being verified to a trusted certificate on the local device. During this traversal the expiration date of each certificate is checked against the time provided by an application callback. The `nx_secure_tls_session_time_function_set` API is used to optionally set up a function pointer for a function that TLS will invoke whenever it needs to get the current time. If a timestamp function is registered with the TLS session, a timestamp will be used in the generation of the Server or Client Hello, and in verifying the remote certificate. Note that omitting this callback function may cause interoperability problems and reduce the security of your TLS session.
- Before attempting to reconnect to the same or another TLS server, the TLS client must clear the TLS session. This is most easily done by calling `nx_secure_tls_session_end`, which should always be called to close down a TLS connection anyway. It is recommended to delete the TLS session and recreate it before making another connection attempt. For applications using SSP 1.3.x, the `nx_secure_tls_session_create` call should be preceded by a `memset` call on the TLS session to clear the session completely:
```
memset(tls_session_ptr, 0, sizeof(NXD_SECURE_TLS_SESSION));
```

### 3.1.2 NetX Duo TLS Module Limitations

- Due to the nature of embedded devices, some systems may not have adequate memory to support the maximum TLS record size of 16KB. NetX Duo TLS Secure can handle 16 KB records on devices with sufficient resources.
- NetX Duo Secure TLS performs basic X.509 certificate chain verification only. NetX Duo TLS Secure will perform basic X.509 chain verification on a certificate to assure that the certificate is valid and signed (directly or indirectly) by a trusted Certification Authority, and can provide the certificate Common Name for the application to compare against the Top-Level Domain Name of the remote host. However, verification of certificate extensions and other data is the responsibility of the application implementer. Refer to the Express Logic NetX Duo TLS Secure User Guide available from the Synergy Gallery for more details.
- Software-based cryptography is processor-intensive and not available. Therefore hardware-based cryptography is used for optimal performance of NetX Duo TLS Secure.

## 4. Including the NetX Duo TLS Module in an Application

This section describes how to include the NetX Duo TLS 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 Duo TLS 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 Duo TLS Module is g_tls_session. This name can be changed in the associated **Properties** window.)

**Table 2 NetX Duo TLS Module Selection Sequence**

| Resource | ISDE Tab | Stacks Selection Sequence |
|---|---|---|
| g_tls_session0 NetX Duo TLS Session | Threads | New Stack> X-Ware> NetX Duo> Protocols> NetX Duo TLS Session |

As shown in the following figure, when the NetX Duo TLS Session 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.

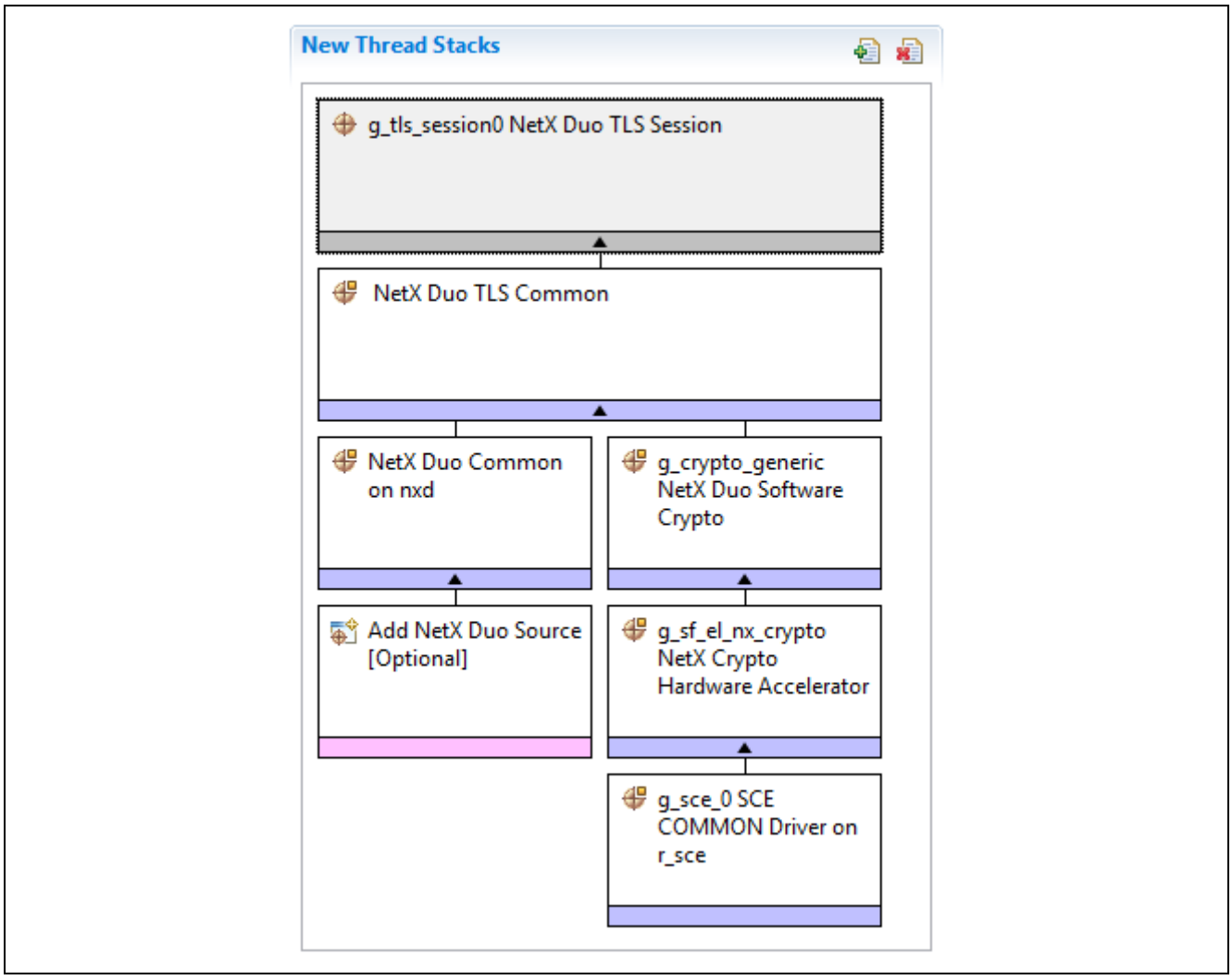


**Figure 4 NetX Duo TLS Module Stack**

To use TLS Secure with 'bare' TCP sockets, you will need to add an IP instance:

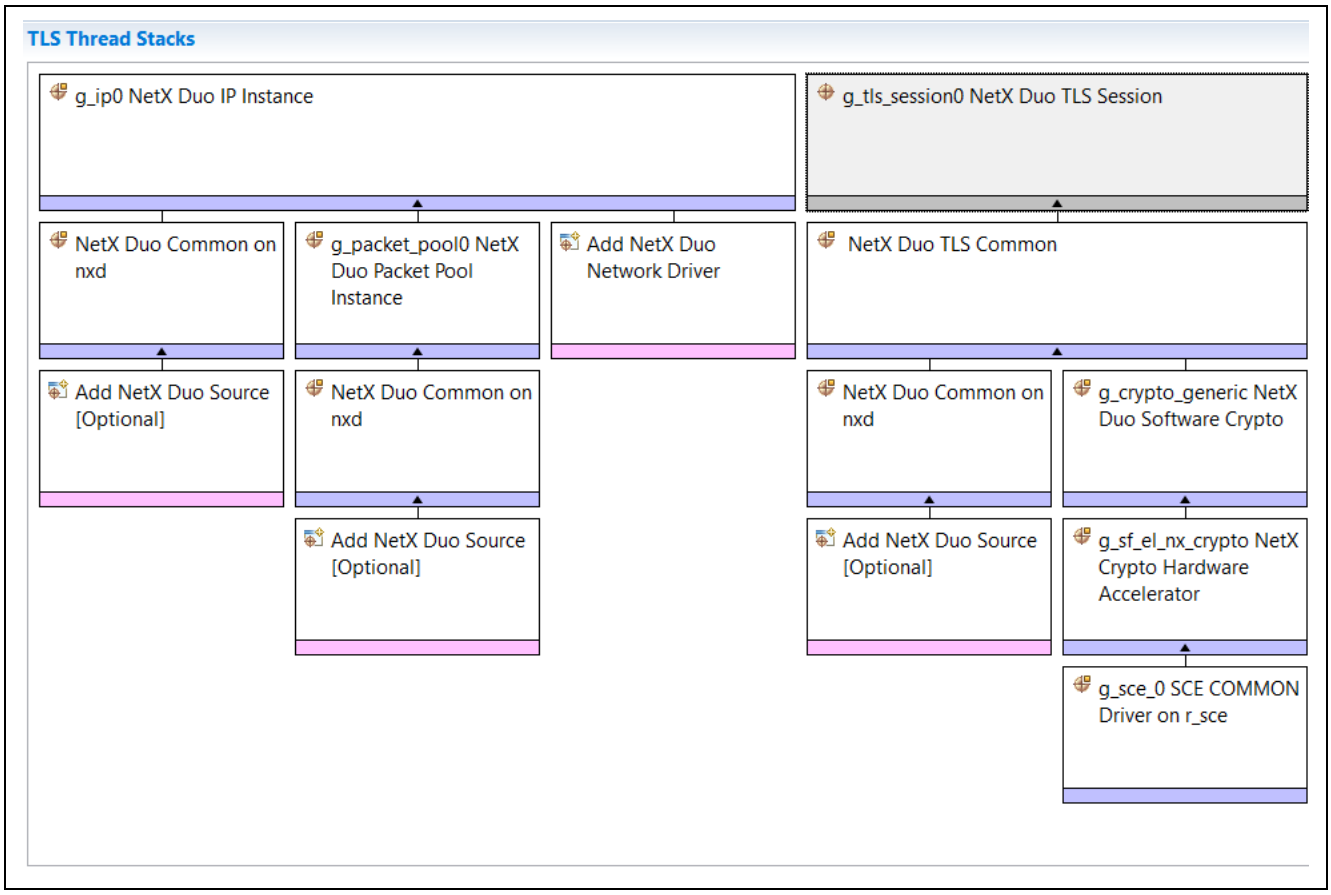


**Figure 5 TLS Secure session using NetX Duo sockets directly**

## 5. Configuring the NetX Duo TLS Session Module

Note that this module guide project creates two sessions, a TLS Client for connecting to a TLS Server, and a TLS Server session to accept connection requests from a TLS Client. The Client session runs and completes, then the Server session runs to completion.

The NetX Duo TLS Session 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 3 Configuration Settings for the NetX Duo TLS Client Session Module**

| ISDE Property | Value | Description |
|---|---|---|
| Name | g_tls_session0 | Module name |
| Meta data size | 4000 | Meta data size selection |
| Name of Timestamp Function | NULL | Name of callback for setting timestamp |
| Name of Certificate Verification Function | NULL | Name of callback for verifying the remote certificate |
| Name of generated initialization function | tls_dtls_session_init0 | Name of generated initialization function selection |

**Table 4 Configuration Settings for the NetX Duo TLS Server Session Module**

| ISDE Property | Value | Description |
|---|---|---|
| Name | g_tls_session1 | Module name |
| Meta data size | 4000 | Meta data size selection |
| Name of Timestamp Function* | NULL | Name of callback for setting timestamp |
| Name of Certificate Verification Function* | NULL | Name of callback for verifying the remote certificate |
| Name of generated initialization function | tls_dtls_session_init0 | Name of generated initialization function selection |

Notes: *The Certificate Verification Function provides the certificate being verified to the application so additional verification steps may be performed. It is set to NULL in the module guide project for the sake of simplicity but in a real TLS session it would be verification tool.

*The Time Stamp Function is discussed in section 3.1. It is omitted in this module guide in the interests of simplicity but it is typically used to check the expiration date of each certificate against the time provided by the application. Not having a timestamp callback may cause interoperability problems and reduce the security of production release TLS application sessions.

Note: The example values and defaults are for a project using the Synergy S7G2 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 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 Duo TLS Lower-Level Modules

Typically, 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 Duo TLS Common**

| ISDE Property | Value | Description |
|---|---|---|
| Crypto Engine | Hardware | Crypto engine selection |
| Self Signed Certificates** | Disable<br><br>Default: Disable | Self-signed certificates selection* |
| PSK Cipher Suite | Enable, Disable<br><br>Default: Disable | PSK cipher suite selection |
| X509 Strict Name Compare | Enable, Disable<br><br>Default: Disable | X509 strict name compare selection |
| X509 Extended Distinguished Names | Enable, Disable<br><br>Default: Disable | X509 extended distinguished names selection |
| Maximum RSA Modulus size (bits) | 1024, 2048, 3072, 4096<br><br>Default: 4096 | Maximum RSA modulus size (bits) selection |
| TLS v 1.0 | Enable, Disable<br><br>Default: Disable | Support TLS v 1.0 |
| TLS v 1.1 | Enable, Disable<br><br>Default: Disable | Support TLS v 1.1 |
| Server Mode | Enable, Disable<br><br>Default: Enable | Support Server mode |
| Client Mode | Enable, Disable<br><br>Default: Enable | Support Client mode |
| Name of generated initialization function | nx_secure_common_init | Name of generated initialization function selection; Initializes secure crypto engine driver and Initializes the various control data structures for the TLS component |
| Auto Initialization | Enable, Disable<br><br>Default: Enable | Enable auto initialization of general IP components including calling nx_secure_common_init |

Notes: 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.

** This option generally **should** be disabled in production mode. Enabling this feature is really useful for debugging because it removes the need to setup trusted certificates. However, enabling this option is also a serious security risk. So it is only intended for development and testing as it immediately treats any self-signed certificate as being trusted (this can be useful when testing a TLS implementation without a complete PKI available).

However, in TLS Secure in SSP 1.3.x, this option is enabled even if the property setting is set to Disabled. This is fixed in SSP 1.4.0. To disable this feature in SSP 1.3.x, modify the `nx_secure_tls.h` file in the project `synergy\ssp\src\framework\el\nxd_application_layer\nxd_tls_secure` directory to comment out `NX_SECURE_ALLOW_SELF_SIGNED_CERTIFICATES` directly. Make the file read only so it is not overwritten in the e$^2$ studio environment.

**Table 6 Configuration Settings for the NetX Duo Common Instance**

| ISDE Property | Value | Description |
|---|---|---|
| Name of generated initialization function | nx_common_init0 | Name of generated initialization function selection |
| Auto Initialization | Enable, Disable<br><br>Default: Enable | Auto initialization selection |

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

**Table 7 Configuration Settings for the NetX Duo Software Crypto**

| ISDE Property | Value | Description |
|---|---|---|
| Name | g_crypto_generic | Module name |

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

**Table 8 Configuration Settings for the NetX Crypto Hardware Accelerator**

| ISDE Property | Value | Description |
|---|---|---|
| Name | g_sf_el_nx_crypto | Module name |

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

**Table 9 Configuration Settings for the SCE HAL Module on r_sce**

| ISDE Property | Value | Description |
|---|---|---|
| Name | g_sce_0 | Module name |
| Endian Flag | CRYPTO_WORD_ENDIAN_BIG | Endian flag selection |

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

**Table 9 Configuration Settings for the NetX Duo IP Instance**

(Highlighted properties need to be configured for your system)

| ISDE Property | Value | Description |
|---|---|---|
| Name | g_ip0 | Module name |
| IPv4 Address (use commas for separation) | 192,168,0,2 | IPv4 Address selection |
| Subnet Mask (use commas for separation) | 255,255,255,0 | Subnet Mask selection |
| IPv6 Global Address (use commas for separation)* | 0x2001, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x1 | IPv6 global address selection |
| IPv6 Link Local Address (use commas for separation, All zeros means NetX Duo will create the LLA address using the device MAC address)* | 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0 | IPv6 link local address selection. This will be configured automatically if left at the default value, based on the device MAC address. |
| IP Helper Thread Stack Size (bytes) | 1024 | 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<br><br>Default: Disable | Reverse ARP selection |
| *TCP* | *Enable* | *TCP selection* |
| UDP | Enable, Disable<br><br>Default: Enable | UDP selection |
| ICMP | Enable, Disable<br><br>Default: Enable | ICMP selection |
| IGMP | Enable, Disable<br><br>Default: Enable | IGMP selection |
| IP fragmentation | Enable, Disable<br><br>Default: Disable | IP fragmentation selection |
| Name of generated initialization function | ip_init0 | Name of generated initialization function selection |
| Auto Initialization | Enable, Disable<br><br>Default: Enable | Auto initialization selection |

Note: *Only necessary if using IPv6 network connections

**Table 10 Configuration Settings for the NetX Duo Common Instance**

| ISDE Property | Value | Description |
|---|---|---|
| Name of generated initialization function | nx_common_init0 | Name of generated initialization function selection |
| Auto Initialization | Enable, Disable<br><br>Default: Enable | Auto initialization selection |

**Table 11 Configuration Settings for the NetX and NetX Duo Packet Pool Instance g_packet_pool0**

| ISDE Property | Value | Description |
|---|---|---|
| Name | g_packet_pool0 | Module name |
| Packet Size in Bytes | 1532 | Packet size selection – certificate packets can be quite large, exceeding the default 640 bytes. So to avoid packet chaining overhead, set the packet payload to the device MTU, which is usually 1518 or so |
| Number of Packets in Pool | 16 | Number of packets in pool selection. For servers with very large certificates, e.g. 4k bytes it might be necessary to increase this number accordingly. |
| Name of generated initialization function | packet_pool_init0 | Name of generated initialization function selection |
| Auto Initialization | Enable, Disable<br><br>Default: Enable | Auto initialization selection |

**Table 12 Configuration Settings for the NetX Port ETHER**

| ISDE Property | Value | Description |
|---|---|---|
| Parameter Checking | BSP, Enabled, Disabled<br><br>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), Priority 1:2, Priority 3 (CM4: valid, CM0+: lowest- not valid if using ThreadX), Priority 4:14 (CM4: valid, CM0+: invalid), Priority 15 (CM4 lowest - not valid if using ThreadX, CM0+: invalid)*<br><br>*Default: Disabled*<br>*Recommend Priority 3 for most applications* | *Ethernet interrupt priority selection. Change from Disabled for the network driver to send and receive packets.* |
| Name | g_sf_el_nx | Module name |
| *Channel*** | *1* | *Channel selection* |
| Callback | NULL | Callback selection |

Note: **This is specific to the SK-S7G2 MCU. The DK-S7G2 default values need not be modified.
For Wi-Fi networks, please refer to the Synergy Wi-Fi Application Project for SK-S7G2 - Application Project on the Renesas Gallery for more information on setting network parameters.

## 5.2 NetX Duo TLS 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 Duo TLS 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 $I^2C$ pins.

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

**Table 13 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 14 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 Synergy S7G2 MCU and the SK-S7G2 Kit. Other Synergy MCUs and other Synergy Kits may have different available pin configuration settings.

## 6. Using NetX Duo TLS for a TLS Session

The specific steps for configuring a TLS session are shown below

1. Add NetX Duo IP instance (**X-Ware -> NetX Duo -> NetX Duo IP Instance**). This automatically adds the IP instance default packet pool g_packet_pool0.
2. Click on **Add Network Driver** and choose **NetX Port Ether**
3. Add a TLS session (**X-Ware -> NetX Duo -> Protocols -> NetX Duo TLS**). This creates g_tls_session0 which will be used for a NetX Duo TLS Client session.
4. Repeat step #3 to create g_tls_session1. g_tls_session1 will be used for a NetX Duo TLS Server session.

Now the thread pane view looks as shown below. This is the project set up for this module guide.

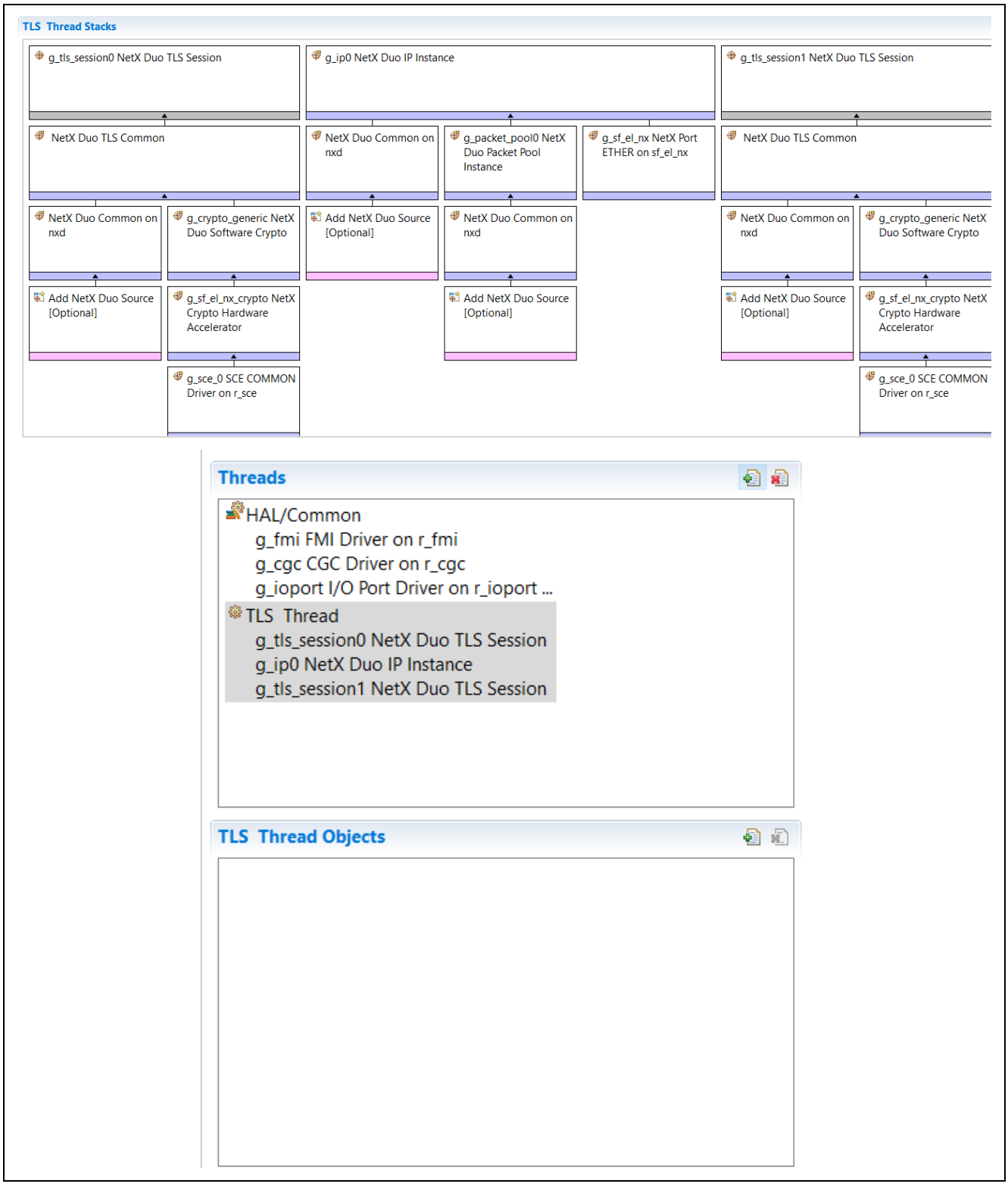


**Figure 6 NetX Duo TLS Secure Session**

5. Configuring the properties as described in the tables in Chapter 5. Then generate the code clicking on the **Generate Project Content** button.
6. This will autogenerate the code under the configured thread. This autogenerated code includes the initialization function with name as specified under the **Name of generated initialization function** property.
7. The name of the thread added in this example is the TLS Thread. Synergy automatically creates the `tls_thread_entry.c` function.

# 7. NetX Duo TLS Module Application Project

The project associated with this module guide demonstrates the steps to set up and run a TLS session. 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 Duo Secure module. You can also read over the code in `tls_thread_entry.c` which sets up and runs a TLS client session and then a TLS server session.

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

**Table 15 Software and Hardware Resources Used by the Application Project**

| Resource | Revision | Description |
|---|---|---|
| e$^2$ studio | 5.4.023 | Integrated Solution Development Environment |
| SSP | 1.3.3 | Synergy Software Platform |
| IAR EW for Synergy | 7.71.3 | IAR Embedded Workbench® for Renesas Synergy™ |
| SSC | 5.4.023 | Synergy Standalone Configurator |
| SK-S7G2 | v3.0 to v3.1 | Starter Kit |

You will need to start a TLS server using OpenSSL or similar utility before starting the module guide project. For more details on installing OpenSSL to your PC, go to the OpenSSL website www.openssl.org. To start a TLS server, open a command shell and cd into the project `src` folder where the cert and key files referred to below are contained. (You can also copy the same files into a separate folder and cd into that folder).

Note: You may need to use the Advanced Security options of your Firewall to allow TCP connections on the ports used by the TLS Client and TLS Server sessions (which by default are 103 and 403).

This is the command line to start a TLS server on a PC host for this project:

0 errors and 0 warnings

- 103 is the server listening port. It is defined in tls_thread_entry:

  `#define REMOTE_SERVER_PORT 103`

- `cert.pem` is the Server's certificate identifying itself
- `key.pem` is the key file for distributing public keys to the client to encrypt its secret during the Cipher Key Exchange. This key file is also necessary to be able to decrypt the data captured in the Wireshark packet trace if Wireshark is running.

Before compiling the project, note that `tls_thread_entry.c` defines the TLS Server IP address as

```
#define REMOTE_SERVER_IP_ADDRESS IP_ADDRESS(192, 2, 2, 201).
```

You will need to change this to match the IP address of your TLS server PC host. Then rebuild your project.

Now run the project. `tls_thread_entry.c` performs the following steps.

1. Initializes NetX Duo TLS by calling the `nx_secure_tls_initilialize` API. This needs to be called only once regardless how many TLS sessions are conducted.
2. Waits for the network link to be enabled by calling the `nx_ip_status_check` API.
3. It calls the `tls_client_session` function which creates a TCP socket for the TLS client g_tls_session0 to connect to the server on the PC host and exchange messages. When that is complete, the debug output displays:

   ```
   TLS Client completed with [number of errors] errors.
   ```

4. It calls `tls_server_session` function immediately after the TLS Client session ends. It creates a TCP server socket for the g_tls_sessoin1 to accept TLS client connection requests on the PC host. You can check your TLS Server is ready for Client requests when the debug output displays:

   ```
   TLS Server is ready for a connection request with [number of errors] errors
   so far
   ```

The “number of errors” come from the status returns from initializing packet buffers, initializing and adding certificates, and other TLS server session setup operations. Ideally there will be no errors. The TLS Server will wait 100 seconds for a client connection request before time out. So you should have plenty of time to set up a TLS Client session on your PC host:

5. Start an OpenSSL TLS client on a local PC by opening a command shell and cd into the module guide project `src` folder (where the cert file referred to is located). Run this command:

   ```
   openssl s_client -connect [server:port] -CAfile root_ca_cert.pem -no_comp
   ```

   where `server` is the IP address of your board, and `port` is the port it is listening on. In the module guide project, the TLS Server listening port defaults to `REMOTE_SERVER_PORT (103)`

The TLS server session uses a three-level public key interface for establishing a secure connection: Root CA certificate, Intermediate CA certificate, Device certificate.

The certificates required by the TLS server on the NetX Duo device are defined in `device_chain.h`. The certificates are converted to C array format for NetX TLS Secure from the original `device_chain.der` file format. `Device_cert_der` defines the local certificate, `device_cert_der_key` defines the key for the certificate and the `ica_cert_der` defines the Intermediate CA certificate for the TLS Server. The server needs to send both local and ICA certificates to the OpenSSL TLS client during the TLS handshake.

The `root_ca_cert.pem` contains the root CA certificate for the PKI. The OpenSS TLS client will only need to have the root CA certificate, which does not require a key. The -no_comp turns off compression which NetX Duo TLS Secure does not support.

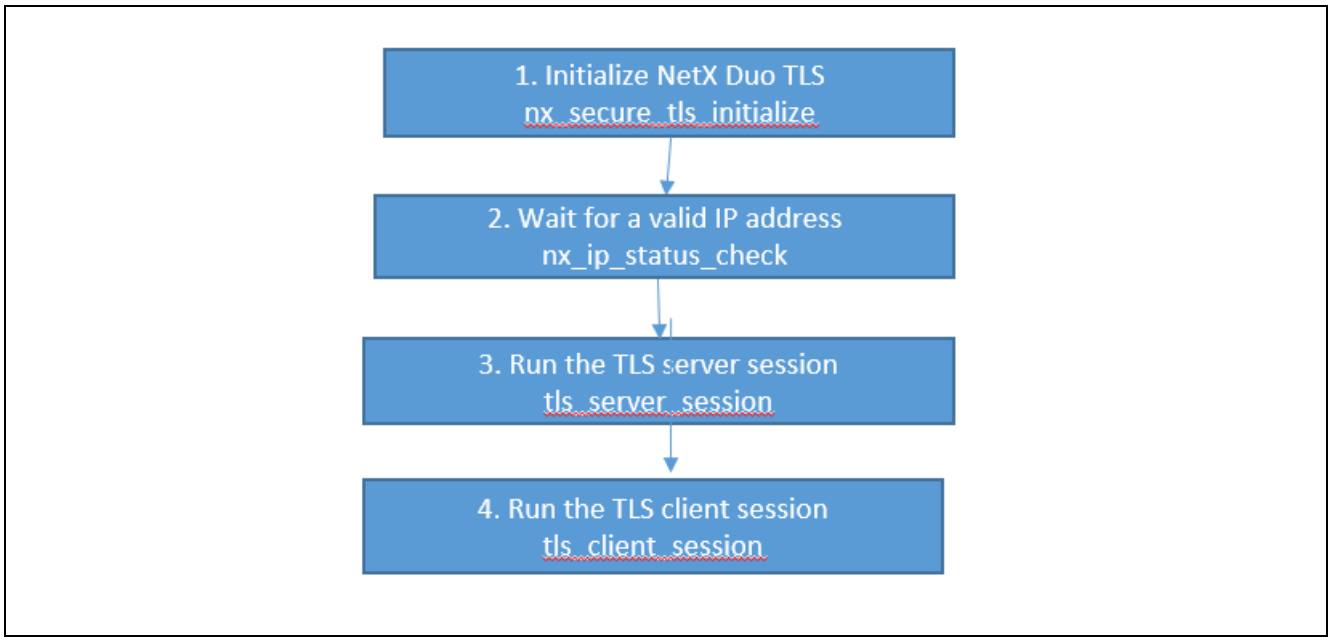


**Figure 7 Main thread entry function for TLS module guide project**

Below are the details of how `tls_client_session()` sets up and runs a secure TCP Client session.

1. Create a TCP socket by calling the `nx_tcp_socket_create` API.
2. Set up a buffer for TLS packet assembly using the `nx_secure_tls_session_packet_buffer_set` API.
3. Bind to a local port by calling the `nx_tcp_client_socket_bind` API.
4. Allocate space for processing received server certificate data by calling the `nx_secure_tls_remote_certificate_allocate` API, for the remote server certificate and for the remote server certificate issuer certificate.
5. Initialize the local identity certificate by calling the `nx_secure_x509_certificate_initialize` API. Do the same for the trusted CA (Certificate Authority) certificate.
6. Add the local and trusted certificates by calling the `nx_secure_tls_local_certificate_add` API and `nx_secure_tls_trusted_certificate_add` API respectively

7. Connect to the TLS server by calling the `nx_tcp_client_socket_connect` API.
8. Start the TLS session by calling the `nx_secure_tls_session_start` API.
9. Allocate a packet for sending data. Because this packet must have the TLS header data, use the `nx_secure_tls_packet_allocate` API.
10. Add data to the packet by calling the `nx_packet_data_append` API.
11. Send the packet by calling the `nx_secure_tls_session_send` API.
12. Receive the TLS server response by calling the `nx_secure_tls_session_receive` API.
13. End the TLS session by calling the `nx_secure_tls_session_end` API.
14. The TLS client socket is still connected to the Server socket at this point, so terminate the TCP socket connection by calling the `nx_tcp_socket_disconnect` API.
15. Release the local port by calling the `nx_tcp_client_socket_unbind` API.
16. Delete the TCP socket by calling the `nx_tcp_socket` delete API.
17. Delete the TLS session by calling the `nx_secure_tls_session_delete` API. See comments in 7.1.1. about ending a TLS session below.

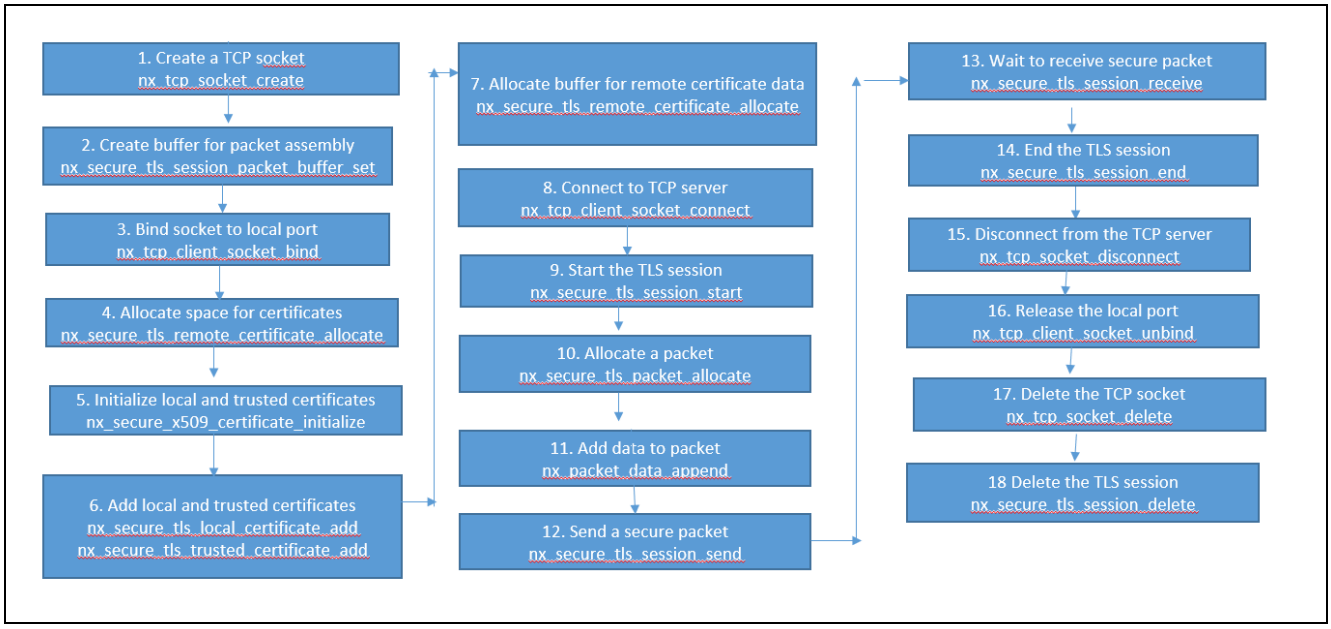


**Figure 8 Running the TLS Client session in the TLS module guide project**

Below are the details of how `tls_server_session()` sets up and runs a secure TCP Server session:

1. Create a TCP socket by calling the nx_tcp_socket_create API.
2. Set up a buffer for TLS packet assembly using the `nx_secure_tls_session_packet_buffer_set` API.
3. Listen on a local port by calling the `nx_tcp_server_socket_listen` API.
4. Initialize the local certificate by calling `nx_secure_x509_certificate_initialize` API. Do the same for the intermediate CA (ICA) certificate.
5. Register the local and ICA certificates to the TLS session by calling the `nx_secure_tls_local_certificate_add` API.
6. Wait for a Client connection request by calling the `nx_tcp_server_socket_accept` API.
7. Start the TLS session by calling the `nx_secure_tls_session_start` API.
8. Receive the TLS Client data by calling the `nx_secure_tls_session_receive` API.

9. Extract data by calling the `nx_packet_data_extract_offset` API.
10. Allocate a packet for sending data. Because this packet must have the TLS header data, use the `nx_secure_tls_packet_allocate` API.
11. Add data to the packet by calling the `nx_packet_data_append` API.
12. Send the packet by calling the `nx_secure_tls_session_send` API.
13. End the TLS session by calling the `nx_secure_tls_session_end` API.
14. Terminate the TCP socket connection by the calling `nx_tcp_socket_disconnect` API.
15. Release the local port by calling the `nx_tcp_server_socket_unaccept` API.
16. Delete the TCP socket by calling the `nx_tcp_socket_delete` API.
17. Delete the TLS session by calling the `nx_secure_tls_session_delete` API.

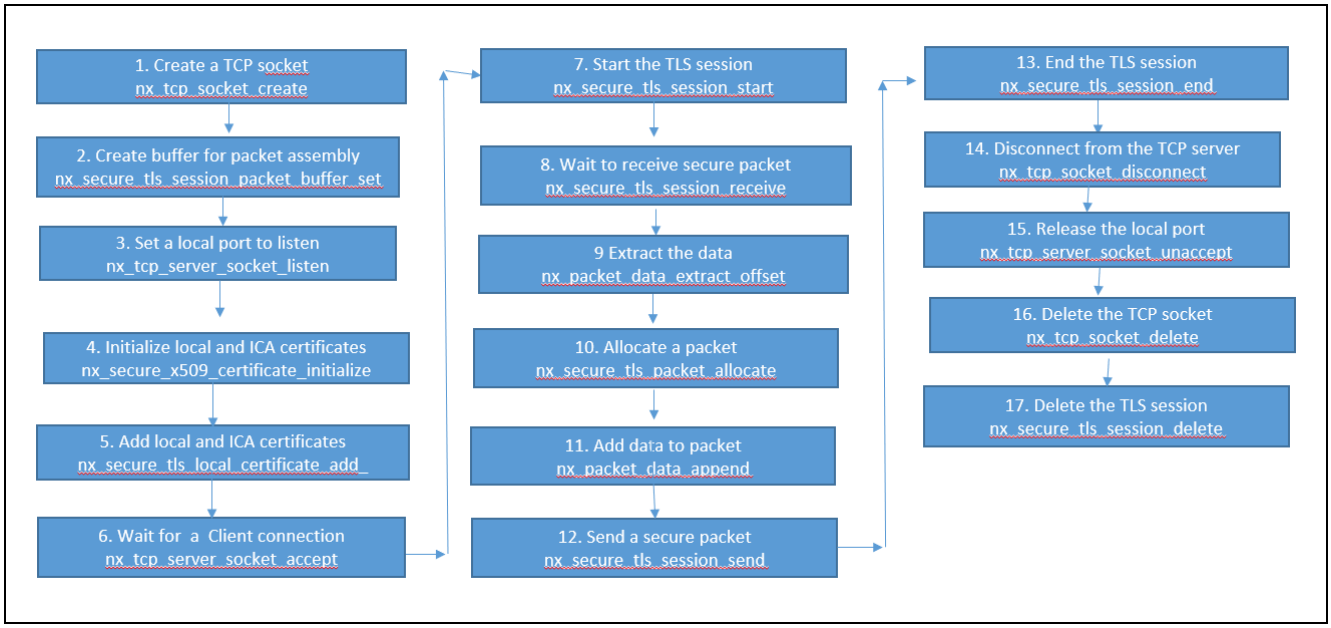


**Figure 9 Running the TLS Server session in the TLS module guide project**

## 7.1 Ending a TLS Client session

If a TLS Client connection fails or the session is not properly connected, you **must** end the TLS session by calling `nx_secure_tls_session_end` to properly reset the TLS session e.g. release certificates that are registered with the TLS session. If your application does not do this, your application may not be able to re-register certificates to the TLS session if your application tries to make another TLS connection. All valid TLS sessions **must** be ended using `nx_secure_tls_session_end` as well to ensure proper shutdown of the TLS session. Failure to close the TLS session properly is a security risk.

It is not necessary to delete a TLS session if you plan to reuse it for a subsequent connection. If you do delete a TLS session, you will need to re-register the local and trusted certificates as well as allocate resources for the server certificate data.

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

**Table 16 Software and Hardware Resources Used by the Application Project**

| Resource | Revision | Description |
|---|---|---|
| e² studio | 5.4.023 | Integrated Solution Development Environment |
| SSP | 1.3.3 | Synergy Software Platform |
| IAR EW for Synergy | 7.71.3 | IAR Embedded Workbench® for Renesas Synergy™ |
| SSC | 5.4.023 | Synergy Standalone Configurator |
| SK-S7G2 | v3.0 to v3.1 | Starter Kit |

## 8. Customizing the NetX Duo TLS Module for a Target Application

TLS security can be added to MQTT and HTTP applications. For MQTT, set up the TLS session in the callback function specified in the nxd_mqtt_client_connect call to create the TLS session*, set up optional timestamp and certificate verification callbacks, register local certificates and allocate memory for processing server certificate data.

Note: *Before re-creating the TLS session for subsequent connection attempts after the first connection, it is strongly recommended to clear the NXD_SECURE_TLS_SESSION data block:

```
NX_SECURE_TLS_SESSION * p_tls_session;
p_tls_session = &(g_mqtt_client0.nxd_mqtt_tls_session);
memset(p_tls_session, 0, sizeof(NX_SECURE_TLS_SESSION));
```

This applies to applications created in SSP 1.3.x. This `memset` call is not necessary in SSP 1.4.0 when recreating a TLS session.

## 9. Running the NetX Duo TLS Module Application Project

To run the NetX Duo TLS 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 *Synergy Project Import Guide* (r11an0023eu0120-synergy-ssp-import-guide.pdf, included in this package) for instructions on importing the project into e² 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 the debug connector J19 on SK-S7G2 Kit.
3. Add the code from the supplied project file `tls_thread_entry.c` or copy over the generated `tls_thread_entry.c` file. Do the same for the file.
4. Right click on the project and choose **New -> Source File**. Enter the name of the new source file, for example, `NetXDuo_TLS_client.c` for the client session and `NetXDuo_TLS_Server.c`. Copy the code from the supplied project files `tls_client_session.c` and `tls_server_session.c` files into these files respectively.
5. Add the Client local and trusted certification files `in test_ca_cert.h` and `test_device_cert.h`. Alternatively copy these arrays into a header file and add that header file to the project.
6. Set optimization to none for easier debugging and stepping through the code. If you only wish to run the project and not debug, use the default level of -O2:

   Right click the project and choose **Properties -> C/C++ Build -> Settings -> Optimization**. Set the Optimization Level to None (-O2).
7. Enable debug output [optional]. This is done most easily by ensuring that the `SEMI_HOSTING` macro in the `tls_thread_entry.h` header file is defined.
8. Make sure the NetX Ether Port component has the changes specified in the table in Chapter 5. Also check the IP and packet pool components to match your environment, for rxample, IP address and local source ports.
9. Click on **Generate Project Content** when you change anything in the thread pane stack or project settings.

10. Right-click the project and choose **Build Project**.
11. Start an OpenSSL session from a command shell. This assumes you have OpenSSL installed. Open the command shell as Administrator. Run this command:

```
    openssl.exe s_server -accept [listening port (default is 103)] -cert
server_cert.pem -certform PEM -key device.key.pem
```

    where the server's identity certificate is `cert.pem` and the key file is `device_key.pem`. These are supplied with the project but you may supply your own. The listening port can be any port but TLS Servers will typically listen on port 4443.
12. Connect to the host PC via Ethernet on the J11 connector on SK-S7G2 Kit.
13. Put a breakpoint on the line in the `tls_thread_entry` function where it calls `tls_server_session()`.
14. Run the application by right-clicking the project and choose **Debug as -> Renesas GDB Hardware Debugging**. This will run the TLS Client session first, which is why we set up the OpenSSL server above.
15. Verify from the command shell that the TLS server has accepted the NetX Duo TLS client session connection request.

    Optionally, you can run Wireshark to capture the packet trace. To decrypt the data afterwards, select a frame of TCP packet exchange, right-click and choose **Protocol Preferences -> Open data preferences**. Select SSL from the left column. Click on the **Edit** button for the RSA keys list and add a new entry specifying the server key file location, TCP and IP address, and port of the server. In the figure below 192.2.2.66 is the NetX Duo device connecting to the TLS server (who is listening on port 103)

| | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 1 | 0.000000 | 192.2.2.66 | 192.2.2.201 | TCP | 62 | 403 → 103 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 |
| 2 | 0.000105 | 192.2.2.201 | 192.2.2.66 | TCP | 58 | 103 → 403 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 |
| 3 | 0.000402 | 192.2.2.66 | 192.2.2.201 | TCP | 60 | 403 → 103 [ACK] Seq=1 Ack=1 Win=8192 Len=0 |
| 4 | 0.108724 | 192.2.2.66 | 192.2.2.201 | TLSv1.2 | 118 | Client Hello |
| 5 | 0.149309 | 192.2.2.201 | 192.2.2.66 | TCP | 54 | 103 → 403 [ACK] Seq=1 Ack=65 Win=64176 Len=0 |
| 6 | 4.605727 | 192.2.2.201 | 192.2.2.66 | TLSv1.2 | 140 | Server Hello |
| 7 | 4.746451 | 192.2.2.66 | 192.2.2.201 | TCP | 60 | 403 → 103 [ACK] Seq=65 Ack=87 Win=8192 Len=0 |
| 8 | 4.746530 | 192.2.2.201 | 192.2.2.66 | TLSv1.2 | 998 | Certificate |
| 9 | 4.946435 | 192.2.2.66 | 192.2.2.201 | TCP | 60 | 403 → 103 [ACK] Seq=65 Ack=1031 Win=8192 Len=0 |
| 10 | 4.946480 | 192.2.2.201 | 192.2.2.66 | TLSv1.2 | 63 | Server Hello Done |
| 11 | 4.948521 | 192.2.2.66 | 192.2.2.201 | TLSv1.2 | 321 | Client Key Exchange |
| 12 | 4.949350 | 192.2.2.66 | 192.2.2.201 | TLSv1.2 | 60 | Change Cipher Spec |
| 13 | 4.949431 | 192.2.2.201 | 192.2.2.66 | TCP | 54 | 103 → 403 [ACK] Seq=1040 Ack=338 Win=63903 Len=0 |
| 14 | 4.949559 | 192.2.2.66 | 192.2.2.201 | TLSv1.2 | 123 | Encrypted Handshake Message |
| 15 | 4.990215 | 192.2.2.201 | 192.2.2.66 | TCP | 54 | 103 → 403 [ACK] Seq=1040 Ack=407 Win=63834 Len=0 |
| 16 | 5.054337 | 192.2.2.201 | 192.2.2.66 | TLSv1.2 | 60 | Change Cipher Spec |
| 17 | 5.246353 | 192.2.2.66 | 192.2.2.201 | TCP | 60 | 403 → 103 [ACK] Seq=407 Ack=1046 Win=8192 Len=0 |
| 18 | 5.246409 | 192.2.2.201 | 192.2.2.66 | TLSv1.2 | 123 | Encrypted Handshake Message |
| 19 | 5.247044 | 192.2.2.66 | 192.2.2.201 | TLSv1.2 | 123 | Application Data |
| 20 | 5.263504 | 192.2.2.201 | 192.2.2.66 | TLSv1.2 | 123 | Application Data |
| 21 | 5.263983 | 192.2.2.66 | 192.2.2.201 | TLSv1.2 | 107 | Encrypted Alert |
| 22 | 5.291615 | 192.2.2.201 | 192.2.2.66 | TCP | 54 | 103 → 403 [FIN, ACK] Seq=1184 Ack=529 Win=63712 Len=0 |
| 23 | 5.357963 | 192.2.2.66 | 192.2.2.201 | TCP | 60 | 403 → 103 [FIN, ACK] Seq=529 Ack=1184 Win=8192 Len=0 |
| 24 | 5.357964 | 192.2.2.66 | 192.2.2.201 | TCP | 60 | 403 → 103 [ACK] Seq=530 Ack=1185 Win=8192 Len=0 |

16. When the breakpoint is reached, open another command shell and start an OpenSSL client session with this command:

```
openssl s_client -connect server:port -CAfile root_ca_cert.pem -no_comp
```

    where `server` is the IP address of the NetX Duo TLS device and `port` is the port it is listening on, and compression is turned off.
17. Verify the TLS session is started from the command shell output. It will pause and wait for your keyboard input for a message to the server. Type something simple like "Hello" and hit enter.

    If you were running Wireshark during this session it might look something like this (192.2.2.66 is the NetX Duo TLS Server):

| | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 1 | 0.000000 | 192.2.2.201 | 192.2.2.66 | TCP | 66 | 58817 → 103 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 2 | 0.000311 | 192.2.2.66 | 192.2.2.201 | TCP | 62 | 103 → 58817 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 |
| 3 | 0.000373 | 192.2.2.201 | 192.2.2.66 | TCP | 54 | 58817 → 103 [ACK] Seq=1 Ack=1 Win=64240 Len=0 |
| 4 | 0.000965 | 192.2.2.201 | 192.2.2.66 | TLSv1.2 | 362 | Client Hello |
| 5 | 0.018709 | 192.2.2.66 | 192.2.2.201 | TLSv1.2 | 101 | Server Hello |
| 6 | 0.018944 | 192.2.2.66 | 192.2.2.201 | TLSv1.2 | 1051 | Certificate |
| 7 | 0.018944 | 192.2.2.66 | 192.2.2.201 | TLSv1.2 | 63 | Server Hello Done |
| 8 | 0.018976 | 192.2.2.201 | 192.2.2.66 | TCP | 54 | 58817 → 103 [ACK] Seq=309 Ack=1054 Win=63187 Len=0 |
| 9 | 0.028795 | 192.2.2.201 | 192.2.2.66 | TLSv1.2 | 396 | Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message |
| 10 | 0.142167 | 192.2.2.66 | 192.2.2.201 | TLSv1.2 | 60 | Change Cipher Spec |
| 11 | 0.142296 | 192.2.2.66 | 192.2.2.201 | TLSv1.2 | 123 | Encrypted Handshake Message |
| 12 | 0.142342 | 192.2.2.201 | 192.2.2.66 | TCP | 54 | 58817 → 103 [ACK] Seq=651 Ack=1129 Win=63112 Len=0 |
| 13 | 2.519868 | 192.2.2.201 | 192.2.2.66 | TLSv1.2 | 107 | Application Data |
| 14 | 2.644036 | 192.2.2.66 | 192.2.2.201 | TLSv1.2 | 331 | Application Data |
| 15 | 2.684149 | 192.2.2.201 | 192.2.2.66 | TCP | 54 | 58817 → 103 [ACK] Seq=704 Ack=1406 Win=62835 Len=0 |
| 16 | 2.753992 | 192.2.2.66 | 192.2.2.201 | TLSv1.2 | 107 | Encrypted Alert |
| 17 | 2.753993 | 192.2.2.66 | 192.2.2.201 | TCP | 60 | 103 → 58817 [FIN, ACK] Seq=1459 Ack=704 Win=8192 Len=0 |
| 18 | 2.754097 | 192.2.2.201 | 192.2.2.66 | TCP | 54 | 58817 → 103 [ACK] Seq=704 Ack=1460 Win=62782 Len=0 |
| 19 | 2.755131 | 192.2.2.201 | 192.2.2.66 | TLSv1.2 | 107 | Encrypted Alert |
| 20 | 2.755361 | 192.2.2.201 | 192.2.2.66 | TCP | 54 | 58817 → 103 [FIN, ACK] Seq=757 Ack=1460 Win=62782 Len=0 |
| 21 | 2.755570 | 192.2.2.66 | 192.2.2.201 | TCP | 60 | 103 → 58817 [ACK] Seq=1460 Ack=758 Win=8139 Len=0 |

18. At the completion of the NetX Duo TLS server and client sessions, the `tls_thread_entry` function will display a result of the number of errors encountered during the sessions in the Renesas Virtual Debug Console.
19. Debug output from the TLS Client and TLS server sessions respectively can be viewed in the Renesas Virtual Debug Console in the e2 studio environment or the Terminal I/O in IAR:

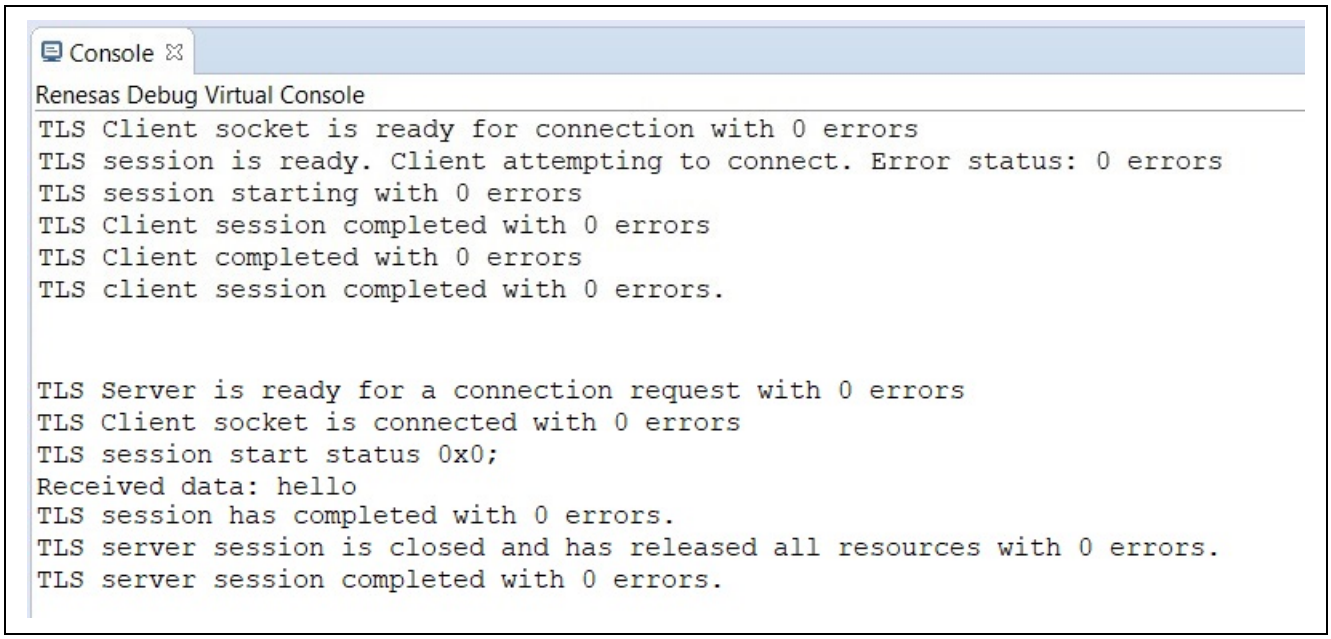


**Figure 10 Example Output from NetX Duo TLS Module Application Project**

## 9.1 Output from the OpenSSL TLS server:

```
openssl.exe s_server -accept 103 -cert cert.pem -certform PEM -key keyout.pem
-rev -debug -msg

Using default temp DH parameters
ACCEPT
bad gethostbyaddr***
read from 0x60006b850 [0x60008e380] (11 bytes => 11 (0xB))
0000 - 16 03 03 00 3b 01 00 00-37 03 03                  ....;...7..
<<< ??? [length 0005]
    16 03 03 00 3b
read from 0x60006b850 [0x60008e38e] (53 bytes => 53 (0x35))
0000 - 00 00 00 00 4b b5 f6 46-47 03 31 29 30 70 5b 04   ....K..FG.1)0p[.
0010 - 20 fd 5d b4 1a 8b 7f 78-50 29 59 d8 2b 89 48 68    .]....xP)Y.+.Hh
0020 - 00 00 04 00 2f 00 ff 01-00 00 0a 00 0d 00 06 00   ..../...........
0030 - 04 02 01 04 01                                    .....
```

```
<<< TLS 1.2 Handshake [length 003b], ClientHello
    01 00 00 37 03 03 00 00 00 00 4b b5 f6 46 47 03
    31 29 30 70 5b 04 20 fd 5d b4 1a 8b 7f 78 50 29
    59 d8 2b 89 48 68 00 00 04 00 2f 00 ff 01 00 00
    0a 00 0d 00 06 00 04 02 01 04 01
>>> ??? [length 0005]
    16 03 03 00 51
>>> TLS 1.2 Handshake [length 0051], ServerHello
    02 00 00 4d 03 03 97 35 a5 51 00 bd 26 01 7c 71
    3e 76 96 e8 d2 ba b7 25 df 10 6f 49 6f 47 22 03
    61 6d 2b a0 87 86 20 61 91 a5 94 95 c0 d7 ff 7e
    ff 3b 0a c5 c4 f5 f1 69 50 c7 f0 72 e1 19 f4 d9
    41 14 d0 7b d5 ea 06 00 2f 00 00 05 ff 01 00 01
    00
write to 0x60006b850 [0x60006ba40] (86 bytes => 86 (0x56))
0000 - 16 03 03 00 51 02 00 00-4d 03 03 97 35 a5 51 00   ....Q...M...5.Q.
0010 - bd 26 01 7c 71 3e 76 96-e8 d2 ba b7 25 df 10 6f   .&.|q>v.....%..o
0020 - 49 6f 47 22 03 61 6d 2b-a0 87 86 20 61 91 a5 94   IoG".am+... a...
0030 - 95 c0 d7 ff 7e ff 3b 0a-c5 c4 f5 f1 69 50 c7 f0   ....~.;.....iP..
0040 - 72 e1 19 f4 d9 41 14 d0-7b d5 ea 06 00 2f 00 00   r....A..{..../..
0050 - 05 ff 01 00 01                                    .....
0056 - <SPACES/NULS>
>>> ??? [length 0005]
    16 03 03 03 ab
>>> TLS 1.2 Handshake [length 03ab], Certificate
    0b 00 03 a7 00 03 a4 00 03 a1 30 82 03 9d 30 82
    02 85 a0 03 02 01 02 02 09 00 c8 35 91 ae 2d 2d
    7a 9f 30 0d 06 09 2a 86 48 86 f7 0d 01 01 0b 05
    00 30 65 31 0b 30 09 06 03 55 04 06 13 02 55 53
    31 0b 30 09 06 03 55 04 08 0c 02 43 41 31 13 30
    11 06 03 55 04 07 0c 0a 53 61 6e 20 4d 61 72 63
    6f 73 31 0b 30 09 06 03 55 04 0a 0c 02 45 4c 31
    11 30 0f 06 03 55 04 0b 0c 08 53 65 63 75 72 69
    74 79 31 14 30 12 06 03 55 04 03 0c 0b 31 39 32
    2e 32 2e 32 2e 31 34 39 30 1e 17 0d 31 36 30 36
    31 37 30 30 32 39 34 39 5a 17 0d 31 36 30 37 31
    37 30 30 32 39 34 39 5a 30 65 31 0b 30 09 06 03
    55 04 06 13 02 55 53 31 0b 30 09 06 03 55 04 08
    0c 02 43 41 31 13 30 11 06 03 55 04 07 0c 0a 53
    61 6e 20 4d 61 72 63 6f 73 31 0b 30 09 06 03 55
    04 0a 0c 02 45 4c 31 11 30 0f 06 03 55 04 0b 0c
    08 53 65 63 75 72 69 74 79 31 14 30 12 06 03 55
    04 03 0c 0b 31 39 32 2e 32 2e 32 2e 31 34 39 30
    82 01 22 30 0d 06 09 2a 86 48 86 f7 0d 01 01 01
    05 00 03 82 01 0f 00 30 82 01 0a 02 82 01 01 00
    b5 20 c1 b4 bc d0 15 20 b6 05 9d 49 5c 1b a8 b5
    86 f6 5d ec 32 6e 25 2d a9 a1 04 57 d0 2b 1a c9
    4b 75 90 3a 68 bd 75 9a 08 e7 a2 96 45 c2 c4 c8
    0f 74 38 70 08 98 3e 3e 03 d1 32 47 d4 ff f7 0f
    0c 25 e5 a1 75 e8 81 12 a1 a3 23 51 bc 64 d3 69
    f4 46 40 f1 91 8e 62 bc ab a4 9f 8e 29 7c 4f 91
    d8 6d 31 a2 e2 f2 8e 34 02 a0 03 d5 c5 ae 9e 8c
    1a 13 cc 04 eb 94 b0 43 dd 63 96 60 ff 37 a8 c2
    ca 17 25 d8 e7 58 4a f9 9d e2 fe 5d 77 a0 1d 7f
    17 11 27 81 21 6b 1a 51 fa 60 8c 16 26 88 64 7e
    e8 47 8d fd 9c dc 13 db 05 8f 73 8f e7 26 ee 85
    79 16 4e af 3d 2c 4e 5b 05 2c 67 bb ba c4 fa 05
    a4 75 9e 74 38 7c fc 7f 9b d1 1f 4a d8 f5 1f e3
```

```
    5f 50 32 22 d1 93 fc 93 ab 63 f9 72 d7 c7 45 f3
    a3 05 ea e1 cc 65 46 7e 6e d7 3d 50 06 69 f7 28
    96 57 2b e1 90 55 b1 9b 1c 1b ee 82 a2 a0 36 23
    02 03 01 00 01 a3 50 30 4e 30 1d 06 03 55 1d 0e
    04 16 04 14 40 8e d9 dc 45 b5 ed ed b9 a2 c2 27
    6e e7 57 2b ab f2 1d 97 30 1f 06 03 55 1d 23 04
    18 30 16 80 14 40 8e d9 dc 45 b5 ed ed b9 a2 c2
    27 6e e7 57 2b ab f2 1d 97 30 0c 06 03 55 1d 13
    04 05 30 03 01 01 ff 30 0d 06 09 2a 86 48 86 f7
    0d 01 01 0b 05 00 03 82 01 01 00 98 63 13 91 a6
    8b b0 0e ef 9a 29 04 e4 07 d3 4d 7c a1 aa a2 a5
    ac 77 62 15 73 e1 dc fe 9d b3 5e 4f a0 66 69 d5
    4b b9 d1 4c d5 cc fa 09 a5 d7 04 8a dc 42 ca be
    f1 76 95 0b e5 2e 11 2a f8 58 05 4a 90 40 e9 74
    ab 1a 8f 55 74 b9 79 76 6e 09 93 b4 e1 9c 99 9d
    a5 da b7 30 0f a1 87 1e 2e 1a 9d 5f fb 65 92 aa
    27 84 eb de ea 8e 4f db 74 d1 30 e1 6b 10 c7 42
    20 4a 74 46 ae 0c 20 98 18 2d 97 9a 42 3c 43 30
    d6 47 49 2a e6 b6 79 46 bb bd a8 76 1b 63 81 88
    25 52 20 34 5e 14 77 ee dd 97 61 c9 6c 0a 3f a3
    b3 88 1f 70 ab 1c 13 41 79 73 2a fe 6f a6 b3 34
    5c a5 21 72 e8 4c 76 ad d6 01 c2 45 85 d0 31 1b
    91 25 e6 36 d3 fb 0b 8f d3 88 77 01 e5 14 65 88
    58 db 2a f0 49 33 5a 7b 8c 77 2f ff 65 43 0c ab
    cb dd 8e ae 93 17 9f 6f 55 bc 38 1b a9 6e be 18
    d4 b2 b9 23 02 2b fc 69 9b d9 0d
write to 0x60006b850 [0x60006ba40] (944 bytes => 944 (0x3B0))
0000 - 16 03 03 03 ab 0b 00 03-a7 00 03 a4 00 03 a1 30   ...............0
0010 - 82 03 9d 30 82 02 85 a0-03 02 01 02 02 09 00 c8   ...0............
0020 - 35 91 ae 2d 2d 7a 9f 30-0d 06 09 2a 86 48 86 f7   5..--z.0...*.H..
0030 - 0d 01 01 0b 05 00 30 65-31 0b 30 09 06 03 55 04   ......0e1.0...U.
0040 - 06 13 02 55 53 31 0b 30-09 06 03 55 04 08 0c 02   ...US1.0...U....
0050 - 43 41 31 13 30 11 06 03-55 04 07 0c 0a 53 61 6e   CA1.0...U....San
0060 - 20 4d 61 72 63 6f 73 31-0b 30 09 06 03 55 04 0a    Marcos1.0...U..
0070 - 0c 02 45 4c 31 11 30 0f-06 03 55 04 0b 0c 08 53   ..EL1.0...U....S
0080 - 65 63 75 72 69 74 79 31-14 30 12 06 03 55 04 03   ecurity1.0...U..
0090 - 0c 0b 31 39 32 2e 32 2e-32 2e 31 34 39 30 1e 17   ..192.2.2.1490..
00a0 - 0d 31 36 30 36 31 37 30-30 32 39 34 39 5a 17 0d   .160617002949Z..
00b0 - 31 36 30 37 31 37 30 30-32 39 34 39 5a 30 65 31   160717002949Z0e1
00c0 - 0b 30 09 06 03 55 04 06-13 02 55 53 31 0b 30 09   .0...U....US1.0.
00d0 - 06 03 55 04 08 0c 02 43-41 31 13 30 11 06 03 55   ..U....CA1.0...U
00e0 - 04 07 0c 0a 53 61 6e 20-4d 61 72 63 6f 73 31 0b   ....San Marcos1.
00f0 - 30 09 06 03 55 04 0a 0c-02 45 4c 31 11 30 0f 06   0...U....EL1.0..
0100 - 03 55 04 0b 0c 08 53 65-63 75 72 69 74 79 31 14   .U....Security1.
0110 - 30 12 06 03 55 04 03 0c-0b 31 39 32 2e 32 2e 32   0...U....192.2.2
0120 - 2e 31 34 39 30 82 01 22-30 0d 06 09 2a 86 48 86   .1490.."0...*.H.
0130 - f7 0d 01 01 01 05 00 03-82 01 0f 00 30 82 01 0a   ............0...
0140 - 02 82 01 01 00 b5 20 c1-b4 bc d0 15 20 b6 05 9d   ...... ..... ...
0150 - 49 5c 1b a8 b5 86 f6 5d-ec 32 6e 25 2d a9 a1 04   I\.....].2n%-...
0160 - 57 d0 2b 1a c9 4b 75 90-3a 68 bd 75 9a 08 e7 a2   W.+..Ku.:h.u....
0170 - 96 45 c2 c4 c8 0f 74 38-70 08 98 3e 3e 03 d1 32   .E....t8p..>>..2
0180 - 47 d4 ff f7 0f 0c 25 e5-a1 75 e8 81 12 a1 a3 23   G.....%..u.....#
0190 - 51 bc 64 d3 69 f4 46 40-f1 91 8e 62 bc ab a4 9f   Q.d.i.F@...b....
01a0 - 8e 29 7c 4f 91 d8 6d 31-a2 e2 f2 8e 34 02 a0 03   .)|O..m1....4...
01b0 - d5 c5 ae 9e 8c 1a 13 cc-04 eb 94 b0 43 dd 63 96   ............C.c.
01c0 - 60 ff 37 a8 c2 ca 17 25-d8 e7 58 4a f9 9d e2 fe   `.7....%..XJ....
01d0 - 5d 77 a0 1d 7f 17 11 27-81 21 6b 1a 51 fa 60 8c   ]w.....'.!k.Q.`.
01e0 - 16 26 88 64 7e e8 47 8d-fd 9c dc 13 db 05 8f 73   .&.d~.G........s
```

```
01f0 - 8f e7 26 ee 85 79 16 4e-af 3d 2c 4e 5b 05 2c 67   ..&..y.N.=,N[.,g
0200 - bb ba c4 fa 05 a4 75 9e-74 38 7c fc 7f 9b d1 1f   ......u.t8|.....
0210 - 4a d8 f5 1f e3 5f 50 32-22 d1 93 fc 93 ab 63 f9   J...._P2".....c.
0220 - 72 d7 c7 45 f3 a3 05 ea-e1 cc 65 46 7e 6e d7 3d   r..E......eF~n.=
0230 - 50 06 69 f7 28 96 57 2b-e1 90 55 b1 9b 1c 1b ee   P.i.(.W+..U.....
0240 - 82 a2 a0 36 23 02 03 01-00 01 a3 50 30 4e 30 1d   ...6#......P0N0.
0250 - 06 03 55 1d 0e 04 16 04-14 40 8e d9 dc 45 b5 ed   ..U......@...E..
0260 - ed b9 a2 c2 27 6e e7 57-2b ab f2 1d 97 30 1f 06   ....'n.W+....0..
0270 - 03 55 1d 23 04 18 30 16-80 14 40 8e d9 dc 45 b5   .U.#..0...@...E.
0280 - ed ed b9 a2 c2 27 6e e7-57 2b ab f2 1d 97 30 0c   .....'n.W+....0.
0290 - 06 03 55 1d 13 04 05 30-03 01 01 ff 30 0d 06 09   ..U....0....0...
02a0 - 2a 86 48 86 f7 0d 01 01-0b 05 00 03 82 01 01 00   *.H.............
02b0 - 98 63 13 91 a6 8b b0 0e-ef 9a 29 04 e4 07 d3 4d   .c........)....M
02c0 - 7c a1 aa a2 a5 ac 77 62-15 73 e1 dc fe 9d b3 5e   |.....wb.s.....^
02d0 - 4f a0 66 69 d5 4b b9 d1-4c d5 cc fa 09 a5 d7 04   O.fi.K..L.......
02e0 - 8a dc 42 ca be f1 76 95-0b e5 2e 11 2a f8 58 05   ..B...v.....*.X.
02f0 - 4a 90 40 e9 74 ab 1a 8f-55 74 b9 79 76 6e 09 93   J.@.t...Ut.yvn..
0300 - b4 e1 9c 99 9d a5 da b7-30 0f a1 87 1e 2e 1a 9d   ........0.......
0310 - 5f fb 65 92 aa 27 84 eb-de ea 8e 4f db 74 d1 30   _.e..'.....O.t.0
0320 - e1 6b 10 c7 42 20 4a 74-46 ae 0c 20 98 18 2d 97   .k..B JtF.. ..-.
0330 - 9a 42 3c 43 30 d6 47 49-2a e6 b6 79 46 bb bd a8   .B<C0.GI*..yF...
0340 - 76 1b 63 81 88 25 52 20-34 5e 14 77 ee dd 97 61   v.c..%R 4^.w...a
0350 - c9 6c 0a 3f a3 b3 88 1f-70 ab 1c 13 41 79 73 2a   .l.?....p...Ays*
0360 - fe 6f a6 b3 34 5c a5 21-72 e8 4c 76 ad d6 01 c2   .o..4\.!r.Lv....
0370 - 45 85 d0 31 1b 91 25 e6-36 d3 fb 0b 8f d3 88 77   E..1..%.6......w
0380 - 01 e5 14 65 88 58 db 2a-f0 49 33 5a 7b 8c 77 2f   ...e.X.*.I3Z{.w/
0390 - ff 65 43 0c ab cb dd 8e-ae 93 17 9f 6f 55 bc 38   .eC.........oU.8
03a0 - 1b a9 6e be 18 d4 b2 b9-23 02 2b fc 69 9b d9 0d   ..n.....#.+.i...
>>> ??? [length 0005]
    16 03 03 00 04
>>> TLS 1.2 Handshake [length 0004], ServerHelloDone
    0e 00 00 00
write to 0x60006b850 [0x60006ba40] (9 bytes => 9 (0x9))
0000 - 16 03 03 00 04 0e                                 ......
0009 - <SPACES/NULS>
read from 0x60006b850 [0x60008e383] (5 bytes => 5 (0x5))
0000 - 16 03 03 01 06                                    .....
<<< ??? [length 0005]
    16 03 03 01 06
read from 0x60006b850 [0x60008e388] (262 bytes => 262 (0x106))
0000 - 10 00 01 02 01 00 3c bb-11 fe cd b6 fd f3 2a dd   ......<.......*.
0010 - ff b9 22 c7 23 a7 1d 56-48 31 ea b8 27 18 23 e3   ..".#..VH1..'.#.
0020 - 1e 88 78 e4 66 7d 42 a5-b2 ce 91 86 3c 81 00 0c   ..x.f}B.....<...
0030 - 61 b3 81 99 64 84 e8 03-68 bb 5d 3c 76 23 1a 5d   a...d...h.]<v#.]
0040 - 76 83 70 e5 84 e8 82 c1-cb 66 f7 be 76 ae 85 d9   v.p......f..v...
0050 - 18 e3 ff 2b c4 7d 86 b8-05 36 42 c1 99 74 cd 94   ...+.}...6B..t..
0060 - 7c ac 81 c7 da 71 bb cb-41 92 02 3a 7f 8b 6f 43   |....q..A..:..oC
0070 - ed a5 63 c5 6a 06 7b b2-fb ab 86 ca 1d 54 82 10   ..c.j.{......T..
0080 - 58 70 43 3d ea 65 34 a0-4d 21 4c 78 fc 47 90 dc   XpC=.e4.M!Lx.G..
0090 - 6e 37 ed fd ad 5c bf c5-b1 06 72 1b b0 e2 36 a7   n7...\....r...6.
00a0 - df 5a d1 b3 46 d8 a4 de-56 1e e1 0e e9 07 dd b4   .Z..F...V.......
00b0 - ad 26 6f 08 d0 7c 1e dd-12 bc c4 45 a5 c1 2b 5f   .&o..|.....E..+_
00c0 - ca 32 2e 35 46 c1 7d 84-a1 bd 3f 03 b9 28 57 5e   .2.5F.}...?..(W^
00d0 - 1a 22 a4 20 e3 e4 85 43-6e 64 85 cc 97 c4 ba de   .". ...Cnd......
00e0 - ea 54 8b 2f c8 58 33 c7-1f 23 6e df 2c 70 b8 32   .T./.X3..#n.,p.2
00f0 - 2b 54 26 96 06 68 ab 72-00 d8 dd f3 40 3a 4e 01   +T&..h.r....@:N.
0100 - 5d 11 ba 60 63 6a                                 ]..`cj
<<< TLS 1.2 Handshake [length 0106], ClientKeyExchange
```

```
    10 00 01 02 01 00 3c bb 11 fe cd b6 fd f3 2a dd
    ff b9 22 c7 23 a7 1d 56 48 31 ea b8 27 18 23 e3
    1e 88 78 e4 66 7d 42 a5 b2 ce 91 86 3c 81 00 0c
    61 b3 81 99 64 84 e8 03 68 bb 5d 3c 76 23 1a 5d
    76 83 70 e5 84 e8 82 c1 cb 66 f7 be 76 ae 85 d9
    18 e3 ff 2b c4 7d 86 b8 05 36 42 c1 99 74 cd 94
    7c ac 81 c7 da 71 bb cb 41 92 02 3a 7f 8b 6f 43
    ed a5 63 c5 6a 06 7b b2 fb ab 86 ca 1d 54 82 10
    58 70 43 3d ea 65 34 a0 4d 21 4c 78 fc 47 90 dc
    6e 37 ed fd ad 5c bf c5 b1 06 72 1b b0 e2 36 a7
    df 5a d1 b3 46 d8 a4 de 56 1e e1 0e e9 07 dd b4
    ad 26 6f 08 d0 7c 1e dd 12 bc c4 45 a5 c1 2b 5f
    ca 32 2e 35 46 c1 7d 84 a1 bd 3f 03 b9 28 57 5e
    1a 22 a4 20 e3 e4 85 43 6e 64 85 cc 97 c4 ba de
    ea 54 8b 2f c8 58 33 c7 1f 23 6e df 2c 70 b8 32
    2b 54 26 96 06 68 ab 72 00 d8 dd f3 40 3a 4e 01
    5d 11 ba 60 63 6a
read from 0x60006b850 [0x60008e383] (5 bytes => 5 (0x5))
0000 - 14 03 03 00 01                                     .....
<<< ??? [length 0005]
    14 03 03 00 01
read from 0x60006b850 [0x60008e388] (1 bytes => 1 (0x1))
0000 - 01                                                 .
<<< TLS 1.2 ChangeCipherSpec [length 0001]
    01
read from 0x60006b850 [0x60008e383] (5 bytes => 5 (0x5))
0000 - 16 03 03 00 40                                     ....@
<<< ??? [length 0005]
    16 03 03 00 40
read from 0x60006b850 [0x60008e388] (64 bytes => 64 (0x40))
0000 - 52 05 a6 f8 ea 50 02 2c-1e b6 c2 7e 91 c9 ea 33   R....P.,...~...3
0010 - 0b 83 2b a1 7c 35 4b 07-89 f2 e3 75 4b 64 52 1b   ..+.|5K....uKdR.
0020 - 2f b8 e0 12 e0 44 1d 39-0a 6f 78 22 41 85 e9 da   /....D.9.ox"A...
0030 - 0e 16 c1 47 ba 8b 79 6c-17 bd c6 3e 9a 9b 8b a3   ...G..yl...>....
<<< TLS 1.2 Handshake [length 0010], Finished
    14 00 00 0c 4b 99 1c 69 ce 03 fe 3c bf b1 52 93
>>> ??? [length 0005]
    14 03 03 00 01
>>> TLS 1.2 ChangeCipherSpec [length 0001]
    01
write to 0x60006b850 [0x60006ba40] (6 bytes => 6 (0x6))
0000 - 14 03 03 00 01 01                                  ......
>>> ??? [length 0005]
    16 03 03 00 40
>>> TLS 1.2 Handshake [length 0010], Finished
    14 00 00 0c a6 c7 e2 27 7e fc a5 90 9a 20 c5 df
write to 0x60006b850 [0x60006ba40] (69 bytes => 69 (0x45))
0000 - 16 03 03 00 40 ef a1 a7-e8 3e 0d c5 7a 2c 3d 54   ....@....>..z,=T
0010 - 57 01 ea b0 c7 47 f8 42-b2 04 f8 9e b3 12 9a 77   W....G.B.......w
0020 - 5e 0a 81 10 d1 2d ad b1-fe 1c fa 87 f1 fb 0e 1b   ^....-..........
0030 - 3b 7f 60 9e 06 df ee b7-2b 44 6c 44 9c b6 f2 19   ;.`.....+DlD....
0040 - 7e 24 a1 29 9d                                    ~$.).
CONNECTION ESTABLISHED
Protocol version: TLSv1.2
Client cipher list: AES128-SHA:SCSV
Ciphersuite: AES128-SHA
Signature Algorithms: RSA+SHA1:RSA+SHA256
No peer certificate
```

```
read from 0x60006b850 [0x60008e383] (5 bytes => 5 (0x5))
0000 - 17 03 03 00 40                                    ....@
<<< ??? [length 0005]
    17 03 03 00 40
read from 0x60006b850 [0x60008e388] (64 bytes => 64 (0x40))
0000 - 0e 16 c1 47 ba 8b 79 6c-17 bd c6 3e 9a 9b 8b a3   ...G..yl...>....
0010 - 4a 1a 4d e3 cd b8 4c 3c-7d 7b 9e 26 89 36 90 17   J.M...L<}{.&.6..
0020 - ca 80 e5 e6 e7 92 d1 de-75 00 11 37 f4 27 2f ed   ........u..7.'/.
0030 - d4 dc 11 18 95 1d d7 c1-dd 25 19 d4 fa 0e a3 0d   .........%......
>>> ??? [length 0005]
    17 03 03 00 40
write to 0x60006b850 [0x6000928d3] (69 bytes => 69 (0x45))
0000 - 17 03 03 00 40 c7 e0 3b-6c f3 fb 59 f0 58 1f 05   ....@..;l..Y.X..
0010 - 22 2e a9 98 b2 e7 5b 53-3b 91 db 4f b2 db b0 46   ".....[S;..O...F
0020 - f2 64 71 23 62 0c 1f 78-6a c1 54 cb 84 95 91 a7   .dq#b..xj.T.....
0030 - e1 bf e8 54 b7 88 54 3a-fd d4 2e b8 d4 74 56 df   ...T..T:.....tV.
0040 - 95 61 0c 54 ac                                    .a.T.
read from 0x60006b850 [0x60008e383] (5 bytes => 5 (0x5))
0000 - 15 03 03 00 30                                    ....0
<<< ??? [length 0005]
    15 03 03 00 30
read from 0x60006b850 [0x60008e388] (48 bytes => 48 (0x30))
0000 - d4 dc 11 18 95 1d d7 c1-dd 25 19 d4 fa 0e a3 0d   .........%......
0010 - f6 59 fe 21 b8 63 32 10-84 63 f8 dc 6f 9f 9e 7a   .Y.!.c2..c..o..z
0020 - a7 db 3b 0f 5a 57 08 5c-ad ba 91 fb 88 77 76 85   ..;.ZW.\.....wv.
<<< TLS 1.2 Alert [length 0002], warning close_notify   01 00
  CONNECTION CLOSED
```

Note: *** The bad_gethostbyaddr is a benign error (in this case) caused by OpenSSL trying to read the Common Name of the certificate and doing a DNS lookup that fails. The lookup in this case fails because DNS isn't implemented for the Client in this module guide project.

## 9.2 Output from the OpenSSL TLS Client

```
C:\Users\janet\e2_studio_ssp_130\NetX_Duo_TLS_Security_MG_AP\src>openssl
s_client -connect 192.2.2.66:103 -CAfile root_ca_cert.pem -no_comp

CONNECTED(00000003)
depth=2 C = US, ST = California, L = San Diego, O = Express Logic, OU = NetX
Secure, CN = NetX Secure TLS Test Root CA
verify return:1
depth=1 C = US, ST = California, O = Express Logic, OU = NetX Secure, CN =
NetX Secure TLS Test ICA
verify return:1
depth=0 C = US, ST = California, O = Express Logic, OU = NetX Secure, CN =
192.2.2.66
verify return:1
---
Certificate chain
 0 s:/C=US/ST=California/O=Express Logic/OU=NetX Secure/CN=192.2.2.66
   i:/C=US/ST=California/O=Express Logic/OU=NetX Secure/CN=NetX Secure TLS
Test ICA
 1 s:/C=US/ST=California/O=Express Logic/OU=NetX Secure/CN=NetX Secure TLS
Test ICA
   i:/C=US/ST=California/L=San Diego/O=Express Logic/OU=NetX Secure/CN=NetX
Secure TLS Test Root CA
---
Server certificate
-----BEGIN CERTIFICATE-----
```

```
MIIDzjCCAragAwIBAgIBATANBgkqhkiG9w0BAQsFADBzMQswCQYDVQQGEwJVUzET
MBEGA1UECAwKQ2FsaWZvcm5pYTEWMBQGA1UECgwNRXhwcmVzcyBMb2dpYzEUMBIG
A1UECwwLTmV0WCBTZWN1cmUxITAfBgNVBAMMGE5ldFggU2VjdXJlIFRMUyBUZXN0
IElDQTAeFw0xODAyMjIyMzM2MDhaFw0yODAyMjAyMzM2MDhaMGUxCzAJBgNVBAYT
AlVTMRMwEQYDVQQIDApDYWxpZm9ybmlhMRYwFAYDVQQKDA1FeHByZXNzIExvZ2lj
MRQwEgYDVQQLDAtOZXRYIFNlY3VyZTETMBEGA1UEAwwKMTkyLjIuMi42NjCCASIw
DQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAMpa8i/QRGlnh5K7AmdaQjMjmFnv
yNE8zb4F3K5yXHhuolxckGYxxV3cuMiJgV1kT/rV4naWX+pPDaSgzrHH6Y2CCndU
E/d04VXJmo1yjVl5X9TI1+5n957e4Q6LLCqYuDKJ5XLBHezV1xAEB0FyzrbEIPiV
iotFc95TNMrpnld/A7slgFcSsEFpAva31VOKE4yn19kmsdaBMbq++cyTPVTOzTqa
wAWKEq/OrUE287PcNUMxW6iY0mJ7D+zY/mXe2if2eLyZ/DuwSdnJoiAPGdRqkgbx
BssOgUPp5dbKuvbeeux9i43Xj8j39XZYIPnE4BLarCwCa7JqLJXU+cwCxu0CAwEA
AaN7MHkwCQYDVR0TBAIwADAsBglghkgBhvhCAQ0EHxYdT3BlblNTTCBHZW5lcmF0
ZWQgQ2VydGlmaWNhdGUwHQYDVR0OBBYEFMUptDq/mK4HfQZHO9PeLZ/pzO6HMB8G
A1UdIwQYMBaAFFnlceWFzS1hzK0hS9/7mO2VNKKKMA0GCSqGSIb3DQEBCwUAA4IB
AQAfiX1r1Xoy1L1nzrUy5Q+OcU0n9M6UJjfLmD3nvW42vRMsL5ZeIp6v2hPiej2j
DR/5GuEH7wgzFREloqPsonQ00fzcT7RgNLN4RxTSrQZNxQN76jSQ+fjGmoMPowYS
tCjTL/kpvvUOM94GB9nHeUaKi5o7K9aqiNLHNNSmRqz+HdEZU4VhBWXUWp96uu5q
BgRaCmlR6u7V5vNaYUoRFjULJ+CzsezeRBkGxgI3HLgt4+DNBuifvNFVFlkpAUUa
k1jKIv8gZvP7CPIMriqd+hk7GJwGAKiayApm9kvfuEtxIUXEUaIfge8l3Ulsf6Cs
sitTKYzQeb9nySfogNEeKgW2
-----END CERTIFICATE-----
subject=/C=US/ST=California/O=Express Logic/OU=NetX Secure/CN=192.2.2.66
issuer=/C=US/ST=California/O=Express Logic/OU=NetX Secure/CN=NetX Secure TLS
Test ICA
---
No client certificate CA names sent
---
SSL handshake has read 2101 bytes and written 649 bytes
---
New, TLSv1/SSLv3, Cipher is AES128-SHA
Server public key is 2048 bit
Secure Renegotiation IS NOT supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : AES128-SHA
    Session-ID:
    Session-ID-ctx:
    Master-Key:
935FC90842A5FD840C456D5E7A560A5AFC09FD449012FD2CD968A0C7C4F8E5D4E7ED22A3E2A5BD
40C305AC9E94326D56
    Key-Arg   : None
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    Start Time: 1519345228
    Timeout   : 300 (sec)
    Verify return code: 0 (ok)
---
hello
HTTP/1.1 200 OK
Date: Fri, 15 Sep 2016 23:59:59 GMT
Content-Type: text/html
Content-Length: 122
```

```
<html>
<body>
<b>Hello NetX Secure User!</b>
This is a simple webpage
served up using NetX Secure!
</body>
</html>
Closed
```

## 10. NetX Duo TLS 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 Duo TLS Module Next Steps

After you have mastered a simple NetX Duo TLS Module project, you may want to review a more complex example. The NetX Duo Secure has a certificate authentication callback and a time stamp callback which were not used in the module guide project. There is also the option of using Pre-Shared-Keys in lieu of the Server Client handshake/cipher key exchange. This is not commonly used but there is sometimes a need for it.

## 12. NetX Duo TLS Module Reference Information

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

Links to all the most up-to-date NetX Duo TLS Module reference materials and resources are available on the Synergy Knowledge Base: https://en-us.knowledgebase.renesas.com/English_Content/Renesas_Synergy%E2%84%A2_Platform/Renesas_Synergy_Knowledge_Base/NetX_NetXD_AutoIP_Module_Guide_Resources. https://en-us.knowledgebase.renesas.com/English_Content/Renesas_Synergy%E2%84%A2_Platform/Renesas_Synergy_Knowledge_Base/NetX_NetXD_AutoIP_Module_Guide_Resources.

OpenSSP tutorials and resources

https://jamielinux.com/docs/openssl-certificate-authority/introduction.html

http://www.devsec.org/info/ssl-cert.html

https://www.area536.com/projects/be-your-own-certificate-authority-with-openssl/

Using OpenSSL in Windows with Cygwin https://www.cygwin.com/

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

## Revision History

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.00 | Sep 29, 2018 | — | First release document |
| | | | |

# 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.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

RENESAS

## SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.


**Renesas Electronics Corporation**
TOYOSU FORESIA, 3-2-24 Toyosu, Koto-ku, Tokyo 135-0061, Japan

**Renesas Electronics America Inc.**
1001 Murphy Ranch Road, Milpitas, CA 95035, U.S.A.
Tel: +1-408-432-8888, Fax: +1-408-434-5351

**Renesas Electronics Canada Limited**
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

**Renesas Electronics Europe Limited**
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-651-700

**Renesas Electronics Europe GmbH**
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

**Renesas Electronics (China) Co., Ltd.**
Room 1709 Quantum Plaza, No.27 ZhichunLu, Haidian District, Beijing, 100191 P. R. China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

**Renesas Electronics (Shanghai) Co., Ltd.**
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, 200333 P. R. China
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

**Renesas Electronics Hong Kong Limited**
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

**Renesas Electronics Taiwan Co., Ltd.**
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

**Renesas Electronics Singapore Pte. Ltd.**
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

**Renesas Electronics Malaysia Sdn.Bhd.**
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

**Renesas Electronics India Pvt. Ltd.**
No.777C, 100 Feet Road, HAL 2nd Stage, Indiranagar, Bangalore 560 038, India
Tel: +91-80-67208700, Fax: +91-80-67208777

**Renesas Electronics Korea Co., Ltd.**
17F, KAMCO Yangjae Tower, 262, Gangnam-daero, Gangnam-gu, Seoul, 06265 Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5338




Colophon 7.2