Application Note

ZigBee 3.0 Base Device - BLE Dual Mode Template

This Application Note provides example applications to demonstrate the features and operation of the Base Device in a ZigBee 3.0 network, in coexistence with a Bluetooth low energy connection that employs the NXP KW41Z wireless microcontroller. An example application can be employed as:

- **A demonstration using the supplied pre-built binaries that can be run using FRDM-KW41Z boards.**
- **A starting point for custom application development using the supplied C source files and associated project files**

The devices described in this Application Note provide the standard mandatory features of the ZigBee Base Device Behaviour Specification for a Coordinator, Router and End Device. They do not implement a complete real device, but provide the mandatory features of the ZigBee Base Device on which an application can be built and further developed.

The ZigBee 3.0 nodes of this Application Note can be used in conjunction with nodes of other ZigBee 3.0 Application Notes, available from the NXP web site.

1 Introduction

A ZigBee 3.0 wireless network comprises a number of ZigBee software devices that are implemented on hardware platforms to form nodes. This Application Note is concerned with implementing the ZigBee Base Device on the NXP KW41Z platform.

This Application Note provides example implementations of the following ZigBee logical device types:

- Coordinator
- Router
- End Device

The examples of the above device types are not real world devices, like those defined in the ZigBee Lighting & Occupancy Device Specification, but provide the basic behaviour required by the ZigBee Base Device Behaviour Specification. They are provided as templates on which to base further development into real physical devices. The ZigBee Base Device is introduced and detailed in the *ZigBee 3.0 Devices User Guide*.

The ZigBee Base Device Behaviour Specification provides definitions, procedures and methods for forming, joining and maintaining ZigBee 3.0 networks. It also defines the method for service discovery, in which a client and server of an operational cluster are bound together in order to achieve the functionality of the physical devices.

The Low Power mode is not supported for the Base Device Hybrid applications.

Note: If you are not familiar with ZigBee 3.0, you are advised to refer the *ZigBee 3.0 Stack User Guide* for a general introduction.

2 Development Environment

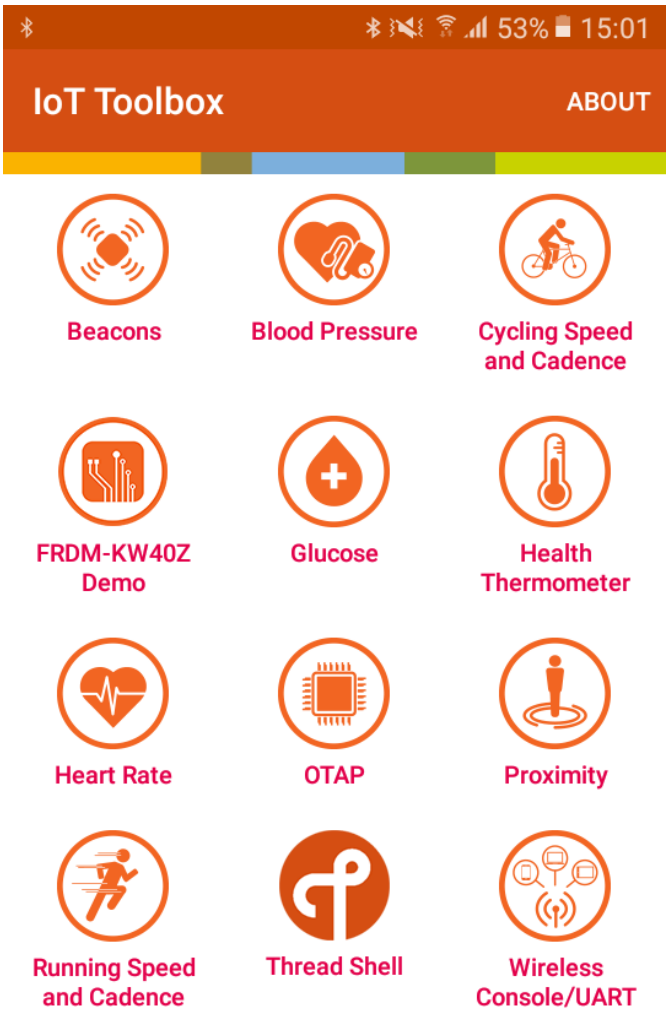
2.1 Software

In order to use this Application Note, you need to install IAR Embedded Workbench 7.80 or MCUXpresso IDE v10.1.1

All other resources are available via the ZigBee 3.0 page of the NXP web site.

The software and documentation resources referenced in this Application Note are available free-of-charge via the ZigBee 3.0 page of the NXP web site.

To demonstrate the profile functionality, the scenario requires a Bluetooth Low Energy capable central device, usually a smartphone or a tablet that runs a compatible BLE application.



IoT Toolbox

The recommended application is the IoT Toolbox, which can be installed on Apple® iOS or Android™ OS handheld devices that support Bluetooth Low Energy. The application can be found on iTunes or Google Play Store.

2.2 Hardware

Evaluation boards are available from NXP to support the development of ZigBee 3.0 applications. The FRDM-KW41Z is the platform used for running these applications.

Also, to demonstrate the Bluetooth Low Energy profile functionality, the scenario requires a Bluetooth Low Energy capable central device, as mentioned above.

3 Application Note Overview

The example applications provided in this Application Note are listed in the following table.

Application	Device Type
Hybrid Coordinator	Coordinator/Trust Centre
Hybrid Router	Router
Hybrid End Device	End Device
Hybrid Control Bridge	Control Bridge

Table 1: Example Applications and Device Types

For each application, source files and pre-built binary files are provided in the Application Note ZIP package. The pre-built binaries can be run on the FRDM-KW41Z development board.

- To load the pre-built binaries into the evaluation board components and run the demonstration application, refer to Section 4.
- To start developing you own applications based on the supplied source files, refer to Section 5.

3.1 Compatibility

The software provided with this Application Note has been tested with the following evaluation boards and SDK versions.

Product Type	Version	Supported Chips
FRDM-KW41Z boards	-	KW41Z
IAR Embedded Workbench	7.80	KW41Z
MCUXpresso	10.1.1	KW41Z
KW41 ZigBee 3.0 - SDK	6.0.6	KW41Z
KW41 Bluetooth Low Energy SDK	1.2.5	KW41Z

Table 2: Compatibility Information

4 Running the Demonstration Application

This section describes how to use the supplied pre-built binaries to run the example applications on the FRDM-KW41Z board.

4.1 Loading the Applications

The table below lists the application binary files supplied with this Application Note. These files are located in the **tools/wireless/binaries** directories for the relevant applications.

Application	Binary File	Hardware Component
Hybrid Coordinator	ble_zigbee_coordinator_frdmkw41z.bin	FRDM-KW41Z
Hybrid Router	ble_zigbee_router_frdmkw41z.bin	FRDM-KW41Z
Hybrid End Device	ble_zigbee_end_device_frdmkw41z.bin	FRDM-KW41Z
Hybrid Control Bridge	ble_zigbee_control_bridge_frdmkw41z.bin	FRDM-KW41Z

Table 3: Application Binaries and Hardware Components

A binary file can be loaded into the Flash memory of a KW41Z device using NXP Test Tool 12, available via the NXP web site.

Note: You can alternatively load a binary file into a KW41Z device using the Flash programmer built into the IDE (see Section 5).

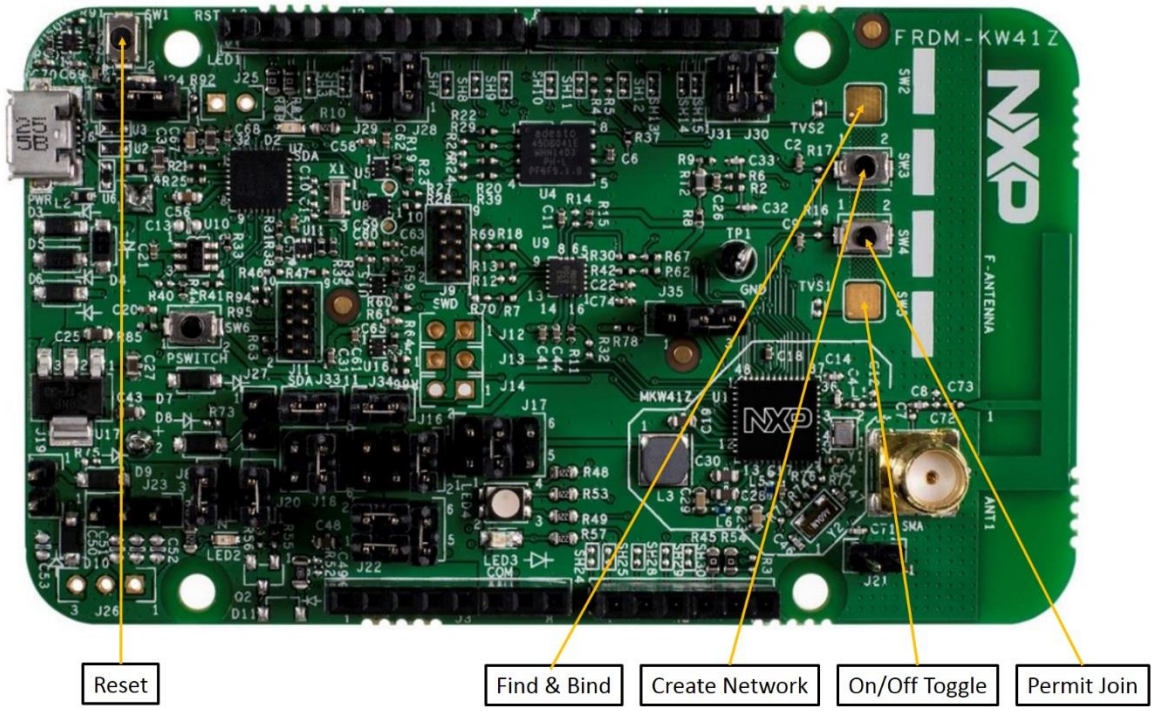
4.2 Using the Coordinator

This section describes how to commission and operate the Coordinator application in a ZigBee 3.0 network. To use this application, you must have programmed the **ble_zigbee_coordinator_frdmkw41z.bin** into the FRDM-KW41Z board, as described in Section 4.1

To incorporate the device into a ZigBee 3.0 network and then operate the device, work through Sections 4.2.2 to 4.2.5.

4.2.1 Coordinator Functionality

The functionality of the Coordinator application is described and illustrated below.



The Coordinator is responsible for initially forming the network and then, via the Trust Centre functionality, managing which other devices are allowed to join the network and distributing security materials to those that are allowed to join. The Coordinator supports the mandatory clusters and features of the Base Device as defined in the ZigBee Base Device Behaviour Specification.

For the purpose of demonstrating the ‘Finding and Binding’ functionality, the Coordinator also supports the On/Off Cluster as a client.

The Coordinator provides two methods for triggering its functionality:

- Using commands from Thread Shell – available from IoT Toolbox - connected to the Coordinator hardware. The serial interface is case-sensitive. For a summary of the serial interface, refer to Section 4.2.8.
- Using the buttons on the FRDM-KW41Z board.

The Coordinator factory default state is indicated by the FRDM-KW41Z board blinking LED3.

4.2.2 Forming a Network

A network can be formed from a factory-new Coordinator (Network Steering while not on a network) in either of the following ways:

- Press the button SW3 on the FRDM-KW41Z board.
- Enter “form” on the Thread Shell.

The Coordinator will then start a network. Using a packet sniffer, the periodic link status messages can then be observed on the operational channel.

On the FRDM-KW41Z board the LED3 solid red indicates that the Coordinator formed successfully the network.

4.2.3 Allowing Other Nodes to Join

Once a network has been formed, the network must be opened to allow other devices to join (Network Steering while on a network) in either of the following ways, depending on the hardware being used:

- Press the button SW4 on the FRDM-KW41Z board.
- Enter “steer” on the Thread Shell.

The Coordinator will then broadcast a Management Permit Join Request to the network to open the ‘permit join’ window for 180 seconds. The Network Steering process (for devices not on a network) can now be triggered on the devices that are to join the network.

4.2.4 Binding Nodes

‘Finding and Binding’ is the process whereby controlling devices find controlled devices by matching operational clusters and create entries in the Binding table. The Coordinator supports Finding and Binding as an ‘Initiator’ that tries to find targets to bind to. For the purpose of the demonstration, the Coordinator supports the On/Off Cluster as a client, so the Finding and Binding process will look for devices that support the On/Off cluster as a server in order to create bindings.

To start Finding and Binding as an Initiator, first trigger Finding and Binding on any ‘Target’ device and then do either of the following on the Coordinator (Initiator):

- Press the button SW2 on the FRDM-KW41Z board.
- Enter “find” on the Thread Shell.

When Finding and Binding for a target has completed and a binding has been created, the Coordinator will send an Identify Off command to the target device, in order to signal completion of the process for the Target. Depending on the type of bindings being created (either unicast or groupcast), an Add Group command may be sent to the target device.

Reporting is a mandatory feature in ZigBee 3.0. A device wishing to receive periodic and on-change reports from an operational server should create a remote binding for itself on the target device. This Coordinator will send a Binding Request to a target with an On/Off cluster server. It will then receive periodic and on-change reports from that device, reporting the state of the OnOff attribute (0x0000) of the On/Off cluster. The frequency of the reports depends on the default report configuration of the individual target device. The device receiving the reports can request that this is changed by sending a Report Configuration command.

4.2.5 Operating the Device

The operational functionality of this device in this demonstration is provided by the On/Off cluster. You can now send an OnOff Toggle command to the bound devices (in the Binding table) in either of the following ways:

- Press the button SW5 on the FRDM-KW41Z board.
- Enter “toggle” in the Thread Shell.

The effect that this command has on a bound device depends on the functionality related to the On/Off cluster on the device – for the Router in this demonstration, it will toggle a light (see Section 4.3).

4.2.6 Re-joining the Network

As a Coordinator, when this device is restarted in a state which is not factory-new, it will just resume operation in its previous state. All application, binding, group and network parameters are preserved in non-volatile memory.

4.2.7 Performing a Factory Reset

The Coordinator can be returned to its factory-new state (erasing all persistent data except the outgoing network frame counter) in either of the following ways, depending on the hardware being used:

- Hold down any of the buttons SW2, SW3, SW4 or SW5 for more than 8 seconds.
- Enter “factoryreset” on the Thread Shell.

  NOTE: This command will reset the MCU, and the Bluetooth connection will end.

4.2.8 Summary of Serial Interface Commands

Button	Serial Command	Action
SW2	find	Triggers Finding and Binding as an Initiator
SW3	form	Triggers network formation for a device not on a network
SW4	steer	Triggers Network Steering for a device on the network
SW5	toggle	Sends an OnOff Toggle command to bound devices
SW2-5 Very Long Press	factoryreset	Factory resets the device, erasing persistent data
Reset	reboot	Triggers a software reset (will end the Bluetooth connection)
-	print	Prints the Aps Key Table to the terminal
-	identify <addr> <src_endpoint> <dst_endpoint> <timeout>	Identify a device from a network
-	code <MAC> <Install Code>	Provisions an install code into the Aps Key Table
-	channel [<hex_value>]	Get channel mask/Set channel number
-	panid [<hex_value>]	Get / Set PAN Id
-	shortaddr [<hex_value>]	Get / Set device’s short address
-	epid	Get the extended PAN Id
-	extendedaddr	Get device’s extended address

The serial commands are case-sensitive. The parameters should be entered as continuous string of hex bytes, separated by space.

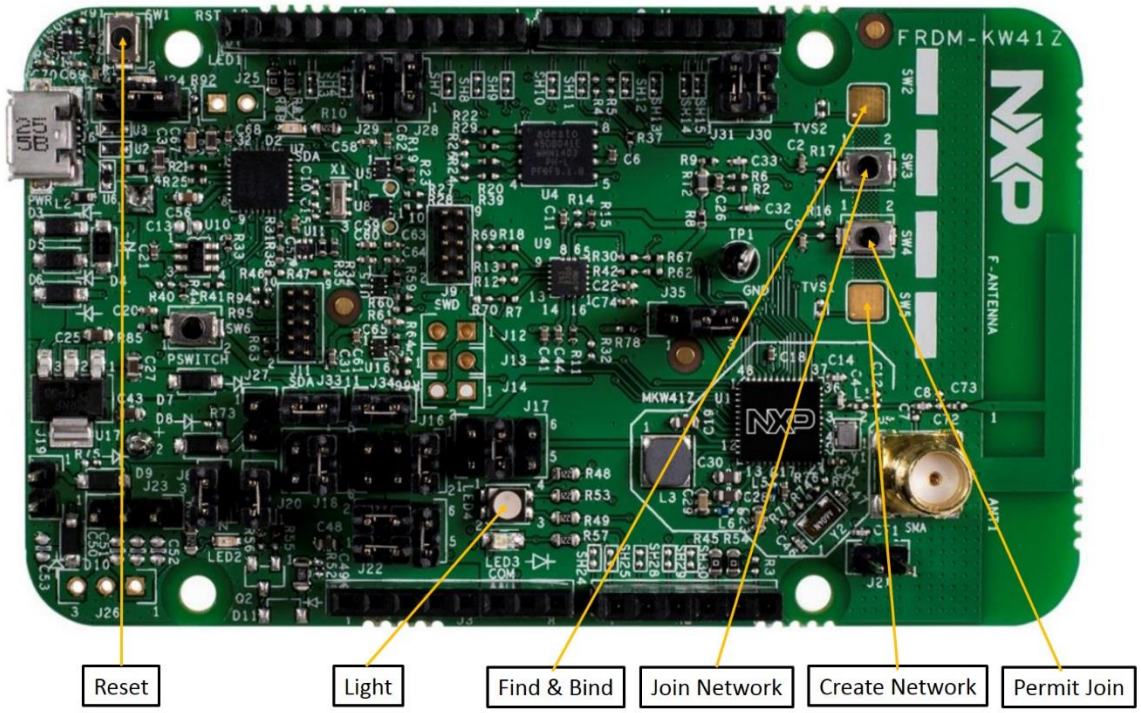
4.3 Using the Router

This section describes how to commission and operate the Router application in a ZigBee 3.0 network. To use this application, you must have programmed the application **ble_zigbee_router_frdmkw41z.bin** into the FRDM-KW41Z board, as described in Section 4.1

To incorporate the device into a ZigBee 3.0 network and then operate the device, work through Sections 4.3.2 to 4.3.5.

4.3.1 Router Functionality

The functionality of the Router application is described and illustrated below.



The Router can either join an existing network or decide to form a distributed network itself for other nodes to join. For details of the differences between a Centralised Trust Centre Network and a Distributed Network, refer to the *ZigBee Devices User Guide*. The Router supports the mandatory clusters and features of the Base Device as defined in the ZigBee Base Device Behavior Specification.

For the purpose of demonstrating the ‘Finding and Binding’ functionality, the Router also supports the On/Off Cluster as a server.

The Router provides two methods for triggering its functionality:

- Using commands from Thread Shell – available from IoT Toolbox - connected to the Router hardware. The serial interface is case-sensitive. For a summary of the serial interface, refer to Section 4.2.8.
- Using the buttons on the FRDM-KW41Z board.

4.3.2 Forming or Joining a Network

The Router is capable of either joining an existing network or, in the absence of a network, to form a distributed network for other devices to join.

4.3.2.1 Joining an Existing Network using Network Steering

A factory-new Router can join an existing network once the network has been opened to accept new joiners (Network Steering for a device on a network). To join an existing network press SW3 on the FRDM-KW41Z board.

This will cause the Router to start a network discovery and the associate process. Association is followed by an exchange of security materials and an update of the Trust Centre Link Key (if joining a Centralised Trust Centre Network).

If the join is unsuccessful, it can be retried by pressing SW3 again. Alternatively, the process for forming a distributed network can be follow, as described in Section 4.3.2.2.

The factory default state is indicated by the FRDM-KW41Z board blinking LED3.

If the join was successful, it is indicated by the FRDM-KW41Z board with the LED3 in solid red state.

4.3.2.2 Forming a Distributed Network

The Router can form a distributed network in the absence of an open network to join.

- To achieve this on a factory-new device press SW5 on the FRDM-KW41Z board.
- Enter “form” on the Thread Shell.

The Router will form a network with a random network key and begin operation. To allow other devices to join this network, follow the instructions in Section 4.3.3.

4.3.3 Allowing Other Devices to Join the Network

Once the Router is part of a network, the network must be opened to allow other devices to join (Network Steering while on a network). To do this:

- Press the SW4 button on the FRDM-KW41Z board.
- Enter “steer” on the Thread Shell.

The Router will then broadcast a Management Permit Join Request to the network to open the ‘permit join’ window for 180 seconds. The Network Steering process (for devices not on a network) can now be triggered on the devices that are to join the network.

4.3.4 Binding Devices

The Router supports the On/Off cluster as a server and implements the Finding and Binding process as a Target. To trigger Finding and Binding as a target, do the following:

1. Press the SW2 button on the FRDM-KW41Z board of all the target devices

   or type “find” in the Thread Shell.
2. Start Finding and Binding on the Initiator device.

This will cause the Router to self-identify for 180 seconds, while the Initiator will try to find the identifying devices, query their capabilities and create bindings on those with matching operational clusters. As part of this process, the Router may receive an Add Group Command and/or a Binding Request Command.

Reporting is a mandatory feature in ZigBee 3.0. The Router supports the On/Off cluster as a server and the OnOff attribute of this cluster is a reportable attribute as defined in the ZigBee Base Device Behaviour Specification. The Router holds a default configuration for reporting the state of the OnOff attribute. Once a device wishing to receive these periodic and on-change reports has created a remote binding, the Router will start to send reports to this bound device. The frequency of the reports depends on the default report configuration of

the individual target device, 61 seconds in this case. The device receiving the reports can request that this is changed by sending a Report Configuration command.

4.3.5 Operating the Device

The operational functionality of this device in this demonstration is provided by the On/Off cluster. Since the device supports the On/Off cluster server, its operation is passive and it responds to commands sent by bound devices. It responds to an OnOff Toggle command from a bound controller device by toggling the light (LED 4) on the FRDM-KW41Z board.

4.3.6 Rejoining a Network

As a Router, when this device is restarted in a state which is not factory-new, it will just resume operation in its previous state. All application, binding, group and network parameters are preserved in non-volatile memory.

4.3.7 Performing a Factory Reset

The Router can be returned to its factory-new state (erasing all persistent data except the outgoing network frame counter) as follows:

- Hold down the buttons SW2, SW3, SW4 or SW5 on the FRDM-KW41Z board for more than 8 seconds.
- Enter “factoryreset” on the Thread Shell.

  NOTE: This command will reset the MCU, and the Bluetooth connection will end.

The Router will then broadcast a Leave Indication on the old network, then delete all persistent data (except the outgoing network frame counter) and perform a software reset.

There are two supported over-the-air commands for removing a device from the network - these are:

- Network Leave Request without rejoin
- ZDO Management Network Leave Request without rejoin

The Reset command of the Basic cluster will cause the ZCL to be reset to its factory-new defaults, resetting all attributes and configured reports. This will not remove the device from the network - all network parameters, groups and bindings will remain in place.

4.3.8 Summary of Serial Interface Commands

Button	Serial Command	Action
SW2	-	Triggers Finding and Binding as a Target
SW3	-	Join an existing network
SW4	steer	Triggers Network Steering for a device on the network
SW5	form	Triggers network formation for a device not on a network
SW2-5 Very Long Press	factoryreset	Factory resets the device, erasing persistent data
Reset	reboot	Triggers a software reset (will end the Bluetooth connection)
-	print	Prints the Aps Key Table to the terminal
-	channel [<hex_value>]	Get channel mask/Set channel number
-	panid [<hex_value>]	Get / Set PAN Id
-	shortaddr [<hex_value>]	Get / Set device’s short address
-	epid	Get the extended PAN Id
-	extendedaddr	Get device’s extended address

The serial commands are case-sensitive. The parameters should be entered as continuous string of hex bytes, separated by space.

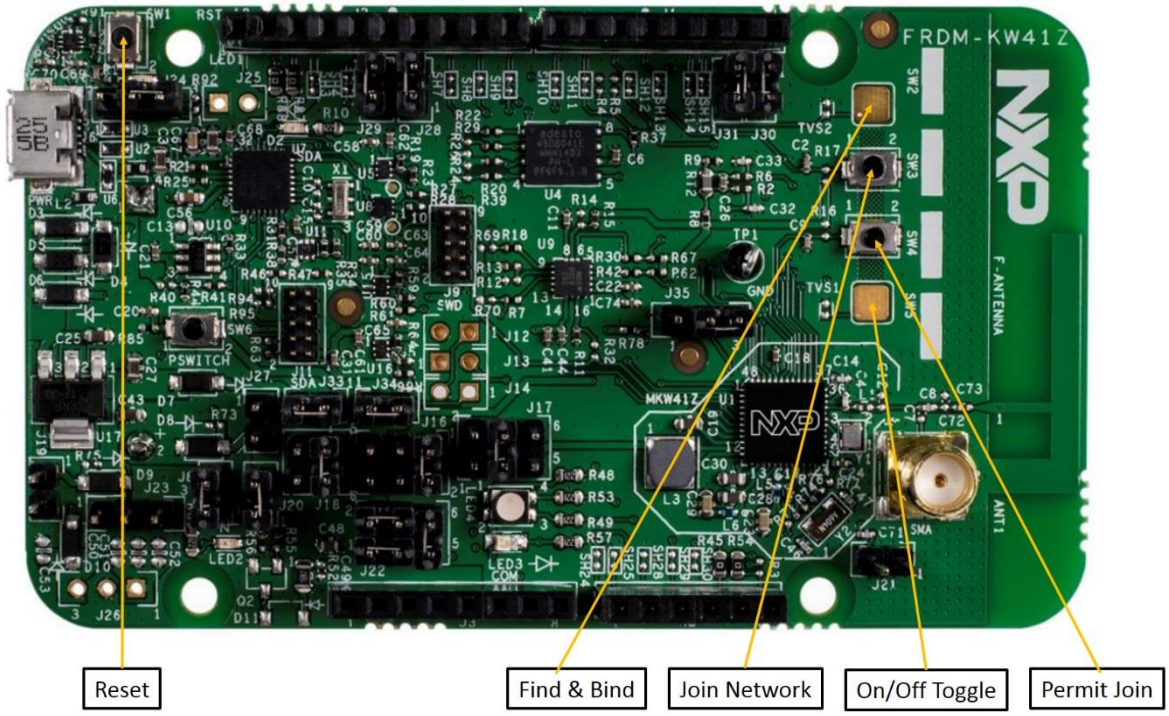
4.4 Using the End Device

This section describes how to commission and operate the End Device application in a ZigBee 3.0 network. To use this application, you must have programmed the application **ble_zigbee_end_device_frdmkw41z.bin** into the FRDM-KW41Z board, as described in Section 4.1.

To incorporate the device into a ZigBee 3.0 network and then operate the device, work through Sections 4.4.2 to 4.4.5.

4.4.1 End Device Functionality

The functionality of the End Device application is described and illustrated below.



The End Device is an ‘Rx Off when Idle’ device. It is not capable of forming a network or being a parent to other devices joining a network. The End Device supports the mandatory clusters and features of the Base Device as defined in the ZigBee Base Device Behaviour Specification.

For purpose of demonstrating the ‘Finding and Binding’ functionality, the End Device also supports the On/Off cluster as a client.

All communications to/from the End Device are passed through its parent Coordinator or Router and the End Device must send periodic Poll Requests to the parent to receive any messages that may be waiting for it.

The End Device factory default state is indicated by the FRDM-KW41Z board blinking LED3.

. For a summary of the serial interface, refer to Section 4.2.8.

4.4.2 Joining a Network

4.4.2.1 Joining an Existing Network using Network Steering

A factory-new End Device can join an existing network once the network has been opened to accept new joiners (Network Steering for a device on a network). This is achieved as follows:

- Trigger Network Steering on one of the devices already on the network.
- Press the button SW3 on the FRDM-KW41Z board of the End Device.
- Or enter “join” on the Thread Shell.

If the join was successful, it is indicated by the FRDM-KW41Z board with the LED3 in solid red state. After joining the network, the end device will be fast polling at 250 ms rate for first 5 seconds.

4.4.3 Allowing Other Devices to Join the Network

Once the End Device is part of a network, the End Device can request that the network is opened to allow other devices to join (Network Steering while on a network). To do this:

- Press the button SW4 on the FRDM-KW41Z of the End Device.
- Or enter “steer” on the Thread Shell.

The End Device will then unicast to its parent a Management Permit Join Request. The parent will then re-broadcast this to the network and open the ‘permit joining’ window for 180 seconds. The Network Steering process (for devices not on a network) can now be triggered on the devices that are to join the network. The End Device is capable of opening the network to new joiners, but it is not capable of being a parent to these new joiners.

4.4.4 Binding Devices

The End Device supports ‘Finding and Binding’ as an Initiator that tries to find targets to bind to. For the purpose of the demonstration, the End Device supports the On/Off cluster as a client, so the Finding and Binding process will look for devices that support the On/Off cluster as a server in order to create bindings.

To trigger the Finding and Binding as an Initiator on the End Device, first trigger Finding and Binding on any target device and then do the following on the End Device:

- Press the button SW2 on the FRDM-KW41Z board of the End Device
- Or enter “find” on the Thread Shell.

When Finding and Binding for a Target is completed and a binding is created, the End Device will send an Identify Off command to the target device to signal completion of the process for this Target. Depending on the type of bindings being created (either unicast or groupcast), an Add Group command may be sent to the Target.

Reporting is a mandatory feature in ZigBee 3.0, but it is not mandatory to request that reports are sent to a device. As an Rx Off when Idle device, the End Device is not in a position to receive any reports, so this device does not create bindings on target devices for them to send reports.

4.4.5 Operating the Device

The operational functionality of this device in this demonstration is provided by the On/Off cluster. You can now send an OnOff Toggle command to the bound devices (in the Binding table) as follows:

- Press the button SW5 on the FRDM-KW41Z board.
- Or enter “toggle” on the Thread Shell.

The effect that this command has on a bound device depends on the functionality related to the On/Off cluster on the device – for the Router in this demonstration, it will toggle a light (see Section 4.3).

4.4.6 Rejoining a Network

As an End Device, when this device is restarted in a state which is not factory-new, it will send a Network Rejoin Request to re-establish contact with its previous parent. If this fails, it will then try to join any Router on the network that will host it. The rejoin is attempted at power-on. All application, binding, group and network parameters are preserved in non-volatile memory.

4.4.7 Performing a Factory Reset

The End Device can be returned to its factory-new state (erasing all persistent data except the outgoing network frame counter) as follows:

- Hold down for more than 8 seconds any of the buttons SW2, SW3, SW4 or SW5 on the FRDM-KW41Z board.
- Enter “factoryreset” on the Thread Shell.

  NOTE: This command will reset the MCU, and the Bluetooth connection will end.

The End Device will then unicast a Leave Indication to its parent, which will re-broadcast this message to the old network. The End Device will then delete all persistent data (other than the outgoing network frame counter) and perform a software reset.

There are two supported over-the-air commands for removing a device from the network - these are:

- Network Leave Request without rejoin
- ZDO Management Network Leave Request without rejoin

The Reset command of the Basic cluster will cause the ZCL to be reset to its factory-new defaults, resetting all attributes and configured reports. This will not remove the device from the network - all network parameters, groups and bindings will remain in place.

4.4.8 Summary of Serial Interface Commands

Button	Serial Command	Action
SW2	find	Triggers Finding and Binding as an Initiator
SW3	-	Join an existing network
SW4	steer	Triggers Network Steering for a device on the network
SW5	toggle	Sends an OnOff Toggle command to bound devices
SW2-5 Very Long Press	factoryreset	Factory resets the device, erasing persistent data
Reset	reboot	Triggers a software reset (will end the Bluetooth connection)
-	print	Prints the Aps Key Table to the terminal
-	polltime [<hex_value>]	Set polling interval in seconds
-	identify	Identify a device from a network
-	channel [<hex_value>]	Get channel mask/Set channel number
-	panid [<hex_value>]	Get / Set PAN Id
-	shortaddr [<hex_value>]	Get / Set device's short address
-	epid	Get the extended PAN Id
-	extendedaddr	Get device's extended address

The serial commands are case-sensitive. The parameters should be entered as continuous string of hex bytes, separated by space.

4.5 Installation Codes

The ZigBee Base Device allows for devices to join a network using unique install codes, rather than the well-known Default Link Key. This install code is only used for the initial join and is replaced by the Trust Centre immediately after the join with a new unique Link Key to secure future communication between the Trust Centre and the individual device. An installation code is 16 bytes in length.

Each joining device (Router or End Device) must have a unique install code. How this code is generated and provisioned in real devices is beyond the scope of this Application Note. For the purpose of the demonstration, each joining device will create an install code that is its 8-byte IEEE/MAC address repeated once. For example, a device with an IEEE/MAC address of 00158D000035C9B8 will generate an install code of:

00:15:8D:00:00:35:C9:B8:00:15:8D:00:00:35:C9:B8

Before a device using install, codes can be joined to the network, the IEEE/MAC address and install code of this device need to be added to the Trust Centre of the network. The serial interface provides a command to do this. This command has the format:

```
Code <MAC Address> <Install code>
```

where:

- `<MAC Address>` is the IEEE/MAC address of the device (MSB first, and alphabetic characters are not case-sensitive).
- `<Install code>` is the install code (MSB first, alphabetic characters are not case-sensitive).

For a device with the above IEEE/MAC address, the command would be:

```
code 00158D000035C9B8 00158D000035C9B800158D000035C9B8
```

After provisioning the install code and IEEE/MAC address in the Trust Centre, the normal procedure for joining a new device to the network can be followed.

Once the install code has been used to join the new device, it is replaced with a new Trust Centre Link Key (the install code is discarded and not stored for re-use). If a device is factory reset, it will not be able to re-associate with the network until the install code is re-provisioned in the Trust Centre.

To build the devices in this Application Note to use install codes for joining, set BDB_SET_DEFAULT_TC_POLICY and BDB_JOIN_USES_INSTALL_CODE_KEY to TRUE in bdb_options.h, then clean and rebuild each device.

4.5.1 Installation codes provided via the NTAG interface

The NTAG interface is an alternative to serial interface to provide installation codes.

The NTAG I2C Plus kit (Product ID: OM23221ARD) is fully compatible with the FRDM-KW41Z board and therefore it can be used as mean to provide the installation codes.

The NTAG I2C Plus kit is interfacing with the FRMD-KW41Z board via only three signals (excluding power supply and GND):

- I2C SCL (PORTC pin 2 on FRDM-KW41Z)
- I2C SDA (PORTC pin 3 on FRDM-KW41Z)
- FD (PORTC pin 17 on FRDM-KW41Z)

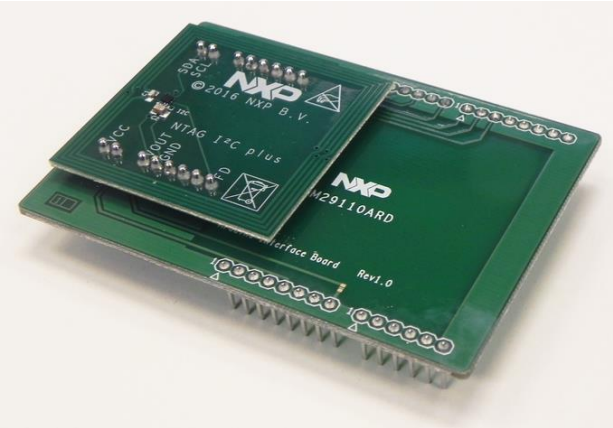


Figure 1: OM23221ARD: NTAG I2C Plus Shield

The Field Detect (FD) pin is connected to PORTC pin 17 on FRDM-KW41Z board and is configured as both-edges interrupt pin. At initialization, the software is configuring via the I2C interface the NTAG shield to generate interrupts on Field Present and Field Absent event.

A Field Absent interrupt means that the user removed a NFC enabled mobile device from the tag. This happens when the tag is read or written. This information is processed by the KW41Z software.

Note that only the Coordinator application enables the FD interrupt.

The chip from the NTAG I2C Plus shield (NT3H2111/NT3H2211) has a built-in 64 bytes SRAM memory array that can be used to pass-through data from the RF interface to the I2C one, or to mirror several EEPROM memory blocks.

From the I2C perspective, the EEPROM memory is accessed in blocks of 16 bytes. From the RF perspective, the EEPROM memory is accessed in blocks of 4 bytes.

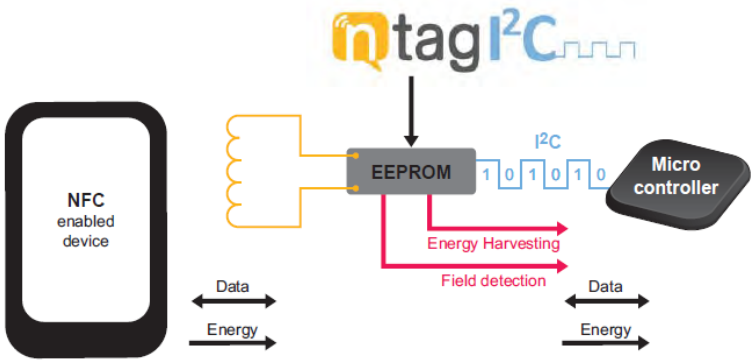


Figure 2: NTAG I2C PLUS contactless and contact system

From security reasons, the installation code and the extended address information is never written into the NTAG EEPROM memory and therefore the SRAM memory mirror feature is used by the ZigBee Coordinator application, to map the EEPROM sectors where the installation code and the device extended address are going to be written by the mobile device. After this information is used to commission the new joining device, it is removed from the SRAM memory.

The mechanism to commission a joining device using the installation codes via NFC is as follow: both devices (the Network Coordinator and the joining device) are required to have their own NTAG shield. Using a NFC-enabled mobile device and a common NFC software mobile application, the installation code as well as the Extended Address is read from the joining device NTAG shield and subsequently written to the Coordinator device NTAG shield.

The installation code and Extended Address are stored as NDEF messages on the NFC Tag.

```
# NDEF message:
[00] 91 02 30 53 70 91 01 18 54 05 64 65 2D 44 45 4E |..0Sp...T.de-DEN|
[10] 58 50 20 53 65 6D 69 63 6F 6E 64 75 63 74 6F 72 |XP Semiconductor|
[20] 73 11 01 09 55 01 6E 78 70 2E 63 6F 6D 2F 51 03 |s...U.nxp.com/Q.|
[30] 01 61 63 74 00 11 01 15 54 05 64 65 2D 44 45 5A |.act....T.de-DEZ|
[40] 42 5F 49 4E 53 54 41 4C 4C 5F 43 4F 44 45 11 01 |B_INSTALL_CODE..|
[50] 16 54 05 64 65 2D 44 45 30 33 41 32 31 31 33 45 |.T.de-DE03A2113E|
[60] 39 38 41 44 30 31 41 33 11 01 12 54 05 64 65 2D |98AD01A3...T.de-|
[70] 44 45 44 45 56 5F 45 58 54 5F 41 44 44 52 51 01 |DEDEV_EXT_ADDRQ.|
[80] 16 54 05 64 65 2D 44 45 30 30 31 31 32 32 33 33 |.T.de-DE00112233|
[90] 34 34 35 35 36 36 37 37                         |44556677        |
```

Figure 3: NDEF messages format (example)

The joining device (e.g. BDB End Device) application is checking at start-up if it has a compliant NDEF message on its own NTAG shield. If it does, it checks the install code and the Extended Address from the NTAG with the ones from its persistent data memory (PDM/NVM). If matches, the checking is complete. If it doesn't, the tag is re-written via I2C interface with the correct install code and Extended Address.

A non-compliant tag will be erased and replaced by a valid one.

To enable the commissioning using install codes via NTAG I2C Plus interface the following macros needs to be defined (config.h or project options):

#define APP_NTAG_I2C_PLUS

If the device is ZigBee Coordinator, the following macro needs to be defined and set.

#define APP_NTAG_DEVICE_IS_COORD_d 1

If the device is ZigBee End Device, the following macro needs to be defined and set.

#define APP_NTAG_DEVICE_IS_ZED_d 1

After enabling the NTAG support, the Coordinator and the End Device projects must be rebuilt.

4.6 Thread Shell

This section describes the user interactions for the Thread Shell application.

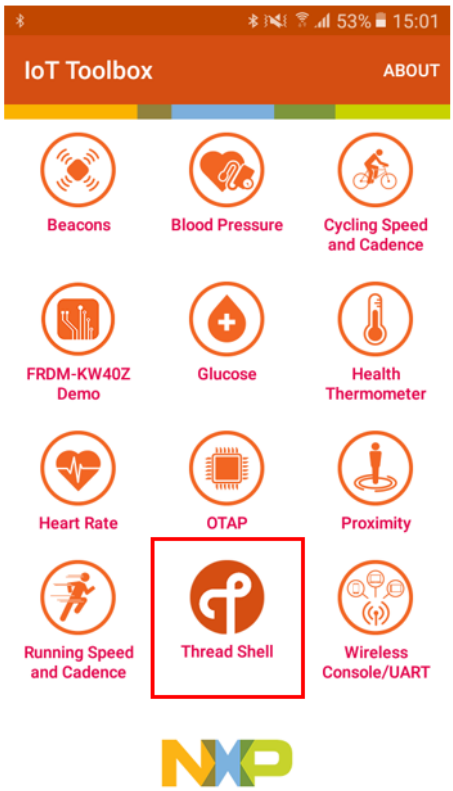
4.6.1 User interface

After the initialization, the BLE application will start advertising. When in Discoverable Mode, LED3 is flashing. When the node connects to a peer device, LED3 turns solid.

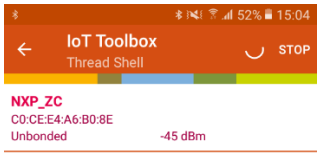
4.6.2 Usage

To test the demo use the following steps:

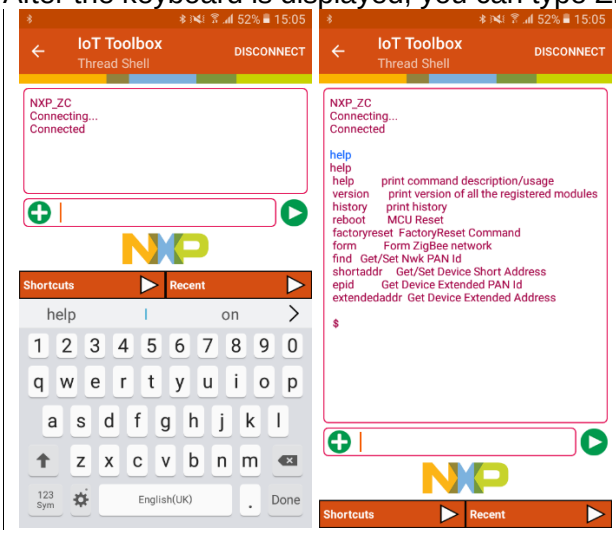
1. Load one board with the hybrid BLE – ZigBee3.0 application.
2. Open the IoT Toolbox application, and select “Thread Shell”



3. Select the device: “NXP_ZC” for Coordinator, “NXP_ZR” for Router and “NXP_ZED” for End Device



4. After the keyboard is displayed, you can type ZigBee Shell commands:



5 Developing with the Application Note

The example applications provided in this Application Note were developed using the KW41 ZigBee 3.0 v6.0.6 SDK and IAR Embedded Workbench 7.80 or MCUXpresso v10.1.1.

These are the resources that you should use to develop KW41Z hybrid ZigBee 3.0 – Bluetooth Low Energy applications. They are available free-of-charge via the ZigBee 3.0 page of the NXP web site.

Throughout your ZigBee 3.0 application development, you should refer to the documentation listed in Section 6.

5.1 Common Code

This section lists and describes the source files that provide functionality common to all the devices in this Application Note. and are held in the **boards\frdmkw41z\wireless_examples\hybrid\[application name]\freertos** directory.

app_zb_utils.c provides functionality for related to the IEEE MAC address, Install Codes and debug information through the serial interface.

app_zb_shell.c provides serial commands to interact with the ZigBee stack.

app_mac_config.h is the MAC layer configuration preinclude file.

app_framework_config.h is the framework configuration preinclude file.

ApplMain.c contains the Bluetooth application task and state machine.

ble_init.c contains the initialization of the Bluetooth Controller and Host stack.

app_config.c contains Bluetooth application config parameters.

gatt_db.h, gatt_uuid128.h contains GATT database settings.

5.2 Coordinator Application Code

This section lists and describes the source files for the Coordinator application code, which are provided in the **boards\frdmkw41z\wireless_examples\hybrid\ble_coordinator** directory for the application. You may wish to use this code as a basis for your own application development. You can rebuild your customised application as described in Section 5.5.

app_main.c hosts the main task, and defines and initialises system resources, queues, timers etc.

app_start.c manages the KW41Z application start-up and calls the initialisation functions.

app_coordinator.c hosts the event handlers for the application and the Base Device callback. The Base Device callback receives Base Device Events and AF Stack events after the Base Device has completed any processing that is required. These events can then be further processed by the application. The events include data indications that are passed to the ZCL for processing and network management events (such a ‘network formed’ and ‘new node has joined’) in order to keep the application informed of the network state. The application event queue is processed to receive button-press events.

app_zcl_task.c hosts the device-specific ZCL initialisation and callback functions. The ZCL callbacks notify the application of the results of received ZCL commands and responses, so that the application can take the appropriate action. The ZCL tick timer is used to provide ticks for the ZCL to manage timer-dependent events or state transitions.

bdb_options.h defines parameters used by the Base Device, such as primary and secondary channel masks.

zcl_options.h defines the features of the ZCL, such as which clusters are supported, whether a client and or a server, and which optional commands and attributes are supported. The mandatory commands and attributes of the selected cluster will be automatically included.

5.3 Router Application Code

This section lists and describes the source files for the Router application code, which are provided in the **boards\frdmkw41z\wireless_examples\hybrid\ble_router** directory for the application. You may wish to use this code as a basis for your own application development. You can rebuild your customised application as described in Section 5.5.

app_main.c hosts the main task, and defines and initialises system resources, queues, timers etc.

app_start.c manages the KW41Z application start-up and calls the initialisation functions.

app_router_node.c hosts the event handlers for the application and the Base Device callback. The Base Device callback receives Base Device Events and AF Stack events after the Base Device has completed any processing that is required. These events can then be further processed by the application. The events include data indications that are passed to the ZCL for processing and network management events (such a 'network formed' and 'new node has joined') in order to keep the application informed of the network state. The application event queue is processed to receive button-press events.

app_zcl_task.c hosts the device-specific ZCL initialisation and callback functions. The ZCL callbacks notify the application of the results of received ZCL commands and responses, so that the application can take the appropriate action. The ZCL tick timer is used to provide ticks for the ZCL to manage timer-dependent events or state transitions.

app_reporting.c provides the support for the reporting functionality of the device. It manages the restoring of the reporting configuration and the saving of any changes, when a Configure Reports command is received.

bdb_options.h defines parameters used by the Base Device, such as primary and secondary channel masks.

zcl_options.h defines the features of the ZCL, such as which clusters are supported, whether a client and or a server, and which optional commands and attributes are supported. The mandatory commands and attributes of the selected cluster will be automatically included.

5.4 End Device Application Code

This section lists and describes the source files for the End Device application code, which are provided in the **boards\frdmkw41z\wireless_examples\hybrid\ble_end_device** directory for the application. You may wish to use this code as a basis for your own application development. You can rebuild your customised application as described in Section 5.5.

app_main.c hosts the main task, and defines and initialises system resources, queues, timers etc.

app_start.c manages the KW41Z application start-up and calls the initialisation functions.

app_end_device_node.c hosts the event handlers for the application and the Base Device callback. The Base Device callback receives Base Device Events and AF Stack events after the Base Device has completed any processing that is required. These events can then be

further processed by the application. The events include data indications that are passed to the ZCL for processing and network management events (such a ‘network formed’ and ‘new node has joined’) in order to keep the application informed of the network state. The application event queue is processed to receive button-press events.

app_zcl_task.c hosts the device-specific ZCL initialisation and callback functions. The ZCL callbacks notify the application of the results of received ZCL commands and responses, so the application can take the appropriate action. The ZCL tick timer is used to provide ticks for the ZCL to manage timer-dependent events or state transitions.

bdb_options.h defines parameters used by the Base Device, such as primary and secondary channel masks.

zcl_options.h defines the features of the ZCL, such as which clusters are supported, whether a client and or a server, and which optional commands and attributes are supported. The mandatory commands and attributes of the selected cluster will be automatically included.

5.5 Rebuilding the Applications

This section describes how to rebuild the supplied applications, which you will need to do if you customise the applications for your own use.

5.5.1 Pre-requisites

It is assumed that you have installed the relevant development software on your PC, as detailed in Section 2.

The project files shouldbe located in the directory:

…\boards\frdmkw41z\wireless_examples\hybrid\[application name]\freertos\iar

5.5.2 Build Instructions

The software provided with this Application Note can be built for the KW41Z device.

The applications can be built from IAR Embedded Workbench 7.80 or from MCUXpresso

5.5.2.1 Using IAR Embedded Workbench

This section describes how to use IAR Embedded Workbench to build the example application.

To build the application and load it on to KW41Z boards, follow the instructions below:

1. Start the IDE and drag&drop the relevant project .eww file.
2. Build an application. To do this use the drop down menu at the top of the Workspace panel to select the relevant build configuration, Debug or Release, right click on the application name in the Workspace panel and select Make.. Repeat this to build the other applications.

   The binary files will be created in a folder with the build configuration name in the relevant directories for the applications.

4. Load the resulting binary files into the board. You can do this using the integrated flash programmer in IAR or using Firmware Loader from NXP Test Tool 12.

5.5.2.2 Using MCUXpresso

In order to load a project using MCUXpresso, please refer to the documentation provided, “Getting started with MCUXpresso SDK.pdf “and follow the steps described in chapter 7 “Run a demo using MCUXpresso IDE”.

6 Related Documents

The following manuals will be useful in developing custom applications based on this Application Note:

- ZigBee 3.0 Stack User Guide
- ZigBee 3.0 Devices User Guide
- ZigBee Cluster Library (for ZigBee 3.0) User Guide
- Bluetooth® Low Energy Demo Applications User's Guide

All the above manuals are available as PDF documents from the NXP web site.

7 Application Code Sizes

The applications of this Application Note have the following memory footprints on the KW41Z device, when using the KW41Z ZigBee 3.0 v6.0.6 SDK.

Application – Release Configuration, IAR 7.80.4.	**RO code (Bytes)**	**RO data (Bytes)**	**RW data (Bytes)**
App_BLE Coordinator	342458	18105	48750
App_BLE Router	335090	17957	46247
APP_BLE End Device	326898	17280	44324

8 Revision History

Version	Notes
0	First KW41Z release
1	Updates for KW41Z ZigBee 3.0 Alpha/EAR Release
2	Updates for KW41Z ZigBee 3.0 Beta/PRC Release
3	Updates for KW41Z ZigBee 3.0 GA/RFP Release

Important Notice

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

All trademarks are the property of their respective owners.

NXP Semiconductors

For the contact details of your local NXP office or distributor, refer to:

www.nxp.com