

# デザイン・ガイド: TIDA-010024 ネットワーク処理能力を強化したセキュアな 6LoWPAN メッシュ・ エンド・ノードのリファレンス・デザイン



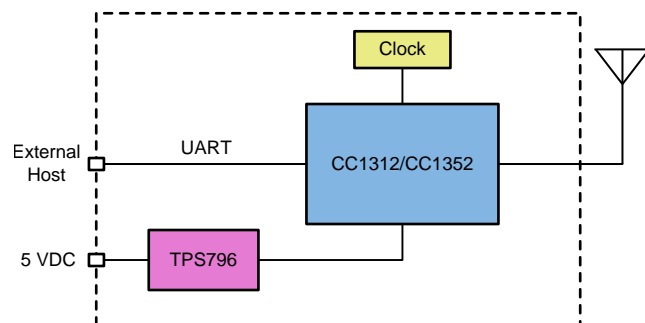
## 概要

このリファレンス・デザインでは、データグラム・トランスポート・レイヤ・セキュリティ(DTLS)を備えた、スマート・メータの高度計量インフラストラクチャ(AMI)ネットワーク用の無線周波数(RF)メッシュ・ネットワークの終端ノードを実装します。このネットワークは、低消費電力ワイヤレス・パーソナル・エリア・ネットワーク上のIPv6 (6LoWPAN)ソリューションです。このデザインでは、このネットワークを単一のCC1312R SimpleLink™ワイヤレスMCUに実装し、パフォーマンスの向上とシステム・コストの最小化を実現します。完全な6LoWPANメッシュ・ネットワークが、IEEE802.15.4e/gプロトコルをベースとするTI-15.4スタック上で動作し、USCH (Un-Slotted Channel Hopping)モードを実装して、ネットワーク干渉に対する保護を実現します。より大きなネイバー・テーブルおよびルーティング・テーブルを使用し、パケットのルーティング決定を最適化することにより、TIDA-010003の終端ノード・デザインと比べてネットワーク容量が増大し、セキュリティが強化されます。

## リソース

<a href="#">TIDA-010024</a>	デザイン・フォルダ
<a href="#">CC1312R</a>	プロダクト・フォルダ
<a href="#">TM4C1294NCPDT</a>	プロダクト・フォルダ
<a href="#">LM4040</a>	プロダクト・フォルダ
<a href="#">TPS796</a>	プロダクト・フォルダ
<a href="#">SN74AVC4T245</a>	プロダクト・フォルダ

[E2E™エキスパートに質問](#)



## 特長

- 1GHz未満のISM帯域における、低消費電力RF上のIPv6ネットワーク
- 6LoWPAN、RPL、IPv6、ICMPv6、UDP、DTLS の IP メッシュ・ネットワーク・プロトコルを実装
- IEEE 802.15.4eベースの周波数ホッピング機能とMAC データ暗号化機能を搭載したTI-15.4スタックを実装
- ソフトウェア層のアーキテクチャはWi-SUN FAN v1.0 と同一
- TIDA-010003の単純な6LoWPANメッシュの終端ノードに、DTLSを追加し、ネイバーおよびルーティング・テーブルを大型化することで、ネットワーク性能を向上したりリファレンス・デザイン
- バッテリ駆動のエンド・ノード用に低消費電力モード構成をサポート

## アプリケーション

- [ワイヤレス通信](#)
- [電気メーター](#)
- [水道メーター](#)
- [ガス・メーター](#)





使用許可、知的財産、その他免責事項は、最終ページにあるIMPORTANT NOTICE (重要な注意事項)をご参照くださいますようお願いいたします。

## 1 System Description

This reference design provides a low-power RF mesh network end node that supports 6LoWPAN mesh protocols. A primary design goal is to add Datagram Transport Layer Security (DTLS) support plus improve network performance with larger neighbor and routing tables compared to the [TIDA-010003](#) end-node reference design. This design retains the Frequency Hopping (FH) techniques that increase robustness in noisy Radio Frequency (RF) environments.

FH is a technique of transmitting data by switching one of many channels where the channel is selected by a pseudo-random sequence known to both sender and receiver. This technique is known as robust versus interference and excellent in coexistence performance. [3.2](#) shows experimental results to measure and verify network performance in different scenarios. These scenarios are used to reproduce real life use cases and showcase how the system will perform in these special cases. The testings also show the negligible impact of DTLS on the overall performance. In addition, [表 2](#) summarizes key system performance.

Another segment of this design is the 6LoWPAN mesh stacks, which improves network coverage and supports IPv6-based applications. The increased network coverage reduces the total system cost by reducing the number of data collectors that are typically more expensive than smart meters. The smart meters are static in the Advanced Metering Infrastructure (AMI) networks. The mesh networking addresses the connectivity issue through multi-hop transmissions when data collectors and smart meters are not reachable with each other.

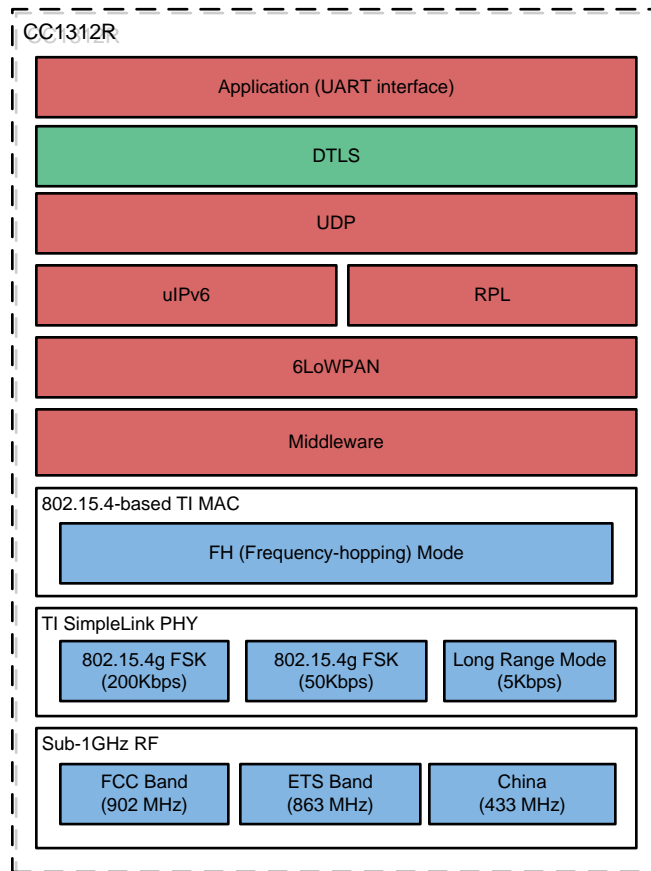
This design is based upon a SimpleLink™ MCU of the CC13x2 wireless MCU. [図 1](#) shows the overall system architecture. The TI-15.4 Medium Access Control (MAC) supports FH mode. The TI SimpleLink PHY supports 200-kbps and 50-kbps frequency-shift keying (FSK) mode, and 5-kbps long range mode. The sub-1 GHz RF on the CC1312R MCU can support three frequency bands: 902 MHz, 863 MHz, and 433 MHz.

This design is an evolution of the [TIDA-010003](#) Simple 6LoWPAN End-Node Improves Network Performance Reference Design. [表 1](#) summarizes the key differences in terms of system features between the two reference designs.

**表 1. System Features of TIDA-010024 and TIDA-010003**

	TIDA-010024	TIDA-010003
Wireless MCU	CC1312R	CC1310
Max neighbor entries	100	10
Max route entries	<ul style="list-style-type: none"> <li>• 200 for root node</li> <li>• 100 for end node</li> </ul>	<ul style="list-style-type: none"> <li>• 10 for end node</li> </ul>
Security	<ul style="list-style-type: none"> <li>• DTLS</li> <li>• IEEE 802.15.4 MAC security</li> </ul>	IEEE 802.15.4 MAC security
Max number of hop	64	64
Total flash size	352 KB	128 KB
Remaining flash memory	255 KB (debug_poll CCS configuration)	4 KB
Total RAM size	80 KB (+8 KB Cache)	20 KB (+8 KB Cache)
Remaining RAM	41 KB (debug_poll CCS configuration)	5 KB

図 1. TIDA-010024 System Architecture



## 1.1 Key System Specifications

表 2. Key System Specifications

PARAMETER	SPECIFICATIONS	DETAILS
Maximum number of hops	<ul style="list-style-type: none"> <li>64 hops (in software, configurable)</li> <li>Tested up to 6-hop networks</li> </ul>	
Maximum number of route entries	<ul style="list-style-type: none"> <li>200 for root node</li> <li>100 for end node</li> <li>Tested up to 100-node networks</li> </ul>	
Maximum number of neighbor nodes	<ul style="list-style-type: none"> <li>100 for end node and route node</li> <li>Tested up to 100-node networks</li> </ul>	
Maximum application data size	<ul style="list-style-type: none"> <li>200 B (in software, configurable)</li> <li>Tested up to 200 B</li> </ul>	
Delivery ratio	<ul style="list-style-type: none"> <li>99.42% (average in 6-hop linear topology with DTLS)</li> <li>97.14 % (average with 100 nodes network)</li> </ul>	
Round-trip time (RTT)	<ul style="list-style-type: none"> <li>0.445 second (100B over 1-hop node with DTLS)</li> <li>2.29 second (100B over 6-hop node with DTLS)</li> </ul>	
MAC data encryption	<ul style="list-style-type: none"> <li>IEEE 802.15.4-based encryption supported</li> <li>MIC-32, MIC-64, MIC-128</li> <li>ENC, ENC-MIC-32, ENC-MIC-64, and ENC-MIC-128</li> </ul>	<a href="#">3.1.2.2.1.3</a>
DTLS encryption	<ul style="list-style-type: none"> <li>DTLS v1.2</li> <li>MAC-level DTLS</li> </ul>	The MAC-level DTLS was activated and verified in this design.
Memory usage	<ul style="list-style-type: none"> <li>End node (debug_poll configuration):               <ul style="list-style-type: none"> <li>Flash: 255 KB free</li> <li>RAM: 41 KB free</li> </ul> </li> <li>Root node (debug_root_poll configuration):               <ul style="list-style-type: none"> <li>Flash: 248 KB free</li> <li>RAM: 34 KB free</li> </ul> </li> </ul>	

## 2 System Overview

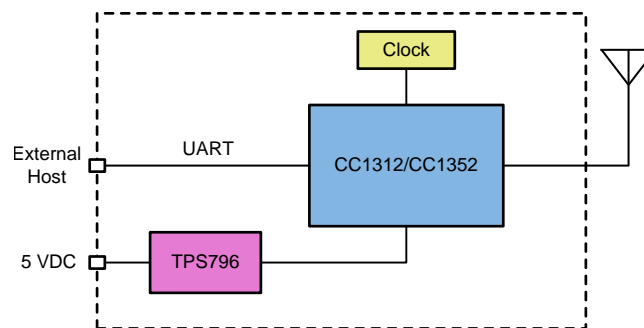
### 2.1 Block Diagram

Figure 2 shows the system block diagram. The CC1312R (or CC1352R) MCU is the 6LoWPAN mesh MCU that runs UDP applications, 6LoWPAN mesh network, and the TI 15.4-Stack over sub-1 GHz RF.

The external DC/DC converter, as shown in Figure 2, is needed when the external power source supplies the voltage level other than 3.3 V. In this reference design, because the evaluation modules (EVMs) are powered by USB, the TPS796 was chosen to convert 5 V to 3.3 V.

For end-equipment designs, the selection of the power supply depends on the input/output voltage level and required current consumption. The [TI WEBENCH Power Designer](#) provides the detailed design of the power supply with the given input requirements.

Figure 2. TIDA-010024 Block Diagram



### 2.2 Design Considerations

For this reference design, these devices perform the following:

- The CC1312R wireless MCU combines an Arm® Cortex®-M3 MCU with a flexible, ultra-low-power RF transceiver with excellent RX sensitivity to provide a robust link budget and execute the TI 15.4-Stack.
- The TPS796 low-power linear regulator offers high power-supply rejection ratio (PSRR), ultra-low noise, fast start-up, and excellent line and load transient responses.

### 2.3 Highlighted Products

#### 2.3.1 CC1312R

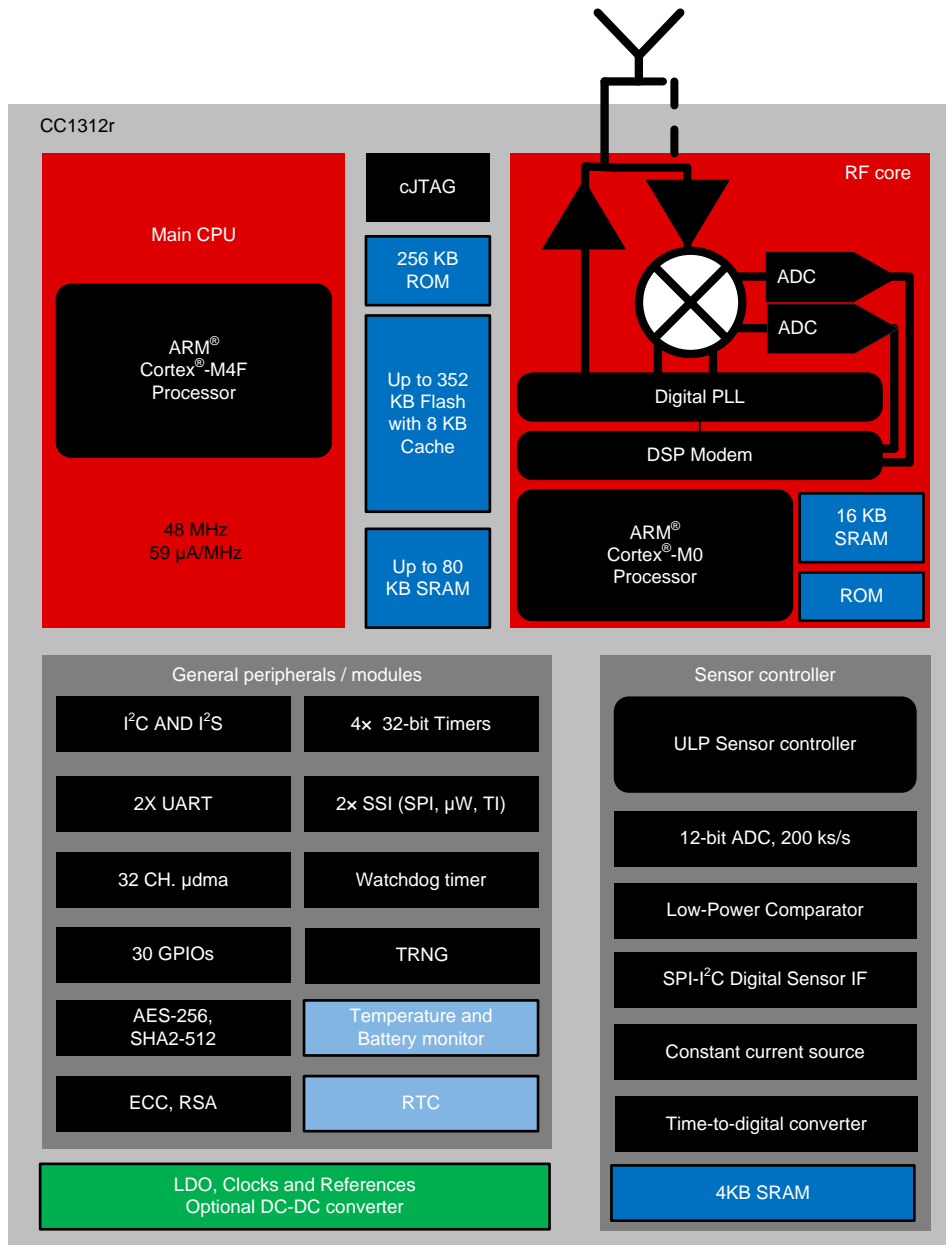
The CC1312R wireless MCU is a member of the SimpleLink MCU platform. Its very low current consumption in both active and standby mode provides excellent lifetime when operating from batteries or super capacitors.

The CC1312R combines a flexible, very low-power RF transceiver with a powerful 48-MHz Cortex-M3 MCU in a platform supporting multiple physical layers and RF standards. A dedicated radio controller (Cortex-M0) handles low-level RF protocol commands that are stored in ROM or RAM, thus ensuring ultra-low power and flexibility. Its RF subsystem offers an excellent link budget with receiver sensitivity with  $-110$  dBm at 50 kbps and output power up to  $+14$  dBm. The CC1312R is a highly integrated, true single-

chip solution incorporating a complete RF system and an on-chip DC/DC converter. The CC1312R wireless MCU is supported by the SimpleLink Software Development Kit (SDK) that offers 100% application code compatibility across the entire SimpleLink MCU portfolio and includes the integrated TI-RTOS, complete peripheral driver libraries with POSIX-compatible APIs, and encryption-enabled security features.

図 3 shows the CC1312R functional block diagram.

図 3. CC1312R Functional Block Diagram







### 3 Hardware, Software, Testing Requirements, and Test Results

#### 3.1 Required Hardware and Software

##### 3.1.1 Hardware

This reference design is built with a standard TI EVM of [LAUNCHXL-CC1312](#), as shown in [Figure 6](#).

**Figure 6. TIDA-010024 Hardware Platform**



### 3.1.2 Software

#### 3.1.2.1 Getting Started

The 6LoWPAN mesh software examples were implemented based on the TI-15.4 sensor example from SimpleLink™ CC13x2 and CC26x2 SDK v2.40. The software examples provided with the reference design run on the CC1312R MCU to support 6LoWPAN, RPL routing, IPv6, ICMPv6, UDP, DTLS and simple applications. The pre-requisite tools to build the software example are Code Composer Studio™ v8.0 or above ([CCSTUDIO](#)) and SimpleLink™ CC13x2 and CC26x2 SDK v2.40 ([SIMPLELINK-CC13X2-26X2-SDK](#)).

表 3 shows the summary of software example with CCS build configuration options.

表 3. CCS Build Configurations

BUILD CONFIGURATION	EXAMPLE	ROLE	DATA ENCRYPTION
debug_poll	UDP Poll	End node	IEEE 802.15.4 MAC encryption
debug_push	UDP Push	End node	IEEE 802.15.4 MAC encryption
debug_poll_dtls	UDP Poll	End node	MAC-level DTLS +IEEE 802.15.4 MAC encryption
debug_poll_demo	UDP Poll	End node (demo with <a href="#">TIDA-010032</a> )	IEEE 802.15.4 MAC encryption
debug_push_leaf	UDP Push	End node ( <b>Low-power mode</b> )	IEEE 802.15.4 MAC encryption
debug_root_poll	UDP Poll	Root node	IEEE 802.15.4 MAC encryption
debug_root_push	UDP Push	Root node	IEEE 802.15.4 MAC encryption
debug_root_poll_dtls	UDP Poll	Root node	MAC-level DTLS + IEEE 802.15.4 MAC encryption

For the UDP poll example, end nodes send UDP data only when they receive the poll message from the root node. This example is popular in dense networks because this approach can control network traffic effectively regardless of the network size with the cost of polling overheads. For the UDP push example, end nodes send UDP data whenever they have data to send. Compared to the UDP poll-based approach, this technique reduces the polling overhead while it increases collision probability among transmissions of the end nodes in a dense network.

The UDP poll and push examples improve the level of security with the DTLS data encryption. The debug\_poll\_dtls build configuration provides an example of UDP poll example with the DTLS encryption. They will work the exact same way in terms of mesh stacks as UDP poll and push examples but with additional MAC-level DTLS encryption.

This design also allows the CC13x2 to be used as the root node. The root node in the UDP poll mode requests data from the nodes and sends the statistics of delivery ratio and round trip time (RTT) through a serial terminal. In the UDP push mode, the root node receives the data from the connected nodes in the network and sends the statistics through a serial terminal.

The leaf node configuration (debug\_push\_leaf) was designed to provide a low power operation mode for battery-operated devices. To achieve low power consumption, this mode disables routing capability, turns RX off when idle and runs on top of the TI 15.4-Stack sleep mode. An advantage is to extend end-nodes' coverage with the built-in mesh network while keeping power consumption low. A use case will be battery-powered flow meters or in-home display connected to an electricity meter acting as a router node.

表 4 summarizes node configurations supported by this design.

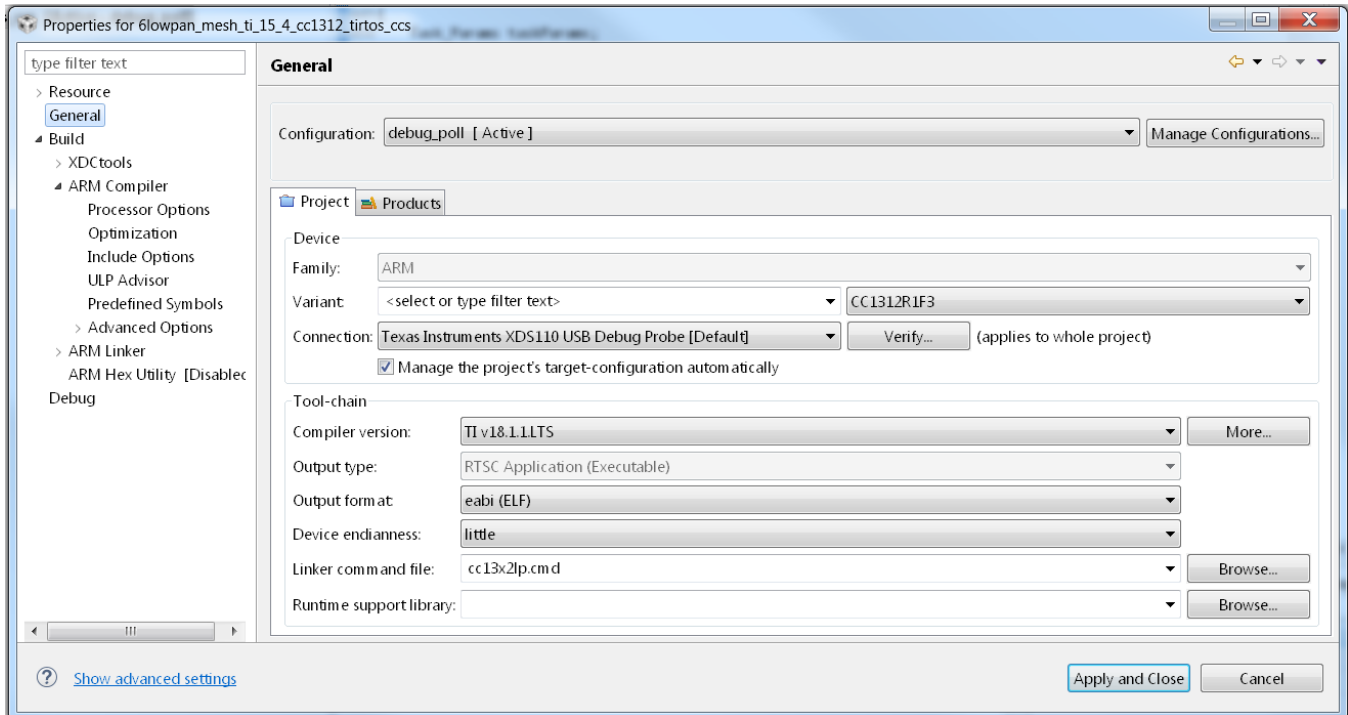
表 4. Node Configurations

Feature	ROOT	INTERMEDIATE	LEAF
Router Capability	Yes	Yes	No
RX ON when Idle	Yes	Yes	No
TI 15.4-Stack	FH non-sleep mode	FH non-sleep mode	FH sleep mode
Build Configuration	<ul style="list-style-type: none"> <li>• debug_root_poll</li> <li>• debug_root_push</li> <li>• debug_root_poll_dtls</li> </ul>	<ul style="list-style-type: none"> <li>• debug_poll</li> <li>• debug_push</li> <li>• debug_poll_dtls</li> <li>• debug_poll</li> </ul>	<ul style="list-style-type: none"> <li>• debug_push_leaf</li> </ul>

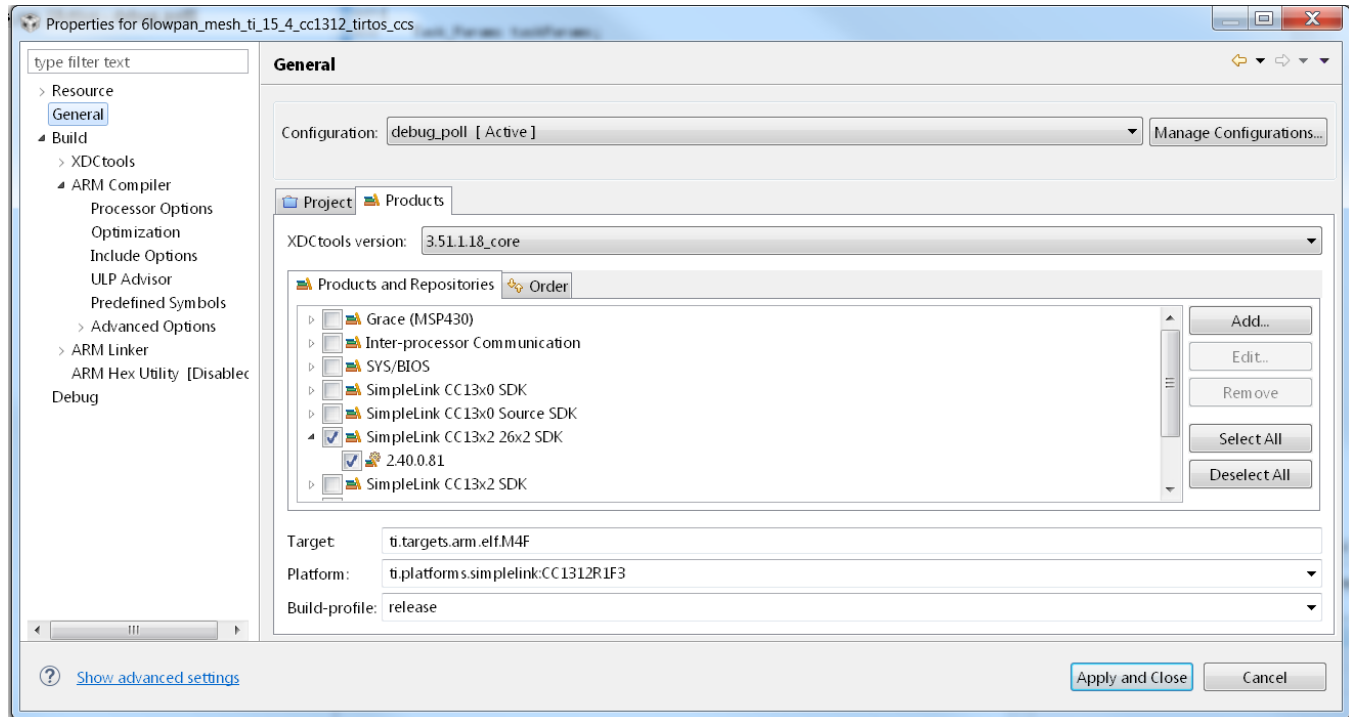
The [TIDA-01547](#), [TIDA-010003](#), and [TIDA-010032](#) are companion reference designs that work with this design. These designs provide multiple options in terms of system performance or cost to build a complete 6LoWPAN mesh network solution.

☒ 7 and ☒ 8 show Code Composer Studio screen captures to show the compiler, XDC tool and SimpleLink SDK versions for the software example.

☒ 7. Code Composer Studio™ Compiler Version



### 8. Code Composer Studio™ XDC Tool and SDK Versions



To implement the 6LoWPAN mesh design on the CC1352R wireless MCU, the same changes can be made with the CC1352R TI-15.4 sensor example from SimpleLink™ CC13x2 and CC26x2 SDK v2.40. The step-by-step procedure is given below:

1. Import CC1352R TI-15.4 sensor example from SimpleLink™ CC13x2 and CC26x2 SDK v2.40
2. Rename CCS project with your project
3. Copy and paste the entire directories of 6lowpan, Application and dtls into the project directory of the CC1352R TI-15.4 sensor example
4. Overwrite app.cfg, cc13x2lp.cmd and ccfg.c
5. Create build configurations and update the CCS property (Predefined symbols and Include options) based on the CC1312R-based 6LoWPAN mesh example project

### 3.1.2.2 6LoWPAN Mesh TI 15.4 Example

This section starts with a software overview followed by details of the software architecture and useful tips for debugging and optimizing the software.

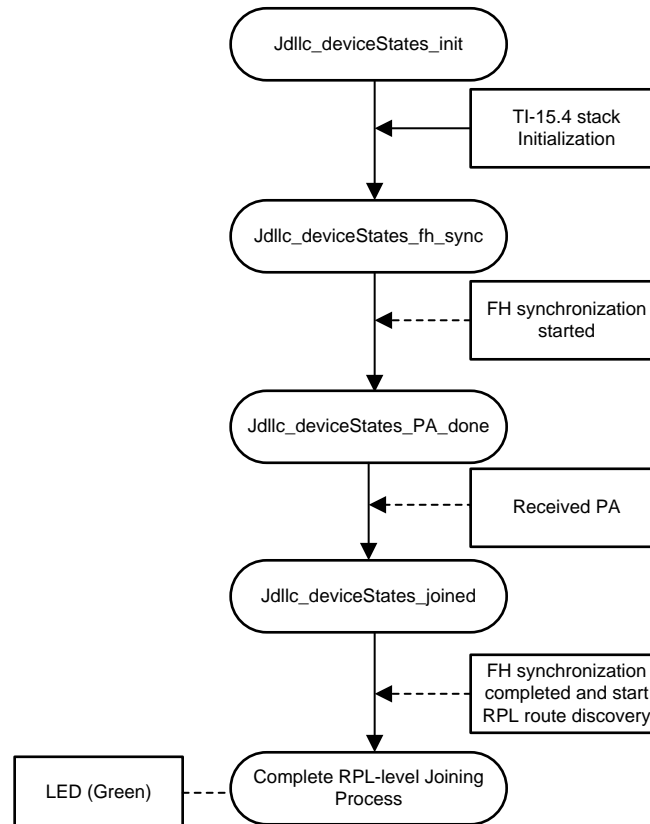
注: This reference design provides an open-source based working example that can be a baseline software to develop end-products. The software example is not optimized in RAM or Flash usage and does not guarantee product-level quality.

#### 3.1.2.2.1 Example Overview

This reference design implements a 6LoWPAN mesh network system working with the FH MAC over sub-1 GHz RF. The 6LoWPAN mesh network stacks run on TI-RTOS, which are implemented based on CONTIKI open source.

図 9 shows the end-node example software state machine. After power on, the end node starts with the `Jdllc_deviceStates_init` state and, after completing the initialization, goes to the `Jdllc_deviceStates_fh_sync` state where the node starts FH synchronization, which consists of two steps, discussed in 3.1.2.2.1.1. Once the node completes the first step of the FH synchronization, the state moves to the `Jdllc_deviceStates_PA_done` state. After the end node completes the FH synchronization, it moves to the `Jdllc_deviceStates_joined` state, which is a ready state to start RPL route discovery. The green LED (DIO7) on the EVM indicates that the node has completed the RPL-level joining process. The state machine transition has been implemented in `/Application/middleware.c`.

図 9. Software State Machine



### 3.1.2.2.1.1 FH Synchronization

FH synchronization is required to discover the FH network and to synchronize the FH timing and schedule. The underlying TI 15.4-Stack adopts WI-SUN FAN v1.0-based mechanism using four command frames:

- PAN advertisement (PA)
- PAN advertisement solicit (PAS)
- PAN configuration (PC)
- PAN configuration solicit (PCS)

FH synchronization starts with discovering neighbors, or candidates of tracking parents, by performing active scan. The end nodes start with sending PAS commands at the time chosen by the trickle algorithm (RFC 6206). The PAS is sent over all the FH channels from the lowest channel number to the highest in sequence as the nodes do not know the FH timing and schedule at this time. As a response to the PAS, the PA is sent by the nodes that has already joined to the FH network. When the end nodes receive multiple PAs during the scan period (SCAN\_TIMEOUT\_VALUE), one will be chosen based on the link-level metric and then update unicast FH timing, schedule, and the PAN information with the tracking FH parent.

The next step is to send the PCS in the same way as the PAS. Once the nodes receive the PC as a response, they update the broadcast FH timing and schedule and the Group Transient Key (GTK) hash information. Receiving PC as a response of the PCS completes the FH synchronization process, which is ready to receive and send data at the network layers.

The FH timing and schedule correction is done with the received data packets that contain FH unicast and broadcast timing and schedule information elements (IEs).

---

注: The FH synchronization mechanism implemented in this reference design is TI proprietary and is not WI-SUN FAN standard compliant.

---

表 5 summarizes the trickle algorithm parameters used for FH synchronization. Depending on the network size, the parameters may need to be adjusted. These parameters are defined in /Application/middleware.h.

**表 5. Trickle Algorithm Parameters**

PARAMETER	VALUE	DESCRIPTION
TRICKLE_TIMEOUT_VALUE	6 seconds	Discovery trickle timer minimum timeout for PAS and PCS
TRICKLE_MAX_BACKOFF	3	Discovery trickle timer backoff exponent for PAS and PCS
SCAN_TIMEOUT_VALUE	20 seconds	Discovery trickle timer timeout for PA and PC

### 3.1.2.2.1.2 Keep-Alive Mechanism

The goal of the keep-alive mechanism is to detect FH sync loss throughout monitoring sync error conditions such as data transmission and reception failures. The keep-alive mechanism broadcasts keep-alive frames (a 10B TI proprietary message defined in the example) to the link-level neighbors periodically (KEEP\_ALIVE\_TX\_TIMEOUT\_VALUE) once end nodes join to the FH network. The keep-alive TX timer is reset when broadcast frames other than keep-alive frames are sent, which reduces the keep-alive traffic overheads to the network.

If end nodes do not receive broadcast frames from their tracking parents in series (KEEP\_ALIVE\_MAX\_ATTEMPTS), FH sync loss occurs and the end nodes move immediately to the *Jdlc\_deviceStates\_fh\_sync* state to restart the FH synchronization process.

In addition to the keep-alive transmissions, each node traces unicast transmission failures to the target parents. If TX attempts fail in series (MAXIMUM\_NUM\_DATA\_FAILURE), this indicates the FH sync loss, which results in moving to the *Jdlc\_deviceStates\_fh\_sync* state to re-start the FH synchronization process. 表 6 summarizes the keep-alive parameters. The keep-alive parameters are defined in /Application/middleware.h.

**表 6. Keep-Alive Parameters**

PARAMETER	VALUE	DESCRIPTION
KEEP_ALIVE_TX_TIMEOUT_VALUE	60 seconds	Keep-alive TX interval for parents
KEEP_ALIVE_MAX_ATTEMPTS	5 times	The maximum number of failed keep-alive RX in series to indicate FH sync loss
MAXIMUM_NUM_DATA_FAILURE	5 packets	The maximum number of TX failure to indicate FH sync loss

### 3.1.2.2.1.3 MAC Data Encryption

The MAC data encryption follows IEEE 802.15.4 standard with the security level options of MIC-32, MIC-64, MIC-128, ENC, ENC-MIC- 32, ENC-MIC-64 and ENC-MIC-128.

In the software example, the default mode is set to the security level of *ApiMac\_secLevel\_encMic32* and the key ID mode of *ApiMac\_keyIdMode\_8*, defined in the /Application/jdlc\_middleware.c. The TI 15.4-Stack supports the pre-shared key mechanism based on IEEE 802.15.4 standard. The security key should be pre-programmed in the software, and all of the nodes in the same network must share the same pre-shared key.

Because the MAC data encryption mechanism requires a node to register its neighbors that use the data encryption, it is required to have a discovery phase throughout unsecured data exchanges. The PAS and PA frames used for the FH synchronization must be sent without data encryption. In addition, the software example uses unsecured transmissions for PC, PCS, and broadcast frames to minimize a chance that some neighbors with a better route are not detected during the discovery phase.

### 3.1.2.2.1.4 DTLS Encryption

This software example uses the tinyDTLS library for the DTLS. The current version of tinyDTLS supports DTLSv1.2 with SHA256. DTLS v1.2 is based on TLS 1.2 and runs over UDP to provide end-to-end secured transport.

This design implements a TI proprietary MAC-level DTLS mechanism. This consists of using the MAC encryption mechanism with the private key generated by DTLS that runs on application and negotiating the key with a two-way handshake mechanism. The negotiation process is performed between link-level neighbors and thus each pair of neighbors shares a different private key.



### 3.1.2.2.1.5 RPL Routing

The 6LoWPAN mesh software example uses the RPL protocol for multi-hop routing. The network formation with the RPL routing is initiated by broadcasting Destination Oriented Directed Acyclic Graph (DODAG) information object (DIO) by the root node. Once child nodes receive the DIOs, they broadcast the DIOs and send back a unicast destination advertisement object (DAO) packet to the parents that provide the best route toward the root. The DIO and DAO transmission times are determined by the trickle algorithm (RFC 6206). For the RPL metric to decide the best route, the expected transmission count (ETX) is used by default. For details on RPL routing, see the RFC standard RFC 6550 or the TI training video on [Wireless Network Challenges and Solutions for a Smarter Grid IoT](#).

### 3.1.2.2.1.6 6LoWPAN

The goal of the 6LoWPAN protocol is to support the IP services by reducing the gap between IPv6 and lower stacks to serve IPv6 applications on the low-end devices typically restricted in processing power, memory, and energy. The primary tasks of the 6LoWPAN are fragmentation and reassembly, IPv6 and UDP header compression, stateless IPv6 address auto-configuration, and neighbor discovery optimization. For details of the 6LoWPAN protocol, see the RFC standards RFC 4944 and RFC 6282 or the TI training video on [Wireless Network Challenges and Solutions for a Smarter Grid IoT](#).

### 3.1.2.2.2 Software Architecture

The 6LoWPAN mesh end-node software consists of TI 15.4-Stack, middleware layer interconnecting between the TI 15.4-Stack and upper layers of 6LoWPAN mesh stacks and the application layer. The middleware layer initializes and configures the TI 15.4-Stack and processes incoming data from the TI 15.4-Stack or mesh network stacks. The 6LoWPAN mesh stacks cover 6LoWPAN, RPL, IPv6/ICMPv6 and UDP protocols. The software example provides two types of UDP applications that can be configured at compile time: UDP poll and push examples.

This design also implements DTLS data encryption. A simplified two-way DTLS handshake mechanism (MAC-level DTLS) is implemented to minimize the transaction overhead on network performance.


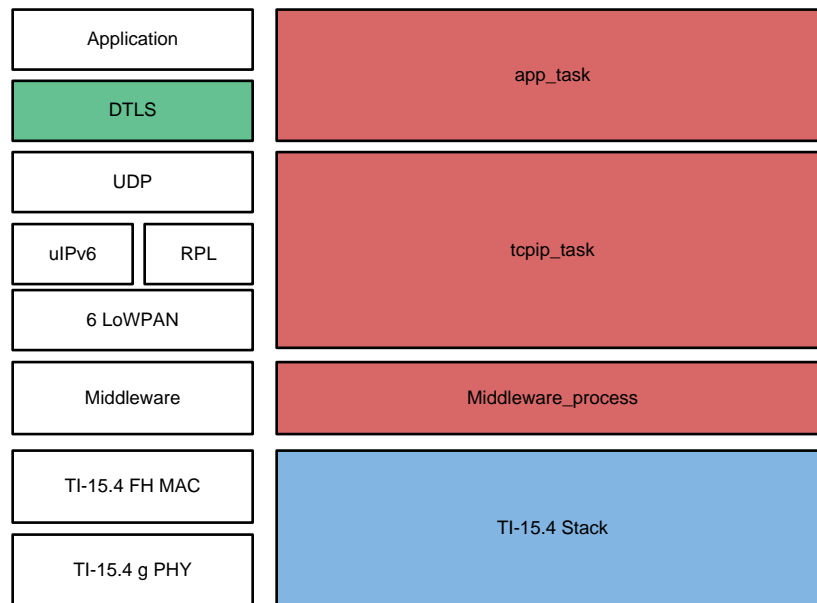
 10 shows the overall software architecture. The software example was implemented based on the sensor\_cc1312r1lp example available in the SimpleLink™ CC13x2 and CC26x2 SDK v2.40. The middleware layer runs the middleware\_process that manages the FH synchronization and keep-alive mechanism, and interfaces between TI 15.4-Stack and mesh network stacks to handle incoming data. The middleware layer follows the same software architecture using the iCall service defined in the sensor\_cc1312r1lp example. The tcpip\_task covers the mesh network layers of 6LoWPAN, uIPv6, RPL and UDP layers. If a build configuration using DTLS, the encryption/decryption layer will be added between the UDP and the application layer. The UDP poll and push examples are parts of the app\_task as the application layer.



図 10. TIDA-010024 Software Architecture



### 3.1.2.2.2.1 TI-15.4 PHY Configuration

The TI 15.4-Stack supports multiple options for the frequency band and mode to run the FH mode. The software example configures the PHY mode to APIMAC\_STD\_US\_915\_PHY\_1. The default configuration can be updated in the /Application/subg/config.h. The following codes list all the options for the PHY ID, which can be found in /Application/api\_mac.h.

注: The TI design software example was verified with the FH operation over the PHY mode of APIMAC\_STD\_US\_915\_PHY\_1.

```

/*! PHY IDs - 915MHz US Frequency band operating mode # 1 */
#define APIMAC_STD_US_915_PHY_1 1
/*! 863MHz ETSI Frequency band operating mode #1 */
#define APIMAC_STD_ETSI_863_PHY_3 3
/*! 433MHz China Frequency band operating mode #1 */
#define APIMAC_GENERIC_CHINA_433_PHY_128 128
/*! PHY IDs - 915MHz LRM US Frequency band operating mode # 1 */
#define APIMAC_GENERIC_US_LRM_915_PHY_129 129
/*! 433MHz China LRM Frequency band operating mode #1 */
#define APIMAC_GENERIC_CHINA_LRM_433_PHY_130 130
/*! 863MHz ETSI LRM Frequency band operating mode #1 */
#define APIMAC_GENERIC_ETSI_LRM_863_PHY_131 131
/*! PHY IDs - 915MHz US Frequency band operating mode # 3 */
#define APIMAC_GENERIC_US_915_PHY_132 132
/*! 863MHz ETSI Frequency band operating mode #2 */
#define APIMAC_GENERIC_ETSI_863_PHY_133 133

```

### 3.1.2.2.2.2 Middleware Layer

The middleware layer initializes the TI 15.4-Stack, maintains MAC-level keep-alive mechanism, registers security entries to the TI 15.4-Stack, processes packets incoming from the TI 15.4-Stack or upper layers, handles message timeout and erroneous transmissions, performs FH synchronization and maintains the state machine which can be one of four states: *Jdllic\_deviceStates\_init*, *Jdllic\_deviceStates\_fh\_sync*, *Jdllic\_deviceStates\_PA\_done*, and *Jdllic\_deviceStates\_joined*.

An end-node starts with the `Jdlc_deviceStates_init` state to reset the TI 15.4-Stack to configure initial MAC PIB and FH PIB values and to start the PAN as end node. The state changes to the `Jdlc_deviceStates_fh_sync` state when the node initiates the FH synchronization process. As the first step of the FH synchronization, if the node receives PA as a response of the PAS command, it moves to the `Jdlc_deviceStates_PA_done` state and proceeds to send the PCS command. After the FH synchronization completes by receiving PC commands, the node moves to the `Jdlc_deviceStates_joined` state, which is the ready state to send data at the mesh network layers.

Similar to the `sensor_cc1312r1lp` example, the communications between TI 15.4-Stack and the middleware is done through iCall messages. Conversely, the message exchanges between the mesh network stacks and the middleware layer are done through mailbox posting or direct MAC API calls. For the details on the iCall framework, refer to `ti-15.4-stack-users-guide` in `docs/ti154stack` under the installation directory of SimpleLink™ CC13x2 and CC26x2 SDK v2.40.

### 3.1.2.2.3 Network Layers (6LoWPAN, IPv6, RPL, and UDP)

The network stacks are implemented based on the CONTIKI open source. The `tcpip_task` (in `6lowpan/uipl_rpl_task.c`) processes messages incoming from the application and lower layers through the mailbox. For details of the implementations, refer to the [CONTIKI open source website](#).

### 3.1.2.2.4 DTLS Layer

The DTLS layer is used for the data encryption between the UDP layer and the Application layer. This layer is based on the tinyDTLS library. Two versions of DTLS were implemented in the software: DTLS v1.2 and MAC-Level DTLS.

The MAC-level DTLS is the mechanism implemented in the software example. With the MAC-level DTLS, a node starts with the pre-shared key encryption based on IEEE 802.15.4 MAC encryption and then MAC-level DTLS handshake negotiates the private key with its neighbor. Once the handshake completes, the node encrypts data with the private key instead of the pre-shared key.

### 3.1.2.2.5 Application Layer

The example includes two types of UDP applications: UDP poll and UDP push. The `app_task` opens the UDP socket (UDP client for the end-node) with known port numbers and starts UDP data transmissions. Depending on the UDP examples, the initiator of the UDP data is different. For the UDP poll example, the root node initiates the poll message to each node to read data. For the UDP push example, each node initiates data transmissions whenever there is application data to send.

The Device Language Message Specification (DLMS) and Companion Specification for Energy Metering (COSEM) for smart metering applications use the poll-based mechanism, and the Constraint Application Protocol (CoAP) uses the mix of UDP poll (GET command) and push (periodic OBSERVE command) mechanisms. The target end-product applications can be easily integrated on top of the given UDP application.

### 3.1.2.2.6 LED Configuration

表 7 summarizes the LED configuration in the software example. The `board_led_type_LED1` toggles when transmission or reception events happen. The `board_led_type_LED2` turns on when end nodes join to the RPL network.

**表 7. LED Configuration (CC1312R)**

NAME (PIN NUMBER)	EVENT	ACTION
board_led_type_LED1 (DIO6)	Data TX/RX	TOGGLE (Red)
board_led_type_LED2 (DIO7)	Complete RPL-level Joining	ON (Green)

### 3.1.2.2.3 Tips for Debugging and Optimization

This section provides useful tips with the Code Composer Studio tool to debug and optimize the software example in the end-product development phase.

#### 3.1.2.2.3.1 Running in Debug Mode

To debug software, run the software in debug mode. The Code Composer Studio tool provides the debug mode operation. To run in debug mode with the Code Composer Studio tool:

1. Launch the target configuration for the target device as shown in [図 11](#).
2. Connect the target EVM, load the program, and then run the software as shown in [図 12](#).
3. Add global variables to debug in the *Expressions* tab as shown in the right top of [図 12](#) to trace the variables in the debug mode.

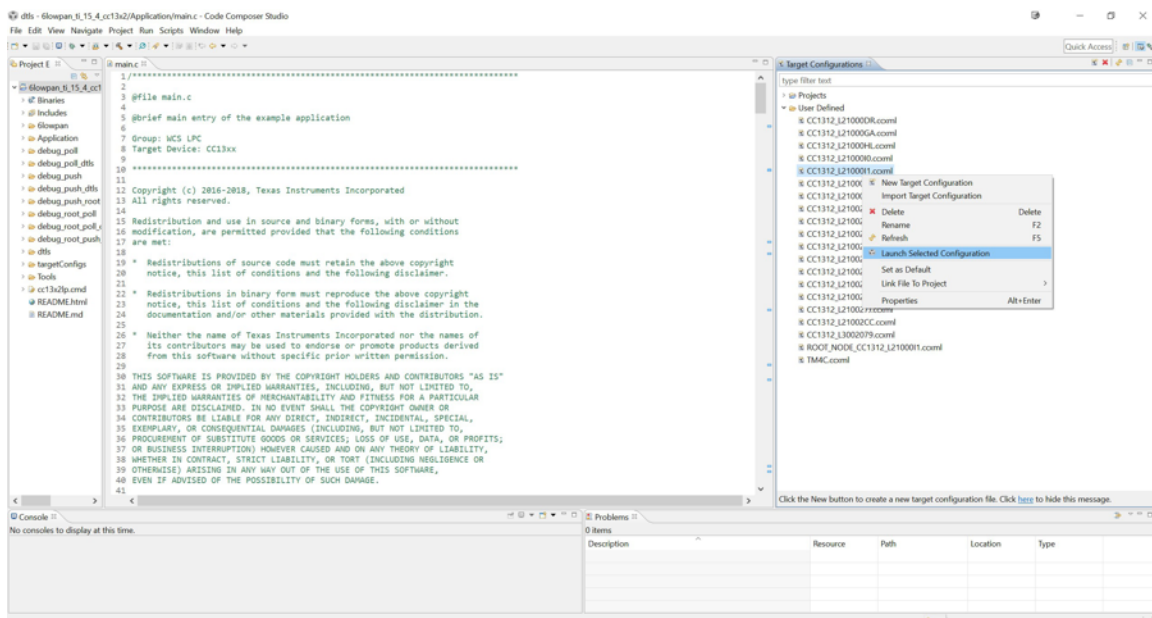
**図 11. Code Composer Studio™ Debug: Launching Debug Window**




表 8 summarizes some global variables to debug the software example.

表 8. Global Variables for Debugging

VARIABLES	DESCRIPTION
TCPIP_Dbg	Debug counts for UDP, IPv6, ICMPv6, RPL layers
LOWPAN_Dbg	Debug counts for 6LoWPAN layer
devInfoBlock	Device state machine and PAN information
mwDbg	Debug counts for middleware layer
macPib	TI-15.4 MAC PIB
FHPIB_db	TI-15.4 FH MAC PIB
joinedDODAG	Flag to indicate RPL DODAG join state
ApiMac_extAddr	8B extended MAC address

### 3.1.2.2.3.2 Sniffer Setup

A sniffer can be used to read the code with Wireshark™. This is useful for debugging purposes such as encryption verification. This sniffer will require use of an extra CC13x2R LaunchPad configured as a sniffer and connected to a computer to log the packets.

Install and set up [SmartRF Protocol Packet Sniffer 2](#) using the [SmartRF Packet Sniffer 2 User's Guide](#).

The software must be configured correctly to facilitate the packet reception. The sniffing process may present challenges because this design uses frequency hopping, and the frequency that must be sniffed often changes.. To avoid such complications, the TIDA-010024 software can be configured with the frequency hopping mode of using a single channel.

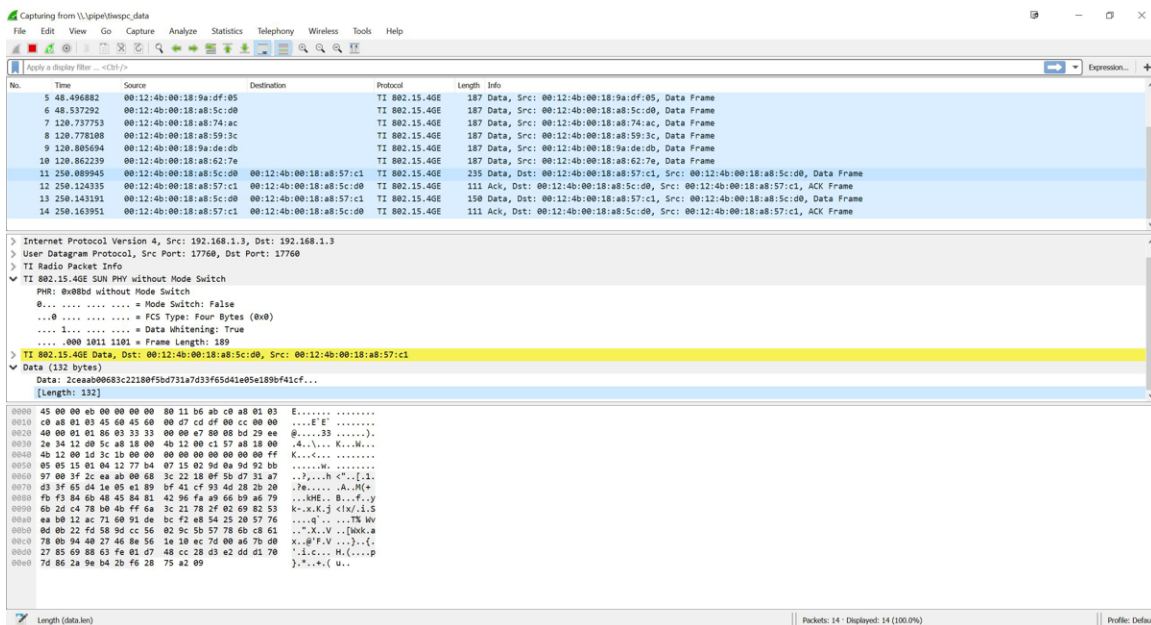
In the file Application/subg/config.h, use the following code to configure the software as one channel communication:

```

/*! Use only one channel for sniffing purpose */
#define SINGLE_FH_CHANNEL
    
```

Once the software is configured and compiled, launch Wireshark to start sniffing as shown in 図 13.

図 13. Wireshark™ Sniffer Window



### 3.1.2.2.3.3 ROV Analysis

CCS provides a useful tool to debug the software, RTOS Object View (ROV). The ROV tool helps to address various software crash issues or to optimize software examples by analyzing peak memory usage. To debug software with the ROV tool:

1. Suspend debug mode and open ROV as shown in [Figure 14](#). [Figure 15](#) shows the screen capture of ROV (on the right-bottom).
2. As an example, go to the *Detailed* and *CallStacks* taps in the *Task* menu to optimize the stack size, to trace the call stacks, or to address the stack overflow issue.

Figure 14. Code Composer Studio™ Debug: Launch ROV

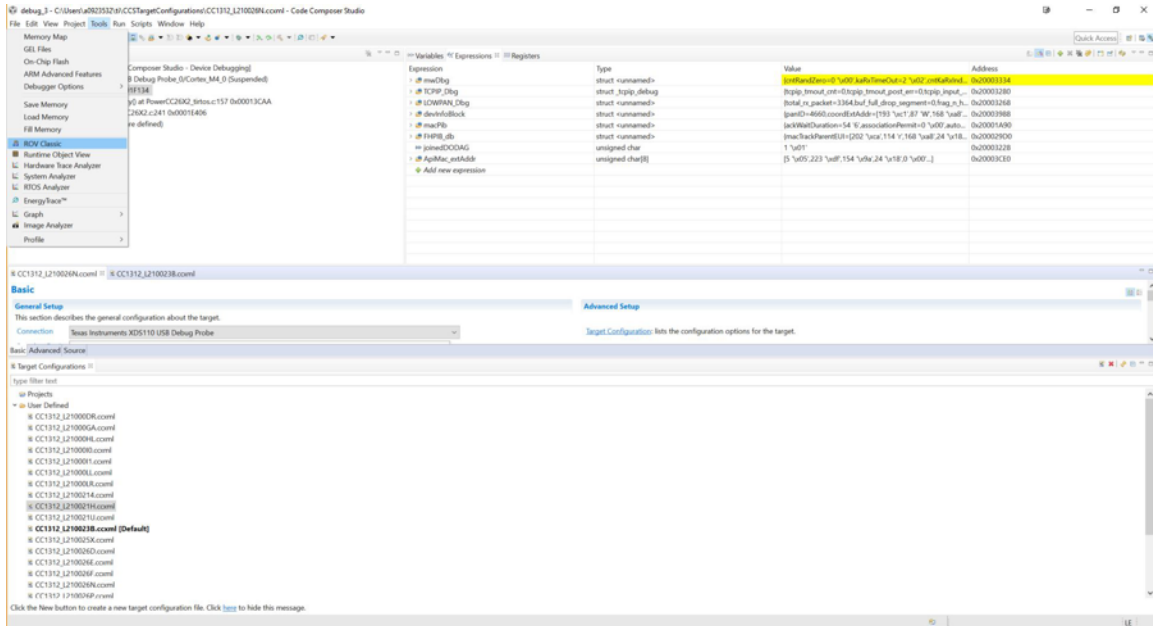
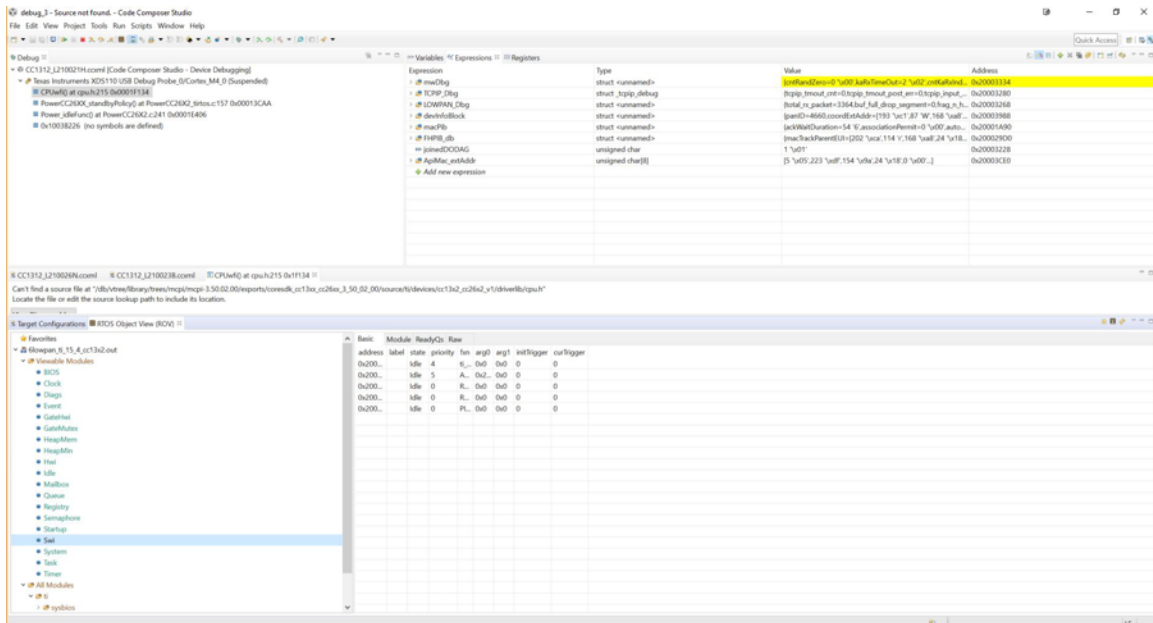


Figure 15. Code Composer Studio™ Debug: Debug Software With ROV Tool





## 3.2 Testing and Results

### 3.2.1 Test Setup

表 9 summarizes the CCS build configuration to set up the end nodes and data collector. For these experiments, the end nodes and the data collector use the CC1312R LaunchPad as shown in 3.1.1. The pre-built binaries are provided in the reference design software package.

**表 9. Firmware for Test Setup**

DEVICE	BUILD CONFIGURATION	NOTE
END NODE	<ul style="list-style-type: none"> <li>debug_poll</li> <li>debug_poll_dtls</li> </ul>	_dtls configuration was used for DTLS verification shown in 3.2
DATA COLLECTOR	<ul style="list-style-type: none"> <li>debug_root_poll</li> <li>debug_root_poll_dtls</li> </ul>	

#### 3.2.1.1 Creating Multi-hop Topology

It is challenging to create multi-hop networks in a small LAB area. For the experiments, the multi-hop topology was created with address filtering in the software. Each node has a list of entries where the node can accept the packet reception. To create a linear multi-hop topology, each node maintains two entries: one is the target parent and one is the target child.

```
#define NUM_ENTRIES 2 //Address filter to create multi-hop topology
//entry #0: target parent, entry #1: target child
//{0xBB, 0x63, 0xA4, 0x13, 0x00, 0x4B, 0x12, 0x00} (hop-1)
uint8_t whitelist[NUM_ENTRIES][8]={
{0x91, 0xC7, 0x27, 0x0A, 0x00, 0x4B, 0x12, 0x00},
{0x70, 0x63, 0xA4, 0x13, 0x00, 0x4B, 0x12, 0x00} };
```

In the software, this feature can be enabled with the "CONFIG\_MULTIHOP\_TESTING" macro defined in /Application/subg/config.h.

---

**注:** For testing setup, modify the list of entries defined in /Application/middleware.c based on the extended address of the test node. For each node, the extended address set at the initial stage is stored in the global variable of "ApiMac\_extAddr".

---

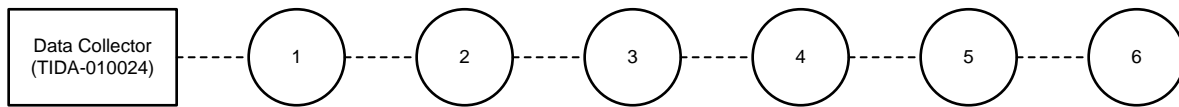
The multi-hop network with the address filtering technique has pros and cons in terms of network performance because a node can communicate with a dedicated set of nodes only while all of the nodes can be seen with each other. A disadvantage is that a node limits the path diversity over multi-hop topology while an advantage is that this setup can avoid hidden node problems.

### 3.2.2 Improved Security Performance

The goal of this experiment is to validate the security functionality implemented in the TIDA-010024 reference design. With DTLS, the TIDA-010024 offers a secured transmission mechanism of using a different private key per link.

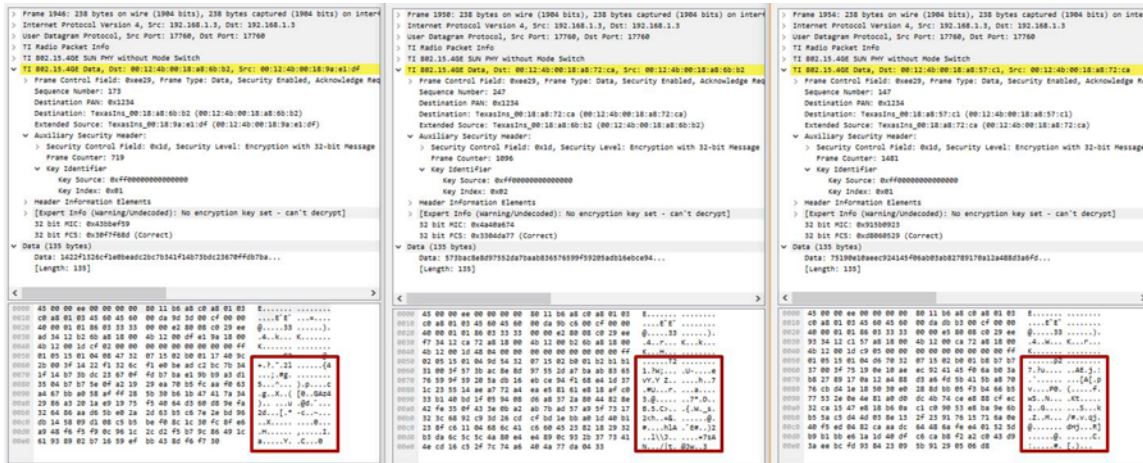
To verify if the encryption works correctly, a packet sniffer was used to read the encrypted messages. To set up the packet sniffer, follow the instructions described in 3.1.2.2.3.2. This experiment was performed with the UDP poll example and uses over a 6-hop linear topology as shown 図 16.

16. 6 Hops Linear Topology



To test the encryption, a message is sent from node 6 to the data collector by passing through every node. The sniffer shows the data transfer between each of the nodes. At each hop, the message is decrypted and then re-encrypted with the private key shared for each link-level connection. 17 shows the same message captured at different hops.

17. Encryption Functional Testing



In 17, the same data is sent between neighbors, but the value intercepted by the sniffer is different because the same data is encrypted with different keys. This mechanism makes this network more robust against tampering.

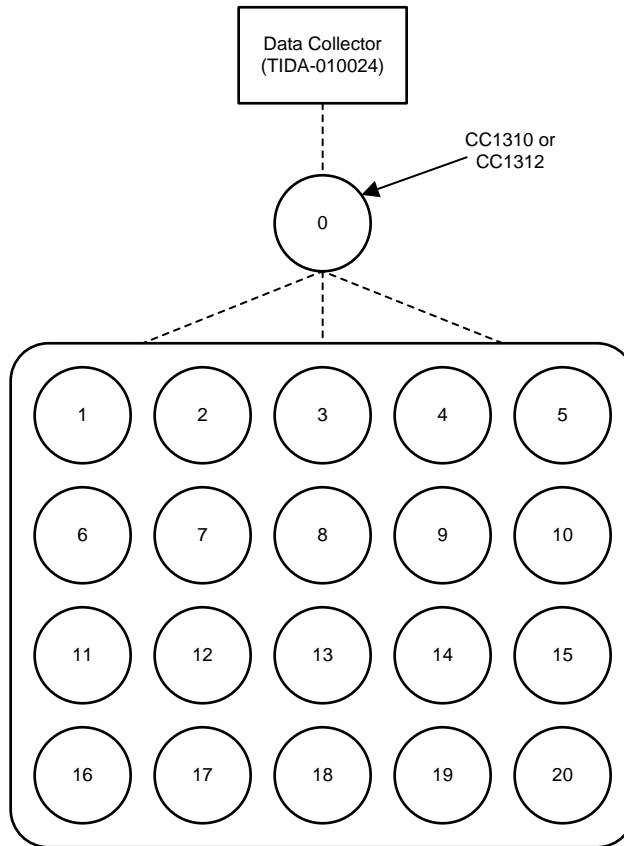
### 3.2.3 Impact of Network Capacity on Performance: An Example of the Bottleneck Scenario

The goal of this experiment is to validate the impact of increased routing capacity on the network performance in a bottleneck scenario. Compared to CC1310, the CC1312R has more memory to support bigger neighbor and routing entries.

For this experiment, the UDP poll example is used over a bottleneck scenario where all the data goes through a single router (node 0) as shown in 18. A data collector sends a 100B poll message every 5 seconds and, as a response, each node sends back the same size data packet to the data collector.



図 18. A Bottleneck Scenario



This experiment was done with the CC1310 and the CC1312 at the bottleneck location of node 0. With the CC1312R, the data collector can reach at every node without losing connections due to the increased capacity.

図 19. Delivery Ratio Performance With CC1312R Over a Bottleneck Scenario

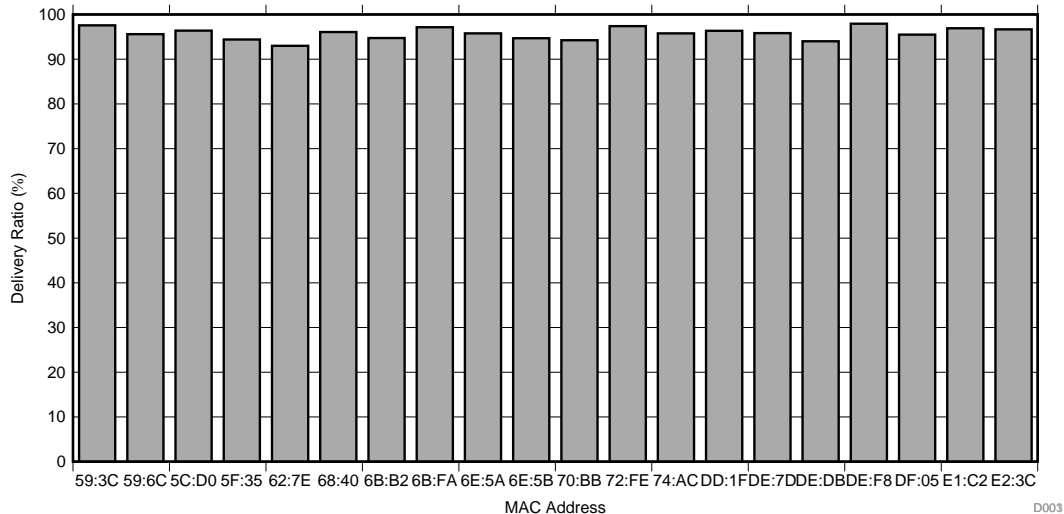


図 19 shows delivery ratio performance with CC1312R over the bottleneck scenario. The x-axis shows MAC address of each node and the y-axis is delivery ratio performance in percentage. The experimental result shows that the average delivery ratio achieves 95.75%.

The same experiment was performed by replacing CC1312R with CC1310 for node 0. The result shows that, as expected, with CC1310 the overall network performance is degraded due to network instability caused by the limited capacity.

### 3.2.4 Impact of DTLS Security on Network Performance

It is important to ensure that the DTLS overhead does not degrade the network performance. To validate the impact of overhead on system performance, the round-trip time and delivery ratio were measured with and without DTLS over a 6-hop linear topology.

図 20. Delivery Ratio Comparison With and Without DTLS

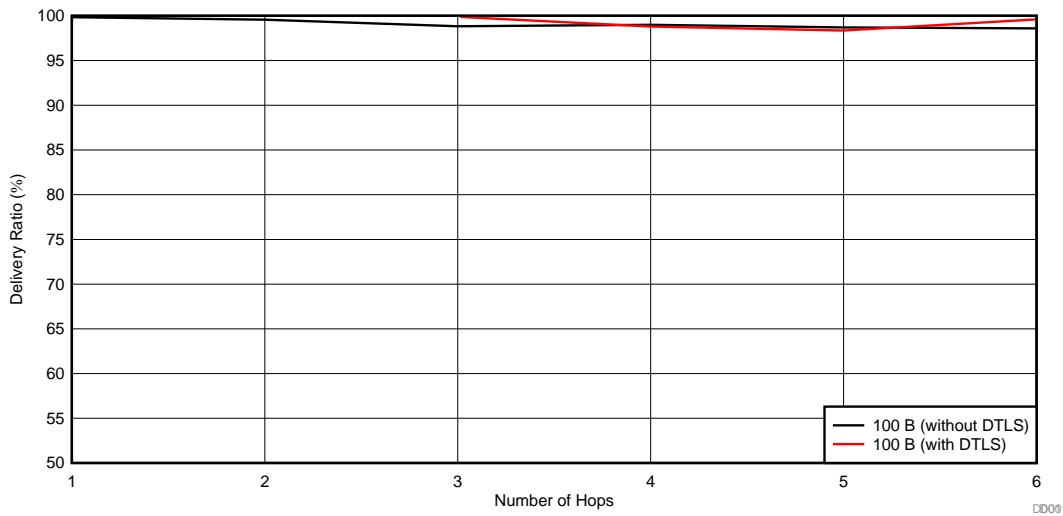
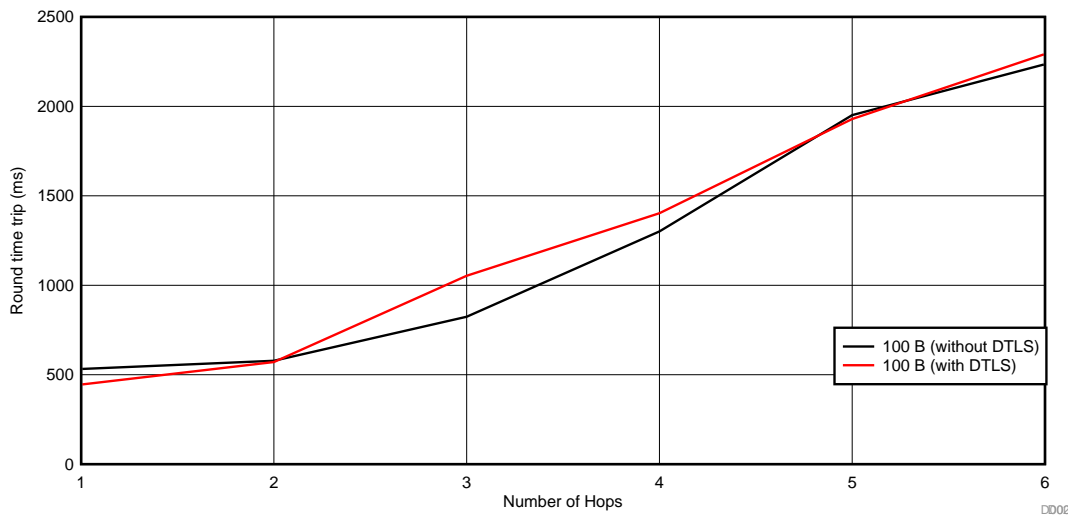


図 20 shows the delivery ratio performance with and without DTLS. The experiment ran for a day. The experimental result shows that both achieve similar delivery ratio performance regardless of the number of hops. Without DTLS, the average delivery ratio achieves 99.08% and with DTLS, the delivery ratio is 99.42%.

図 21. Round Time Trip With and Without DTLS



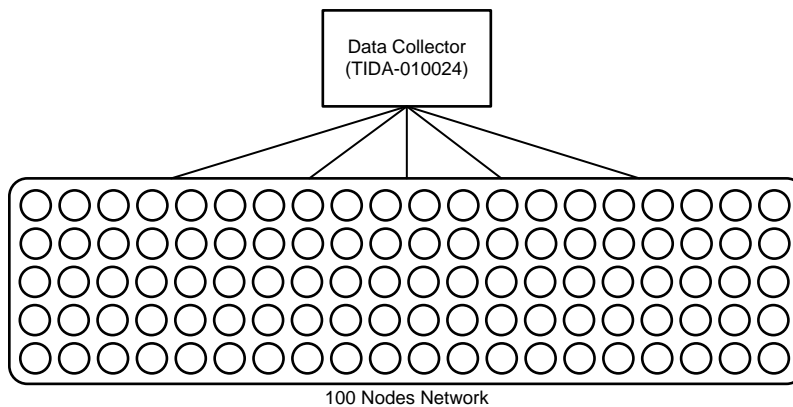
In 図 21, with the same experiment, the average round-trip time was measured. For a six-hop distance, the round-trip time performance shows 2.294 seconds with DTLS and 2.236 seconds without DTLS.

In conclusion, the experimental results show that the impact of the DTLS overhead on network performance is negligible. This is because the simplified 2-way DTLS handshaking mechanism reduces the impact of the DTLS overhead on network performance and the transaction happens one time at the beginning.

### 3.2.5 100-Node Network Testing

The goal of this experiment is to measure delivery ratio performance as well as to verify the software reliability in a large network. For the experiment, the UDP poll with DTLS example was used. The data collector polls each node every 10 seconds with 100B of data. 図 22 shows the topology of the network used for this experiment.

図 22. 100-Node Network



This experiment runs for 3 days to collect enough data. A CC1312R LaunchPad was used as data collector and the 100 LaunchPads shown in 図 23 are 50 CC1312R LaunchPads and 50 CC1352R LaunchPads.

図 23. 100-Node Network Test Setup



図 24 shows the delivery ratio performance as a function of the MAC address of each node in the 100-node setup. The x-axis shows the MAC address of each node and the y-axis shows the delivery ratio in percentage. The experimental result shows that all of the nodes achieve good delivery ratio performance. The average delivery ratio with 100 nodes is 97.14%.

図 24. Delivery Ratio Performance With 100B UDP Polling Example

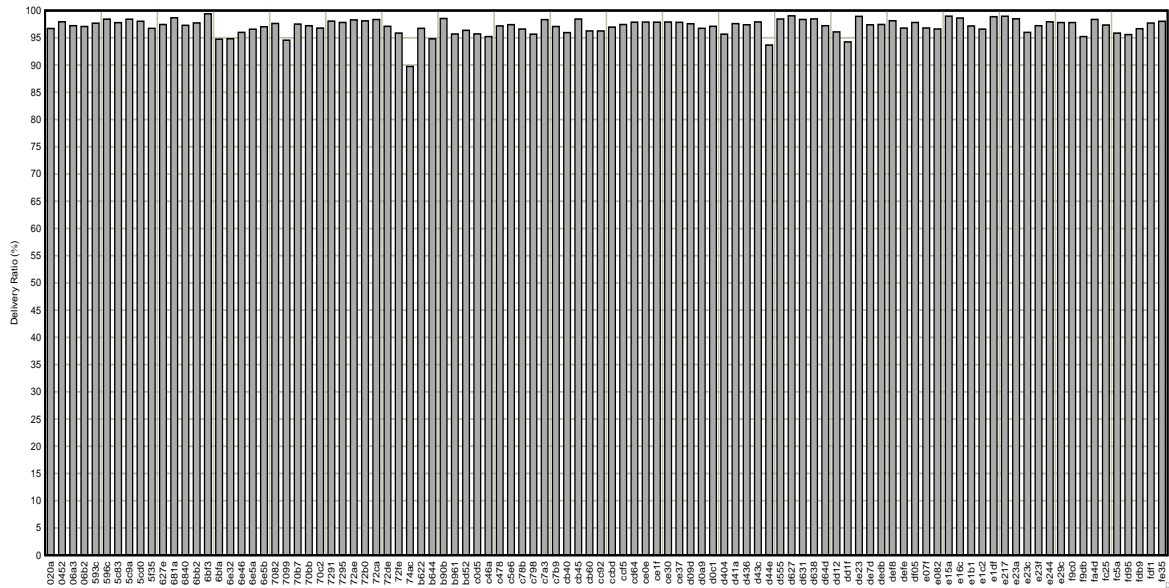


表 10 summarizes test cases to verify the software reliability and self-healing feature with the 100-node network setup. The experiments use the UDP polling example with 100B of data. The experimental results show that the 6LoWPAN mesh software works reliably without losing connections in the 100-node setup, and the keep-alive mechanism implemented in the software is functional by detecting and recovering connection losses.

表 10. Software Reliability Test Summary

TEST CASE	DESCRIPTION	PASS or FAIL
Long-run data testing (> 3 days)	Run 100B UDP polling example in the 100-node setup	Pass (All 100 nodes stayed connection to the data collector)
Self-healing testing (data collector failure)	Power-cycle the data collector to see if all the 100 nodes are reconnected to the network.	Pass
Self-healing testing (end-node failure)	Power-cycle some end-nodes to see if all the 100 nodes are reconnected to the network.	Pass

## 4 Design Files

### 4.1 Schematics

To download the schematics, see the design files at [TIDA-010024](#).

### 4.2 Bill of Materials

To download the bill of materials (BOM), see the design files at [TIDA-010024](#).

### 4.3 PCB Layout Recommendations

#### 4.3.1 Layout Prints

To download the layer plots, see the design files at [TIDA-010024](#).

### 4.4 Altium Project

To download the Altium Designer® project files, see the design files at [TIDA-010024](#).

### 4.5 Gerber Files

To download the Gerber files, see the design files at [TIDA-010024](#).

### 4.6 Assembly Drawings

To download the assembly drawings, see the design files at [TIDA-010024](#).

## 5 Software Files

To download the software files, see the design files at [TIDA-010024](#).

## 6 Related Documentation

1. Texas Instruments, [Simple 6LoWPAN Mesh End-Node Improves Network Performance Reference Design](#)
2. Texas Instruments, [Simple 6LoWPAN Mesh Data Collector Improves Network Performance Reference Design](#)
3. Texas Instruments, [Universal data concentrator reference design supporting Ethernet, 6LoWPAN RF mesh and more](#)
4. Texas Instruments, [SmartRF Packet Sniffer 2 v1.5.0 User's Guide](#)

### 6.1 商標

SimpleLink, E2E, Code Composer Studio are trademarks of Texas Instruments.  
Altium Designer is a registered trademark of Altium LLC or its affiliated companies.  
Arm, Cortex are registered trademarks of Arm Limited.  
Wireshark is a trademark of Wireshark Foundation, Inc.  
すべての商標および登録商標はそれぞれの所有者に帰属します。

### 6.2 Third-Party Products Disclaimer

TI'S PUBLICATION OF INFORMATION REGARDING THIRD-PARTY PRODUCTS OR SERVICES DOES NOT CONSTITUTE AN ENDORSEMENT REGARDING THE SUITABILITY OF SUCH PRODUCTS OR SERVICES OR A WARRANTY, REPRESENTATION OR ENDORSEMENT OF SUCH PRODUCTS OR SERVICES, EITHER ALONE OR IN COMBINATION WITH ANY TI PRODUCT OR SERVICE.

## 7 About the Author

**WONSOO KIM** is a system engineer at Texas Instruments, where he is responsible for driving grid communication system solutions, defining future requirements in TI product roadmap, and providing system-level support and training focusing on communication systems for Smart Grid customers. He received a Ph.D. degree in electrical and computer engineering from the University of Texas, Austin, Texas.

**MICKAEL CHOUTEAU** is a field application engineer at Texas Instruments, where he is responsible for technical support for industrial customers. His design experience covers both hardware design (signal chain, battery monitoring) and software design (low level drivers, RTOS, and RF protocols). He received a master degree in embedded system engineering from ECE Paris University in France.

## 改訂履歴

資料番号末尾の英字は改訂を表しています。その改訂履歴は英語版に準じています。

2018年10月発行のものから更新	Page
• 「特長」セクションに箇条書き項目を 追加 .....	1
• 「アプリケーション」セクションから「街灯」を 削除 .....	1
• 「アプリケーション」セクションに「水道メーター」および「ガス・メーター」を 追加 .....	1
• information regarding what is supported by TI SimpleLink PHY 変更 .....	2
• information and values in the System Features of TIDA-010024 and TIDA-010003 table 変更 .....	2
• the TIDA-010024 System Architecture image 変更 .....	3
• information and values in the Key System Specifications table 変更 .....	4
• CC1312 to CC1312R throughout document 変更 .....	5
• information regarding the CC1352EVM from the Hardware section 削除 .....	9
• title of the <i>Build Configurations of TIDA-010024</i> table to <i>CCS Build Configurations</i> 変更 .....	10
• information in the CCS Build Configurations table 変更 .....	10
• Code Composer Studio XDC Tool and SDK Versions image 変更 .....	12
• title of the <i>6LoWPAN_TI_15_4_Example</i> section to <i>6LoWPAN Mesh TI 15.4 Example</i> 変更 .....	13
• title of the <i>DTLS Data Encryption</i> section to <i>DTLS Encryption</i> 変更 .....	15
• information in the DTLS Encryption section 変更 .....	15
• <i>sensor_cc13x2r1p</i> to <i>sensor_cc1312r1p</i> 変更 .....	16
• <i>SimpleLink CC13x2 SDK v2.10.00.48</i> to <i>SimpleLink CC13x2 and CC26x2 SDK v2.40</i> 変更 .....	16
• information in the Firmware for Test Setup table 変更 .....	23
• title of <i>Creating Multi-hop Linear Topology</i> section to <i>Creating Multi-hop Topology</i> 変更 .....	23
• information in the Improved Security Performance section 変更 .....	23
• title of the <i>Improved Security Performance</i> section to <i>Impact of Network Capacity on Performance: An Example of the Bottleneck Scenario</i> 変更 .....	24
• information in the Impact of Network Capacity on Performance: An Example of the Bottleneck Scenario section 変更 ..	24
• title of the <i>Bottleneck Scenario With CC1312 Delivery Ratio</i> image to <i>Delivery Ratio Performance With CC1312R Over a Bottleneck Scenario</i> 変更 .....	25
• information in the Impact of DTLS Security on Network Performance section 変更 .....	26
• information in the 100-Node Network Testing section 変更 .....	27
• title of the <i>100 Nodes Network</i> image to <i>100-Node Network</i> 変更 .....	27
• title of the <i>100 Nodes Network Test Setup</i> image to <i>100-Node Network Test Setup</i> 変更 .....	27
• Wonsoo Kim to About the Author section 追加 .....	30

## 重要なお知らせと免責事項

TI は、技術データと信頼性データ(データシートを含みます)、設計リソース(リファレンス・デザインを含みます)、アプリケーションや設計に関する各種アドバイス、Web ツール、安全性情報、その他のリソースを、欠陥が存在する可能性のある「現状のまま」提供しており、商品性および特定目的に対する適合性の黙示保証、第三者の知的財産権の非侵害保証を含むいかなる保証も、明示的または黙示的にかかわらず拒否します。

これらのリソースは、TI 製品を使用する設計の経験を積んだ開発者への提供を意図したものです。(1) お客様のアプリケーションに適した TI 製品の選定、(2) お客様のアプリケーションの設計、検証、試験、(3) お客様のアプリケーションに該当する各種規格や、その他のあらゆる安全性、セキュリティ、規制、または他の要件への確実な適合に関する責任を、お客様のみが単独で負うものとし、

上記の各種リソースは、予告なく変更される可能性があります。これらのリソースは、リソースで説明されている TI 製品を使用するアプリケーションの開発の目的でのみ、TI はその使用をお客様に許諾します。これらのリソースに関して、他の目的で複製することや掲載することは禁止されています。TI や第三者の知的財産権のライセンスが付与されている訳ではありません。お客様は、これらのリソースを自身で使用した結果発生するあらゆる申し立て、損害、費用、損失、責任について、TI およびその代理人を完全に補償するものとし、TI は一切の責任を拒否します。

TI の製品は、[TI の販売条件](#)、または [ti.com](https://www.ti.com) やかかる TI 製品の関連資料などのいずれかを通じて提供する適用可能な条項の下で提供されています。TI がこれらのリソースを提供することは、適用される TI の保証または他の保証の放棄の拡大や変更を意味するものではありません。

お客様がいかなる追加条項または代替条項を提案した場合でも、TI はそれらに異議を唱え、拒否します。

郵送先住所 : Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

Copyright © 2022, Texas Instruments Incorporated