life.augmented

AN5506

Application note

Getting started with Zigbee® on STM32WB Series

Introduction

This application note guides the designers through the steps required to build specific Zigbee® applications based on STM32WB Series microcontrollers. It explains how to interface with the STM32WB Series microcontroller. It groups together the most important information related to Zigbee®.

To fully benefit from the information in this document and to develop an application, the user must be familiar with STM32 microcontrollers.

1 General information

This document applies to the STM32WB Series dual-core Arm®-based microcontrollers.

Note: Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

1.1 Acronyms and definitions

Table 1. Acronyms and definition

Acronym	Definition
API	Application programming interface
APS	Application support sub-layer
BDB	Base device behavior
HAL	Hardware abstraction layer
IAS	Intruder alarm system
IPCC	Inter-processor communication controller IP
MAC	Media access control
PAN	Personal area network
SED	Sleepy end device
ZCL	Zigbee cluster library
ZDO	Zigbee device object

1.2 Reference documents

- [1] AN5289 Building wireless applications with STM32WB Series microcontrollers
- [2] AN5492 Persistent data management Zigbee® and non-volatile memory in STM32WB Series
- [3] AN5491Creating manufacture specific clusters on STM32WB Series
- [4] AN5498 How to use Zigbee clusters templates on STM32WB Series
- [5] AN5500 ZSDK API implementation for Zigbee® on STM32WB Series

2 Zigbee communication protocol

2.1 Zigbee overview

Zigbee is an IEEE 802.15.4-based IOT protocol used to create wireless personal area (WPAN) networks. The aim is to provide a simple networking layer and standard application profiles that can be used to create interoperable solutions, with low-power and low-bandwidth constraint.

It concerns, among other things:

- Home automation
- Industrial control systems
- Building automation
- Medical data collection and monitoring
- HVAC control
- Wireless sensor networks

The data throughput is 250 write Kbit/s in 2.4 GHz band and the typical range is 10-20 meters.

2.2 Zigbee network

2.2.1 Type of devices

In Zigbee, there are three logical device types:

- Coordinator (ZC): This is the first node to be started. The coordinator is responsible for forming the network by allowing other nodes to join the network through it. The coordinator is responsible for starting the network and for choosing certain key network parameters. Once the network is established, the coordinator has a routing role. In a centralized network, every Zigbee mesh network must have one and only one coordinator.
- Router (ZR): This is a node with a routing capability which is also able to send and receive data. It also allows other nodes to join the network through it. A Zigbee mesh network can have multiple routers.
- End Device (ZED): This is a node which is only capable of sending and receiving data. It has no routing capability. A Zigbee mesh network can have multiple end devices. Some end device can also be sleepy end device allowing very low power consumptions.

2.2.2 Type of network

To satisfy a wide range of applications and to ensure the optimal balance of security, Zigbee offers two types of network: distributed and centralized:

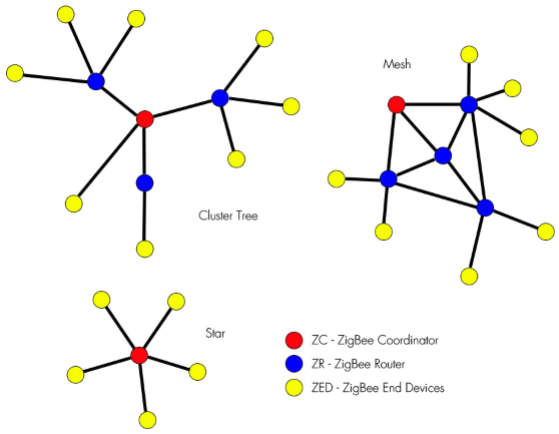
- In a distributed network, there are no coordinator. In this configuration, any router can issue network security keys. As more routers and end devices join the network, a router that is already on the network securely sends the network key. All devices on the network use the same network key to encrypt messages.
- In a centralized network, there is an entity named Trust Center (TC), which is typically the coordinator. The TC forms a centralized network and allows routers and end devices to join the network if they have proper credentials. In a centralized network, only the TC can issue encryption keys. The TC also establishes a unique TC Link Key for each device on the network as they join and link keys for each pair of devices as requested.

For obvious reasons, the centralized network is much more secure than the distributed one. Most of the Zigbee examples provided inside the STM32WB firmware package use a centralized network.

2.2.3 Zigbee network topologies

In a centralized network, Zigbee supports 3 types of network topologies as shown in the figure below.

Figure 1. Zigbee network topologies (centralized network)



2.2.4 Touchlink commissioning

Touchlink is a Zigbee feature which allows devices physically close to each other to communicate without being in the same Zigbee network. This is based on Inter-PAN communication mechanism, where devices can exchange information in their local area without having to form or join the same Zigbee network.

Touchlink process allows to discover and join together two devices in close proximity into the same PAN.

Touchlinking involves two different device roles:

- an initiator, which is the device initiating the Touchlink process. The initiator has to discover other devices, the targets, that can join him into the same PAN
- a target which is the device being discovered and joined to the initiator PAN.

A device that has not engaged in any touchlink process (since leaving a Zigbee network or forming a new one) is known as factory new. The device acts as a new one, according to its initiator/target role.

For a non-factory new device, all the Zigbee stack parameters (essentially network ones) are kept unchanged. This concerns the main Touchlink commissioning process steps such as

- Device discovery, including device identification
- Zigbee network formation and join use cases.

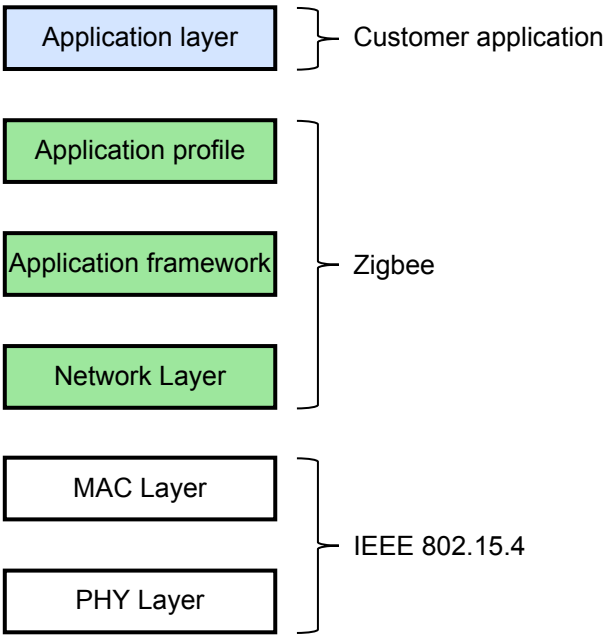
2.3 Zigbee architecture

2.3.1 General architecture

As described before, Zigbee is built on top of the IEEE 802.15.4 standard. Zigbee provides routing and multi-hop functions to the packet-based radio protocol. It is build on top of two layers specified by 802.15.4: the physical (PHY) and MAC layers.

The following figure describes the main components of a Zigbee stack and its articulation with IEEE 802.15.4 and general application layer.

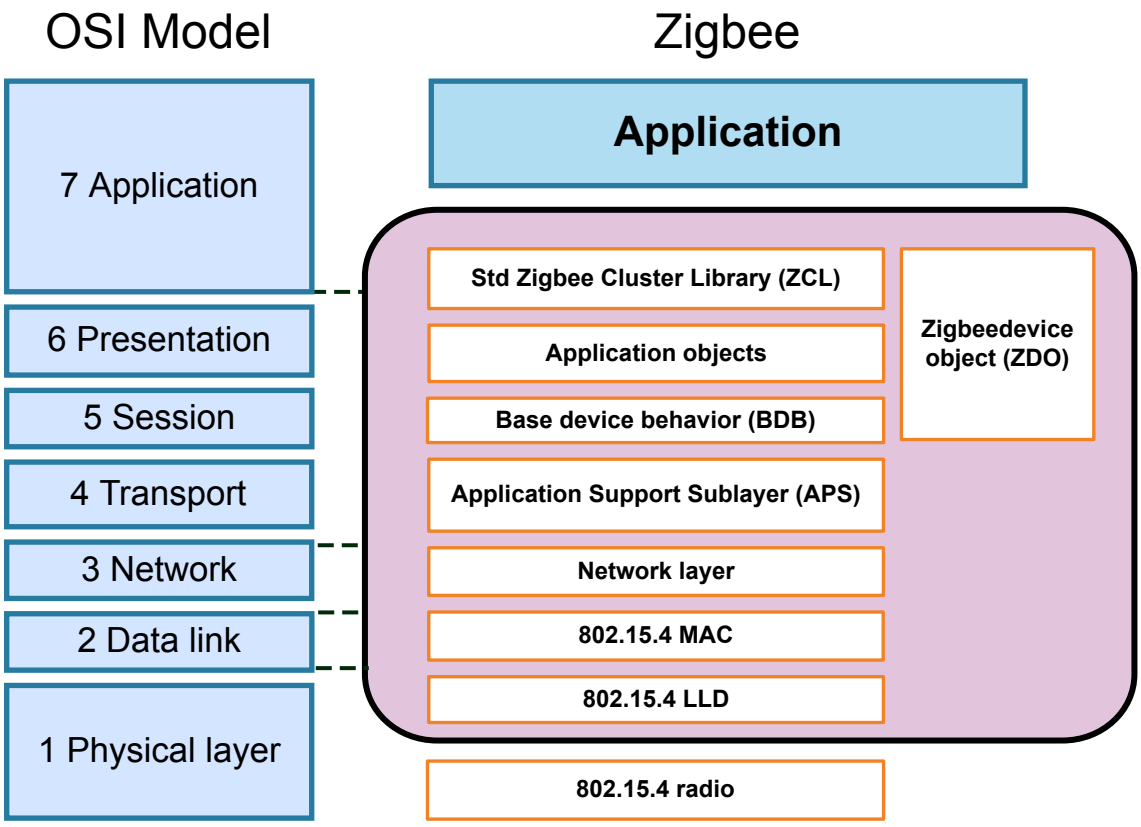
Figure 2. Zigbee stack overview



2.3.2 Zigbee stack layers

The stack layers defined by the ZigBee specification are based on the OSI 7-layer model. For Zigbee it concerns the network and application framework layers. The Zigbee stack is divided in multiple components, as shown in the figure below.

Figure 3. Zigbee stack description



Network (NWK) layer

The network layer is required to provide functionality to ensure correct operation of the IEEE 802.15.4 MAC sub-layer and to provide a suitable service interface to the application layer.

Among other things, this is the layer where networks are started, joined, left and discovered.

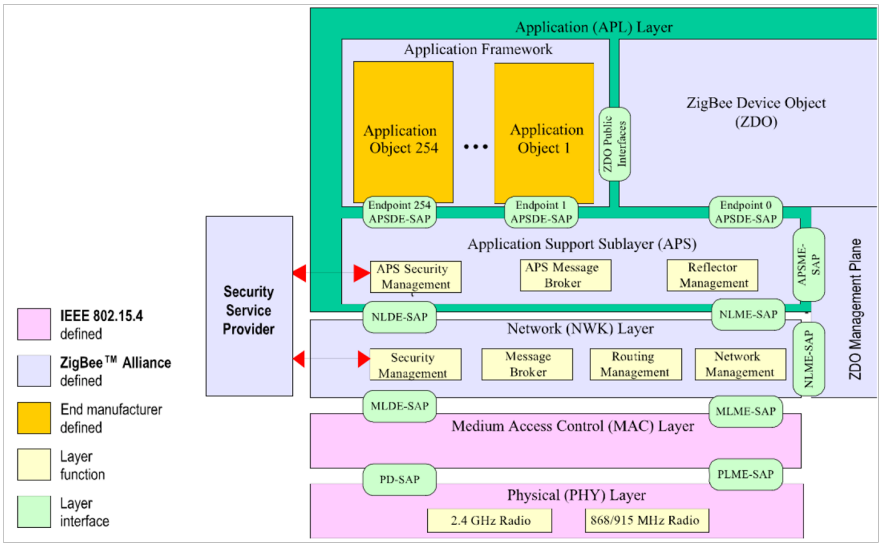
Table 2. Network layer functionalities

Zigbee network layer function	Coordinator	Router	End device
Establish a Zigbee network	x	-	-
Permit other devices to join or leave the network	x	x	-
Assign 16-bit network addresses	x	x	-
Discover and record paths for efficient message delivery	x	x	-
Discover and record list of one-hop neighbors	x	x	-
Route network packets	x	x	-
Receive or send network packets	x	x	x
Join or leave the network	x	x	x
Enter sleep mode	-	-	x

Application (APL) Layer

The APL layer is composed of several sublayers. The components of the APL layer are shown below:

Figure 4. Application layer sublayers



- **Application Support Sub-Layer (APS)**

  APS stands for application support sub-layer. It provides an interface between the network layer (NWK) and the application layer through a general set of services that are used by both the ZDO and the manufacturer-defined application objects.

  The APS is responsible for:

  – binding management
  – message forwarding between bound devices
  – group address definition and management
  – address mapping from 64-bit extended addresses to 16-bit NWK
  – addresses (dedicated table)
  – fragmentation and reassembly of packets
  – reliable data transport

  Binding in Zigbee allows an endpoint on one node to be connected, or “bound” to one or more endpoints on another node.

  The binding table maps a source address and source endpoint to one or more destination addresses and endpoints. This table is available and kept on all devices in the network.

- **Zigbee Device Object (ZDO)**

  The ZDO component handles the device management and communication functions. It includes:

  – initializing the APS sublayer and the NWK layer
  – device discovery
  – service discovery
  – network management, including defining the operating mode of the device (ZC, ZR or ZED).
  – security management
  – initiating and/or responding to remote binding requests

- **Base Device Behavior (BDB)**

  It is a standard SW component, which handles fundamental operations such as commissioning, network security and persistent data management. This device does not need an endpoint.

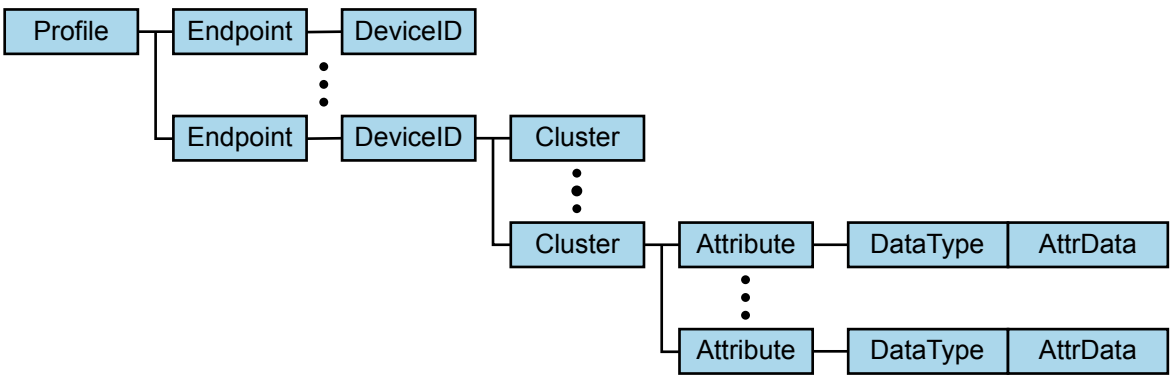
- **Application Framework**

  The application framework in Zigbee is very rich and it defines the environment in which application objects are hosted on devices.

  Data exchange between Zigbee device is performed in a client server model. It relies on an Application Profile, Cluster, Attribute model.

  The Application profile is a collection of device descriptions, which together form a cooperative application. The Profile defines the data exchange form for the application functions of a Zigbee physical device. A Profile consists of one or more Endpoints, each with one or more clusters associated.

Figure 5. Zigbee application profile organization



The clusters are a group of commands and attributes that define what a device can do. Clusters are managed by the ZCL (Zigbee Cluster Library).

Endpoint number that can be used for a Zigbee application are comprise between 1 and 240.

- **Zigbee cluster library (ZCL)**

  ZCL is the library which manage the Clusters. Clusters can be considered as a group of commands and attributes specific to a dedicated kind of application (DoorLock, OnOff, etc.).

  ZCL is defined by the Zigbee Alliance in order to speed the development and standardization of public profiles. With ZCL, manufacturers are able to quickly build Zigbee products with consistency and compatibility.

  A cluster is a related collection of commands and attributes, which together defines an interface to specific functionality. commands are actions that a cluster can take. Attributes are data or states within a cluster.

2.4 Zigbee profiles

2.4.1 Zigbee application profiles

A profile is a message-handling agreement between applications on different devices. It describes the logical components and their interfaces.

The aim of profiles is to provide interoperability between different manufacturers.

There are three types of profiles:

- public (standard), managed by the Zigbee Alliance
- private, defined by Zigbee vendors for restricted use
- published. This concerns previously private profiles that became published ones the owner profile decided to publish it

All profiles must have a unique profile identifier.

A profile uses a defined language for data exchange and a defined set of processing actions. Indeed, an application profile will specify the following:

- set of devices required in the application area
- functional description for each device
- set of clusters to implement the functionality
- which clusters are required by which devices

Each information that can be transferred between devices is called an attribute.

Attributes are grouped into clusters. All clusters and attributes are given unique identifiers. There are input cluster identifiers and output cluster identifiers. It is linked to client/server cluster architecture.

2.4.2 Zigbee device profiles

The Zigbee device profile is a collection of device descriptions and clusters performed directly by the ZDO. It applies to all Zigbee devices.

The Zigbee device profile is a template that show of how to write an application profile. It is defined in the Zigbee application level specification.

2.5 Zigbee addressing

Before joining a Zigbee network, a device with an IEEE 802.15.4 compliant
radio has a 64-bit address that is globally unique. For Zigbee, this MAC address is called an extended address.
When the device joins a Zigbee network, it receives a 16-bit address
called the NWK address. Either of these addresses, the 64-bit extended
address or the NWK address, can be used within the PAN to communicate
with a device.

2.5.1 Zigbee messaging

Once a device has joined the Zigbee network, it can send commands to other ones on the same network. There are two ways to address a device within the Zigbee network: direct and indirect addressing.

Direct addressing

The sender has to provide three pieces of information regarding the destination device.

This is unicast messages and they are composed of:

- Modification section indirect addressing
- Device address (NWK or IEEE extended address)
- Endpoint number
- Cluster ID

Indirect addressing

Indirect addressing is a local feature that simplifies the process. Therefore, even with indirect addressing, unicast messages are sent.

It requires that the above three types of information are available in the binding table.

The sending device only needs to know its own address, endpoint number and cluster ID. The binding table entry provides the destination information (device address and endpoint).

2.5.2 Broadcast addressing

In a Zigbee network, there are two broadcast levels:

- the broadcast with a MAC layer destination address of 0xFFFF. Any transceiver that is awake will receive the packet.

Note: The packet is re-transmitted three times by each device. This broadcast type should only be used when necessary.

- the broadcast of a message to all of the endpoints on the specified device. For such a broadcast, it is used the endpoint number 0xFF.

2.5.3 Group addressing

An application can assign multiple devices and specific endpoints on those devices to a single group address. The group assignation in based on the cluster ID, profile ID and source endpoint.

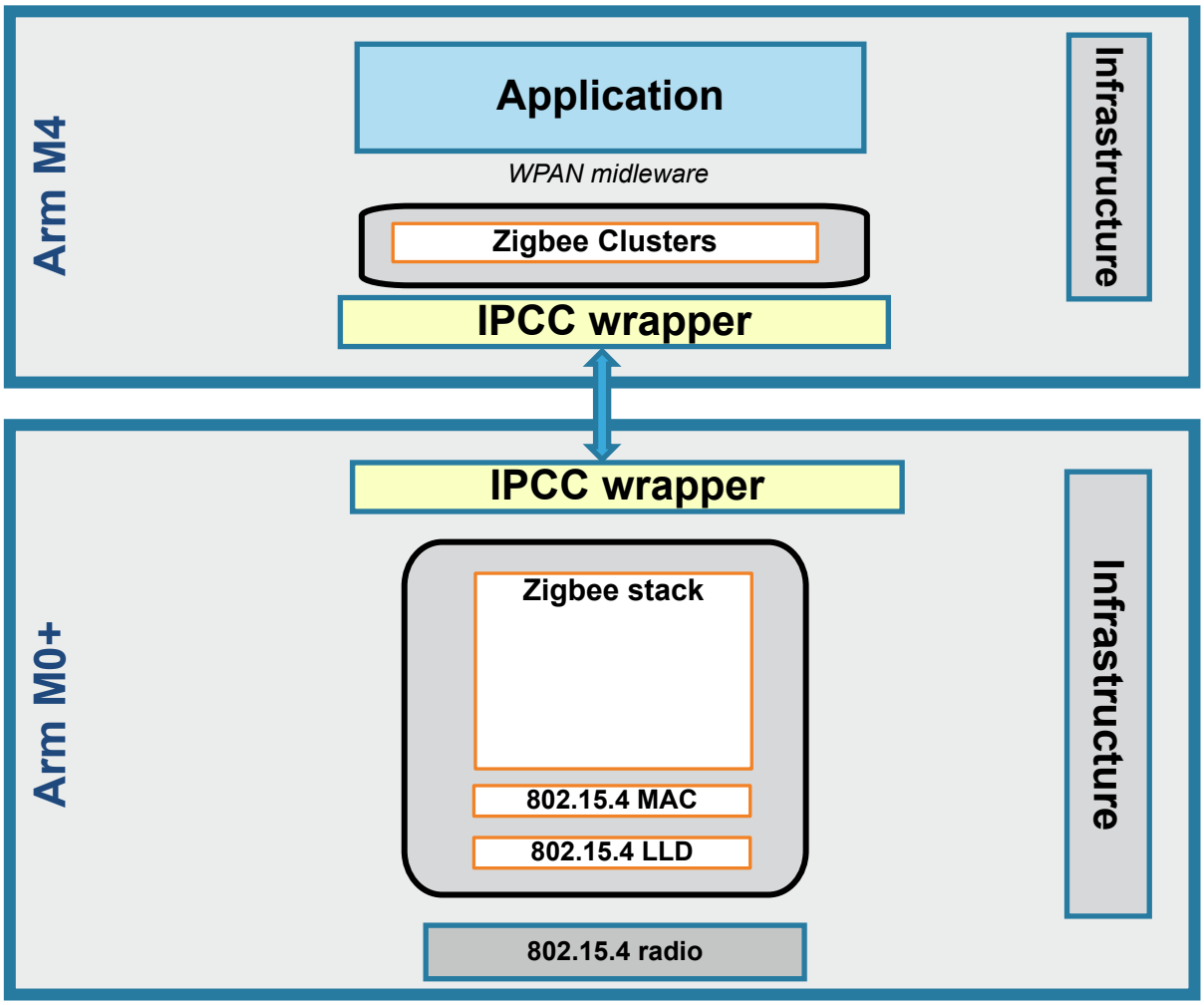
3 Zigbee on STM32WB

3.1 Architecture overview

The figure below gives an overview of the overall architecture. It shows in particular the split between the M4 and the M0. All the code running on the M0 is delivered as a binary library.

The customer has access only to the M4 core and sees the firmware running on M0 as a black box. The framework hides all the intercommunication between the M4 and M0. Dedicated IPCC channels are allocated for Zigbee.

Figure 6. Zigbee architecture overview on STM32WB

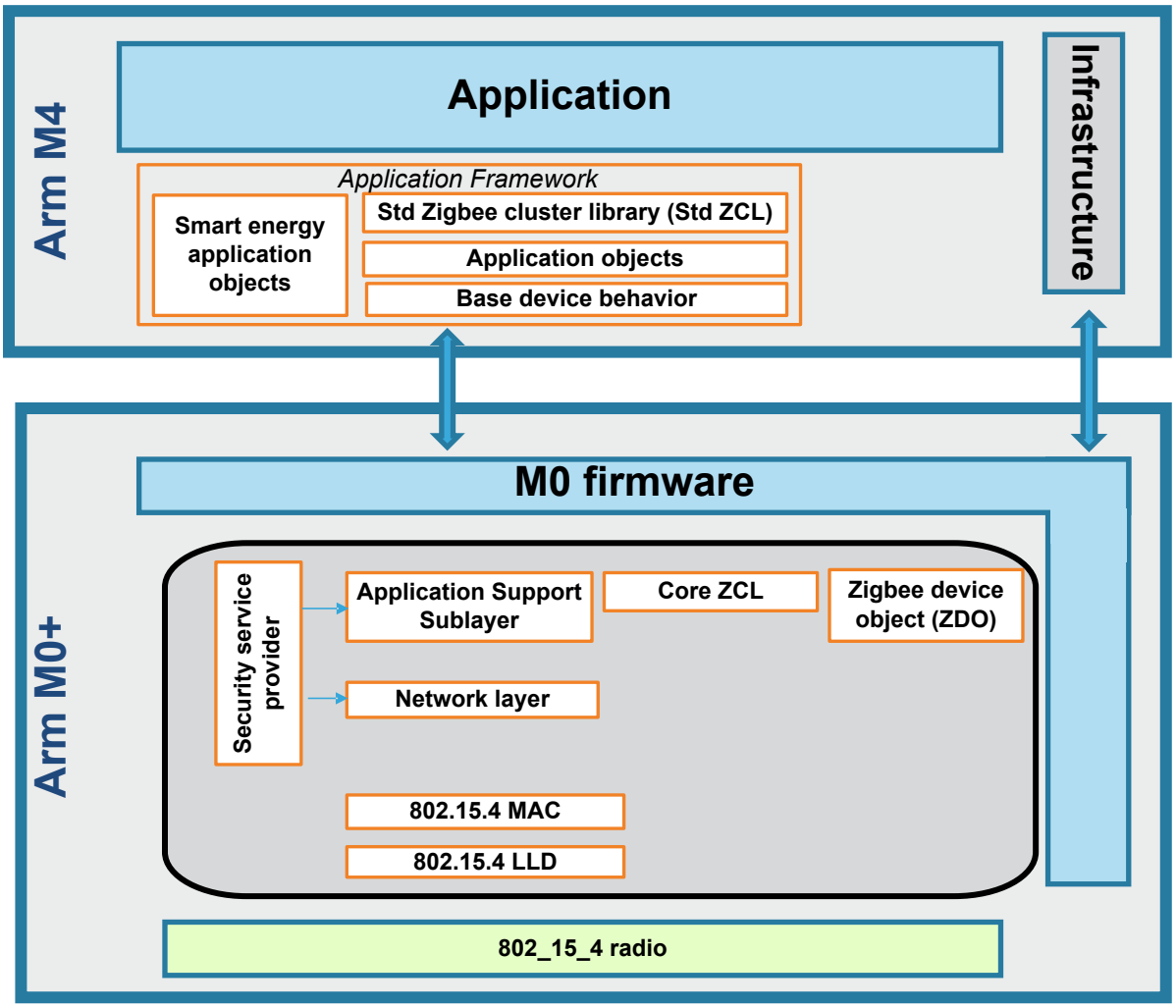


The Zigbee stack is running on top of the 802.15.4 MAC layer which itself use services provided by the 802.15.4 low level driver in charge of controlling the radio.

3.2 Zigbee stack layers on STM32WB

Figure 7. Zigbee layers and modules shows in more details the different Zigbee layers and how they are implemented on STM32WB MCU.

Figure 7. Zigbee layers and modules



3.3 Zigbee firmware supported

The STM32WB55 device supports two flavors of the stack . Both stacks are Zigbee PRO 2017 (revision 22) certified.

Table 3. Firmware supported (Zigbee standalone)

Stacks supported	Firmware associated
Zigbee FFD (Full feature device)	stm32wb5x_ZigBee_FFD_Full_fw.bin
Zigbee RFD (Reduced feature device	stm32wb5x_ZigBee_RFD_fw.bin

- An FFD can accept any role in the network. It can be a router, a coordinator or an end device.
- An RFD can support only end device role. An RFD has a smaller footprint compared to an FFD. When building an Application acting as a ‘sleepy end device’, in order to reach optimal low power consumption, it is mandatory to build this application using the Zigbee RFD stack.

ST supports both BLE and Zigbee protocol within a single binary firmware.

Table 4. Firmware supported (Zigbee concurrent mode)

Stacks supported	Firmware associated
BLE and Zigbee (static mode)	stm32wb5x_BLE_ZigBee_FFD_static_fw.bin

This binary is used for static concurrent mode applications. The example of such applications is provided under: `Projects\P-NUCLEO-WB55.Nucleo\Applications\BLE_ZigBee directory.`

Static mode

In static mode, it is possible to switch from BLE to Zigbee and vice-versa. When the BLE protocol is running, the Zigbee stack is no more running. When the BLE is stopped, the system can switch back to Zigbee. In this case, the Zigbee stack is fully re-initialized.

Note: Before running any Zigbee application on STM32WB, the user has to ensure that the proper firmware is downloaded on the M0. If it is not the case, he has to use STM32CubeProgrammer to load the appropriate binary.

All available Zigbee binaries are located under:
`/Projects/STM32WB_Copro_Wireless_Binaries/STM32WB5x.`
Refer to
`/Projects/STM32WB_Copro_Wireless_Binaries/STM32WB5x/Release_Notes.html`
for the detailed procedure on how to change the Wireless Coprocessor binary.

3.4 Zigbee clusters supported

The ZigBee ecosystem available on STM32WB supports ZigBee 3.0. ZigBee 3.0 clusters are ZCL 7 compliant. It includes BDB (base device behavior), Zigbee green power and several specific ZCL clusters as listed below:

Table 5. Zigbee cluster list ecosystem

Nb	Cluster ID	Cluster name
1	0x0000	Basic
2	0x0001	Power configuration
3	0x0003	Identify
4	0x0004	Groups
5	0x0005	Scenes
6	0x0006	On/Off
7	0x0008	Level control
8	0x000a	Time
9	0x0019	OTA upgrade
10	0x0020	Poll control
11	0x0021	Green power proxy
12	0x0102	Window covering

Nb	Cluster ID	Cluster name
13	0x0202	Fan control
14	0x0204	Thermostat user interface configuration
15	0x0300	Color control
16	0x0301	Ballast configuration
17	0x0400	Illuminance measurement
18	0x0402	Temperature measurement
19	0x0406	Occupancy sensing
20	0x0502	IAS warning device (WD)
21	0x0b05	Diagnostics
22	0x1000	Touchlink
23	0x0002	Device temperature configuration
24	0x0007	On/Off switch configuration
25	0x0009	Alarms
26	0x000b	RSSI location
27	0x0015	Commissioning
28	0x001a	Power profile cluster
29	0x0024	Nearest gateway cluster
30	0x0101	Door lock
31	0x0200	Pump configuration and control
32	0x0201	Thermostat
33	0x0203	Dehumidification control
34	0x0401	Illuminance level sensing
35	0x0403	Pressure measurement
36	0x0405	Relative humidity measurement
37	0x0500	IAS zone
38	0x0501	IAS ancillary control equipment (ACE)
39	0x0700	Price
40	0x0701	Demand response and load control
41	0x0702	Metering
42	0x0703	Messaging
43	0x0704	Smart energy tunneling (Complex metering)
44	0x0800	Key establishment
45	0x0904	Voice over Zigbee
46	0x0b01	Meter identification
47	0x0b04	Electrical measurement

- All these 47 clusters are available through the STM32_WPAN middleware. This middleware is common to BLE and Thread. For specific needs, a customer may create its own ‘proprietary’ cluster’. Refer to [3] for more details.
- The APIs relative to these clusters can be found under the following directory: `\Middlewares\ST\STM32_WPAN\ZigBee\stack\include`
- By default, all clusters are delivered as a single library. Nevertheless, it is possible to have access to the source code on demand.

4 STM32WB Zigbee application design

4.1 Zigbee application framework

The overall Zigbee application framework is based on a Client Server model. Each Zigbee application provided inside the STM32WB firmware package is delivered as two separate projects: one project handling the server part, and the other one handling the client part. To run these applications, it is required to have one board configured in client/coordinator mode, and all the other ones configured in server/router mode.

Note: It is possible to map a Client and or Server on any Zigbee, independently from the role it supports (ZC/ZR/ZED). Moreover, a single device can be a client and a server at the same time.

4.1.1 Application framework

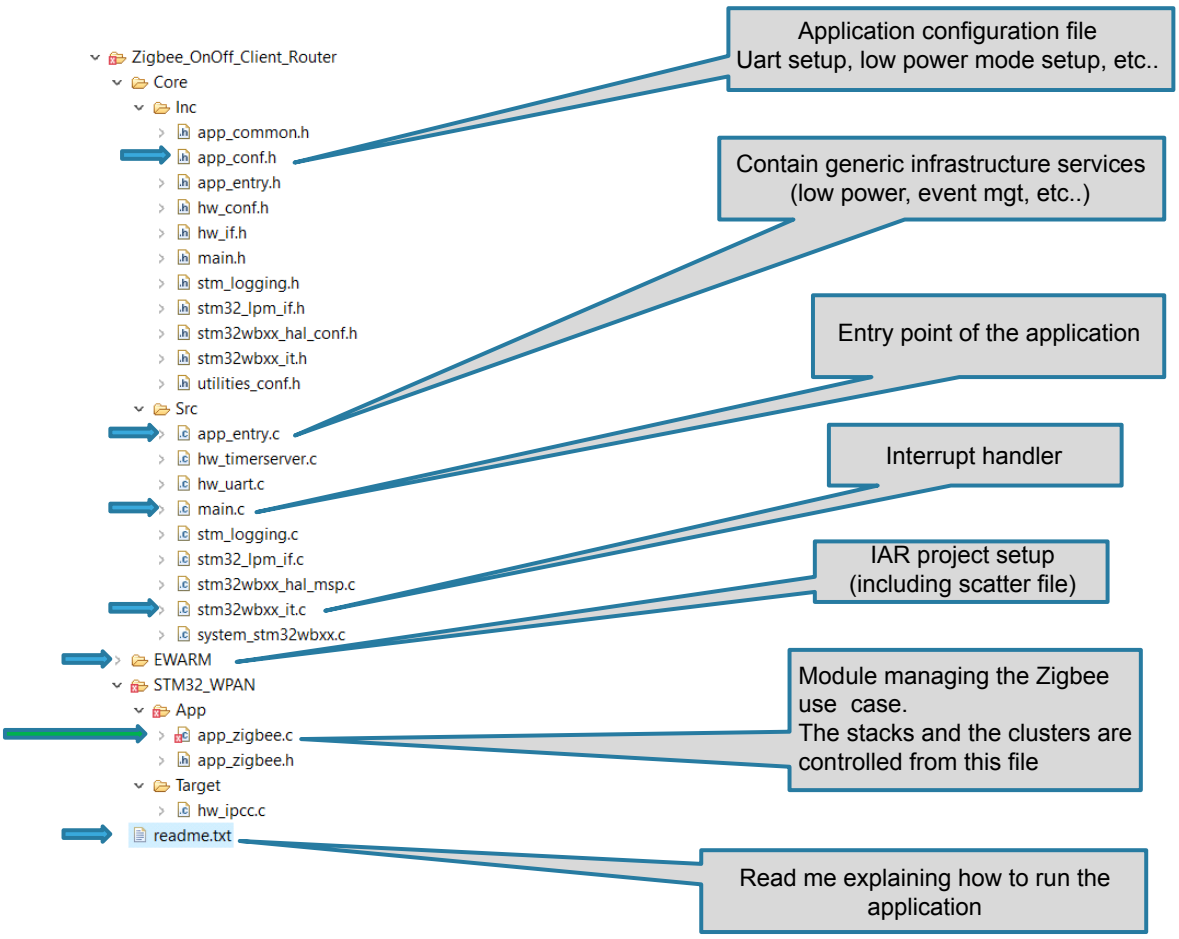
All projects are built using the same framework. In all projects, the Zigbee use case is set, defined and implemented inside the app_ZigBee.c file.

Under:

`Projects\Board_X\Applications\ZigBee\Zigbbe_Y_app\STM32_WPAN\App.`

All the other files present in the application projects are mainly used for the global infrastructure management (Interrupt management, IPCC wrapper, system startup and configuration, etc…)

Figure 8. Zigbee OnOff cluster application



4.1.2 Zigbee application architecture

This section explains the general STM32WB Zigbee application architecture which are defined in the Zigbee use case module files app_zigbee.h and app_zigbee.c.

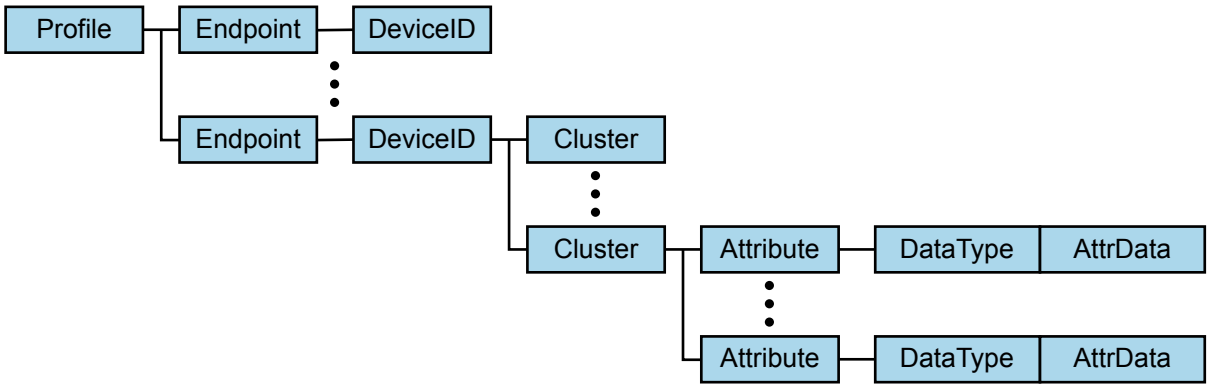
- **Application initialization**

  This is common step to all STM32WB application, including Thread and BLE ones. It includes the stack layer initialization (the Zigbee stack in this case) and the Zigbee Endpoint configuration step.

- **End point management**

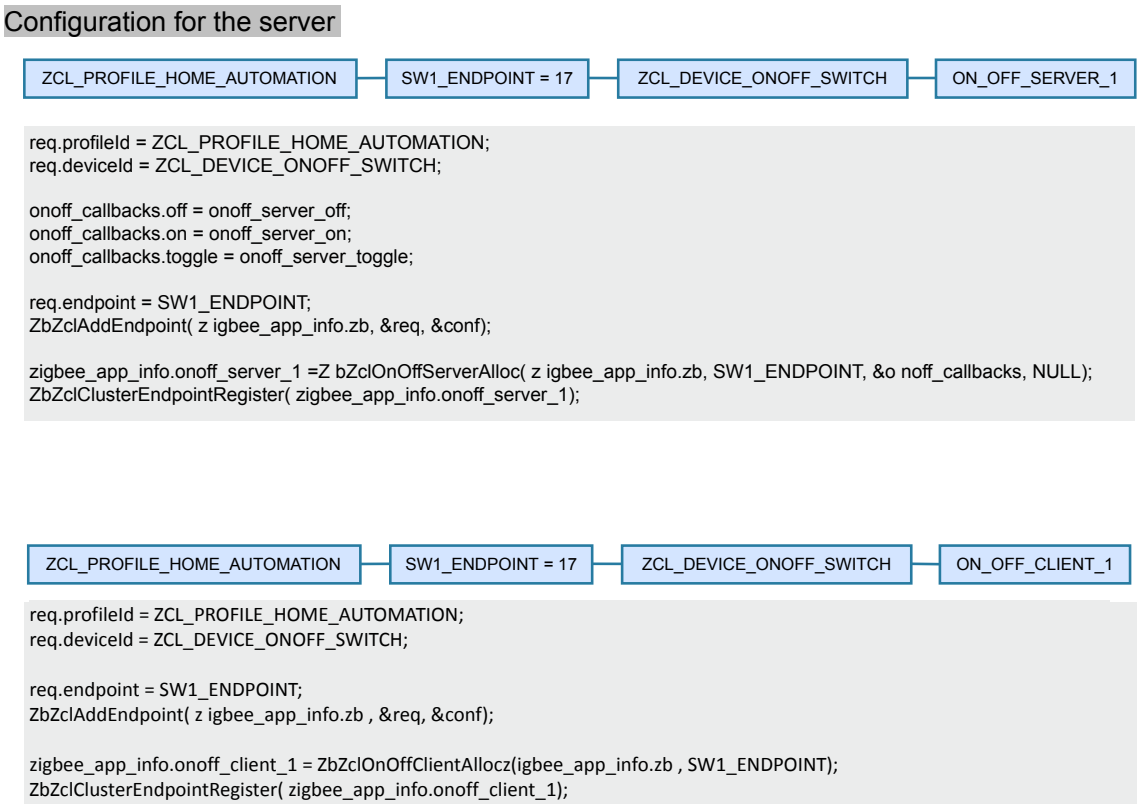
  For any Zigbee application, a Profile consists of one or more EndPoints, each with one or more clusters and a vertical structure of the attributes. The generic links between all these entities is the following:

Figure 9. Zigbee endpoint/cluster relationship



For example, in the OnOff application, the end point configuration is managed as follows:

Figure 10. Zigbee OnOff application end point configuration



Zigbee network management

This step includes the Zigbee network formation or joining. They are managed by a specific task, TASK_ZIGBEE_NETWORK_FORM. The task mostly includes the Zigbee network configuration and the associated startup procedure. Further details are given in the dedicated section.

The blue led lights on when the procedure has been successful. When for any reason, the joining procedure fails, the joining task is rescheduled.

In conclusion, if the application needs it, the APS layer can be addressed using the Group management. The Groupcast, as the STM32WB Zigbee framework, allows accessing the APS layer.

Group management

A group being a collection of nodes inside a network. Some APS primitives allow the higher layer to request that group membership for a particular group to be added for a particular endpoint.

For example, for the OnOff application sample, multiple OnOff clients can interact with the unique OnOff Server, which act as coordinator using groups.

User specific code

At the end of the TASK_ZIGBEE_NETWORK_FORM task the generic task TASK_ZIGBEE_APP_START is launched if needed. At this point the application in functional for a Zigbee point of view.

It's here the user can perform the next steps of its application. This includes local or remotes Zigbee commands with the associated callbacks.

Note: When implementing a callback for any command, the user should wait to return to this callback before requesting another command in order to avoid IPCC deadlock. Indeed, many M4 application commands are going through the IPCC wrapper in order to control the M0. For this reason the user should use available scheduler events API just after a Zigbee command. Tthe M4 CPU should wait for the event that is raised at the end of the associated Zigbee command callback. For instance, there is below an example of scheduler events usage with a Zigbee remote write request commands.

Figure 11. Zigbee command using scheduler events

```
static void APP_ZIGBEE_RemoteWrite_cb(…){
  …
  /* Unlock the waiting event */
  UTIL_SEQ_SetEvt(EVENT_ZIGBEE_CONTINUE_INIT);
  …
}

…
status = ZbZclWriteReq(zigbee_app_info.commissioning_client, &RemoteWriteReq, APP_ZIGBEE_RemoteWrite_cb, NULL);
UTIL_SEQ_WaitEvt(EVENT_ZIGBEE_CONTINUE_INIT);
…
```

4.1.3 Zigbee network startup procedure

The Zigbee network startup procedure used in the Zigbee network management is based on waiting events. Indeed, the application waits until the network startup result.

- For a Zigbee coordinator on a centralized Zigbee network, it waits until the network is formed (PAN ID selection).
- For a Zigbee router/end device on a centralized network, it waits until the association result.

There are 3 main startup procedure types. This concern all types of network:

- Centralized
- Distributed networks
- Touchlink Commissioning:
- – The usual Zigbee startup as shown on the Zigbee_OnOff_Server_Coord application example.
  – The Zigbee startup with persistence where the persistent data, previously stored before the device power off, are restored. This include Zigbee stack and cluster parameters. Detailed explanation is given on a specific Zigbee Persistent data document (Ref[1]). Refer to Zigbee_OnOff_Coord_NVM and Zigbee_OnOff_Router_NVM applications examples.
  – The CBKE (Certificate Based Key Establishment) Zigbee startup. This startup includes a complete certificate exchange/validation for the joining process.

Zigbee device network joining timeout

After a device has set up or joining a Zigbee network, the time when another device is allowed to join to this device (coordinator or router for a Zigbee centralized network for example) is fixed.

As a result, after this delay, joining the network for a device is not permitted (Zigbee association permit value). This is also the case when a device is reset after this time.

In order for a device to join a Zigbee network when the permit join timeout is over, the Zigbee coordinator can allow a parent device (for instance a Zigbee router) to permit joining. This is done by sending a ZDO message to parent.

For the coordinator, it can sent a ZDO message to itself to allow permit join for an extra amount of time.

Note: Zigbee persistent data feature can also be used in order to keep stack parameters and be able to reconnect to a network (or setting up) after a reset/shutdown.

4.1.4 Traces

Both traces coming from the stack itself (running on M0 core) and from the application itself (running on M4 side) are managed by the M4 and are routed through an UART (configured at compilation time using app_conf.h file).

To get the traces you need to connect your Board to the HyperTerminal (through the STLink Virtual COM Port). The UART must be configured as follows:

- BaudRate = 115200 baud
- Word Length = 8 Bits
- Stop Bit = 1 bit
- Parity = none
- Flow control = none

5 STM32WB Zigbee application

Several Zigbee applications are delivered inside the STM32WB firmware package. These applications are available on P-NUCLEO-WB55.Nucleo boards and on P-NUCLEO-WB55.USBDongle.

The full list of applications is available in STM32CubeProjectsList.html file (under \Projects directory).

The purpose of these applications is mainly to provide simple examples highlighting the usage of specific clusters.

The easiest way to start playing with Zigbee on STM32WB is to use the ZigBee_OnOff_Server_Coord and ZigBee_OnOff_Client_Router applications.

5.1 Zigbee general applications

Table 6. Zigbee applications available

Projects name	Description
Zigbee_APS_Coord and Zigbee_APS_Router	The purpose of this application is to show: • how to create a Zigbee centralized network • how to use the APSDE interface to the Zigbee stack directly • how to send and receive raw APS messages between devices on the network
Zigbee_Commissioning_Client_Coord Zigbee_Commissioning_Server_Router	The purpose of this application is to show how to create a Zigbee centralized network, and how to operate a commissioning process using the commissioning cluster
Zigbee_DevTemp_Server_Coord Zigbee_DevTemp_Client_Router	How to use device temperature cluster on a centralized Zigbee network.
Zigbee_Diagnostic_Server_Coord Zigbee_Diagnostic_Client_Router	How to use diagnostic on a centralized Zigbee network.
Zigbee_DoorLock_Server_Coord Zigbee_DoorLock_Client_Router	How to use door lock cluster on a centralized Zigbee network.
Zigbee_IAS_WD_Server_Coord Zigbee_IAS_WD_Client_Router	How to use IAS WD cluster on a centralized Zigbee network.
Zigbee_MeterId_Server_Coord Zigbee_MeterId_Client_Router	How to use meter identification cluster on a centralized Zigbee network.
Zigbee_OnOff_Client_DistribZigbee_OnOff_Server_Distrib	How to use OnOff cluster on a distributed Zigbee network.
Zigbee_OnOff_Server_Coord Zigbee_OnOff_Client_Router	How to use OnOff cluster on a centralized Zigbee network.
Zigbee_OnOff_Server_Coord Zigbee_OnOff_Client_SED	How to use OnOff cluster on a centralized Zigbee network with a sleepy end device (SED) as client. The SED client is configured to support low power mode STOP2 with a 3µA consumption when IDLE.
Zigbee_OnOff_Coord_NVM Zigbee_OnOff_Router_NVM	Description of the Zigbee OnOff cluster application with usage of persistent data using a centralized network.
Zigbee_OTA_Client_Router Zigbee_OTA_Server_Coord Zigbee_OnOff_Client_Router_Ota	Description of the Zigbee OTA cluster application
Zigbee_PollControl_Client_Coord Zigbee_PollControl_Server_SED	How to use poll control cluster on a centralized Zigbee network. Poll control cluster is used for remotely operating on a Sleepy End Device (SED).

Projects name	Description
Zigbee_PowerProfile_Client_Coord Zigbee_PowerProfile_Server_Router	How to use power profile cluster on a centralized Zigbee network. This demo shows how to use power profile cluster in an appliance/home gateway configuration by simulating a white food generic behavior.
Zigbee_PressMeas_Server_Coord Zigbee_PressMeas_Client_Router	How to use pressure measurement cluster on a centralized Zigbee network
Zigbee_SE_Msg_Client_Coord Zigbee_SE_Msg_Server_Router	How to use SE messaging cluster on a centralized Zigbee network.

5.2 Zigbee commissioning

This application shows the commissioning process between a Zigbee commissioning server and a commissioning client.

It shows a device distributing its Zigbee parameters (channel, panid, …) to another device using to the commissioning process. Commissioning is based on Inter-PAN communication mechanism where devices can exchange information in their local area without having to form or join the same Zigbee network.

One device acts as commissioner and the other one as joiner.

In this application, the commissioner accepts a newcomer in its Zigbee network.

This application requires two STM32WBxx_NUCLEO boards.

5.3 Sleepy End Device

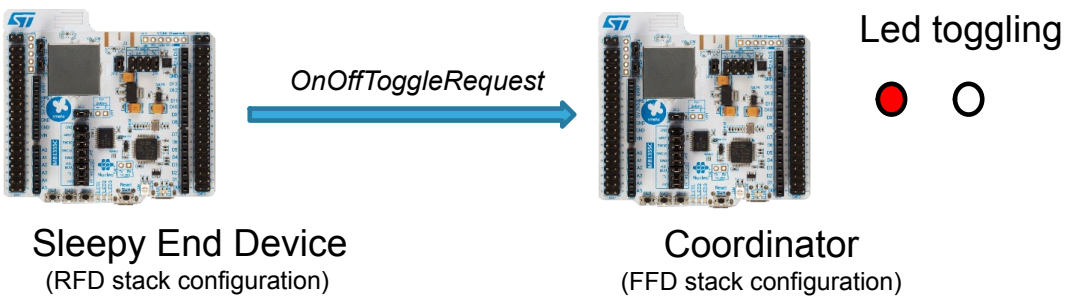
5.3.1 Sleepy End Device principle

To run this application, it is requested to have two boards

- - Board 1: STM32WB55xx Nucleo board loaded with:
  - wireless coprocessor: stm32wb5x_Zigbee_FFD_fw.bin
  - application: Zigbee_OnOff_Server_Coord
- Board 2: STM32WB55xx Nucleo board loaded with:
  - wireless coprocessor: stm32wb5x_Zigbee_RFD_fw.bin
  - application: Zigbee_OnOff_Client_SED

Once the Sleepy End Device (Board 2 acting as client) has joined the Zigbee network controlled by the Coordinator (Board 1 acting as Server), it will send a unicast OnOff toggle request every second to the Coordinator. At this stage, the LED1 on Board 1 should toggle every second when receiving the request coming from the SED.

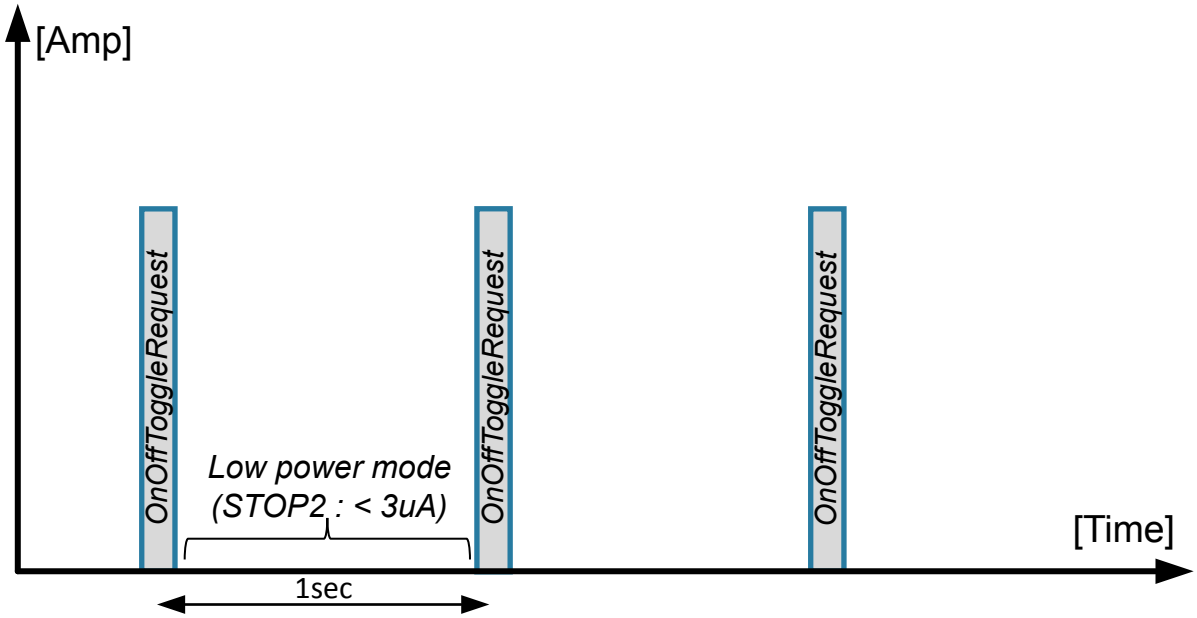
Figure 12. Sleepy End Device user case



In order to have the lowest power consumptions as possible on SED side, this application is by default compiled with the flag CFG_FULL_LOW_POWER set to 0 (in app_conf.h file). In this configuration, LEDS are no more available, and debug access to the M4 core is also disabled.

In this configuration, using the power shield, it is possible to check that between the sending of two requests to the Coordinator, the SED is able to reach low power mode (STOP2).

Figure 13. Sleepy End Device consumption

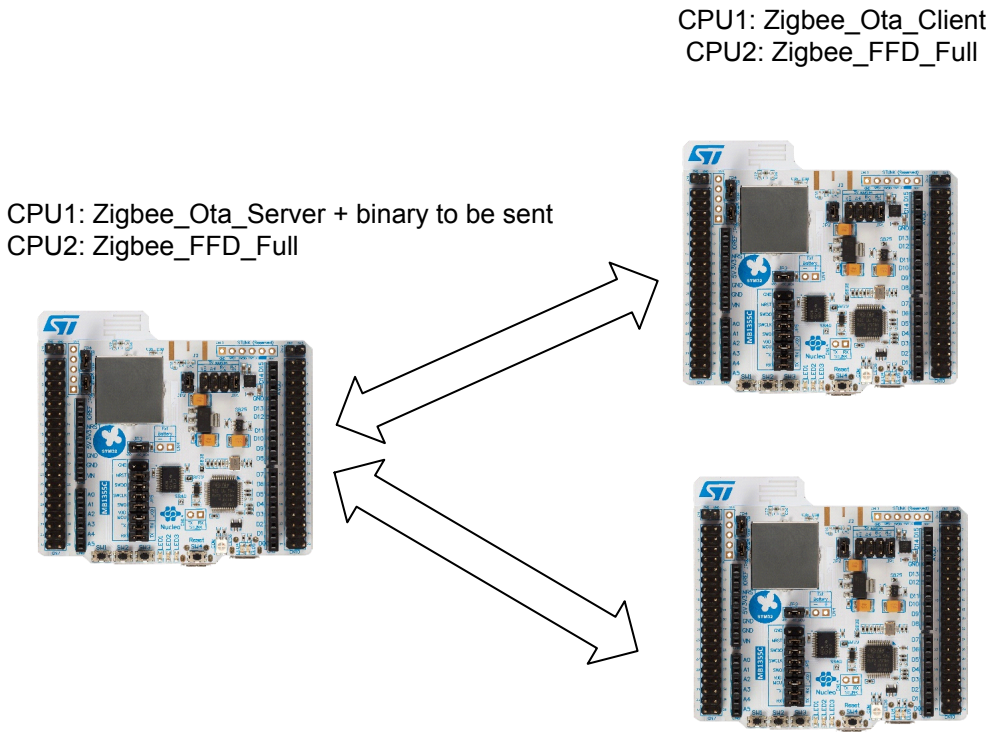


5.4 ZigBee FUOTA

5.4.1 Zigbee FUOTA principle

The goal is to use Zigbee protocol to update CPU1 application binary or CPU2 wireless coprocessor binary on a remote device.

Figure 14. Zigbee FUOTA network topology



This thread requires at least two STM32WBxx boards (see Figure 13) running Zigbee protocol with specific applications:

- one board running ZigBee_Ota_Server application
- one or more boards running ZigBee_Ota_Client application

FUOTA process can take place only on one device at a time.

The server initiates a FUOTA provisioning process and one client must respond to it.

Multiple clients are updated one at the time.

5.4.2 Memory mapping

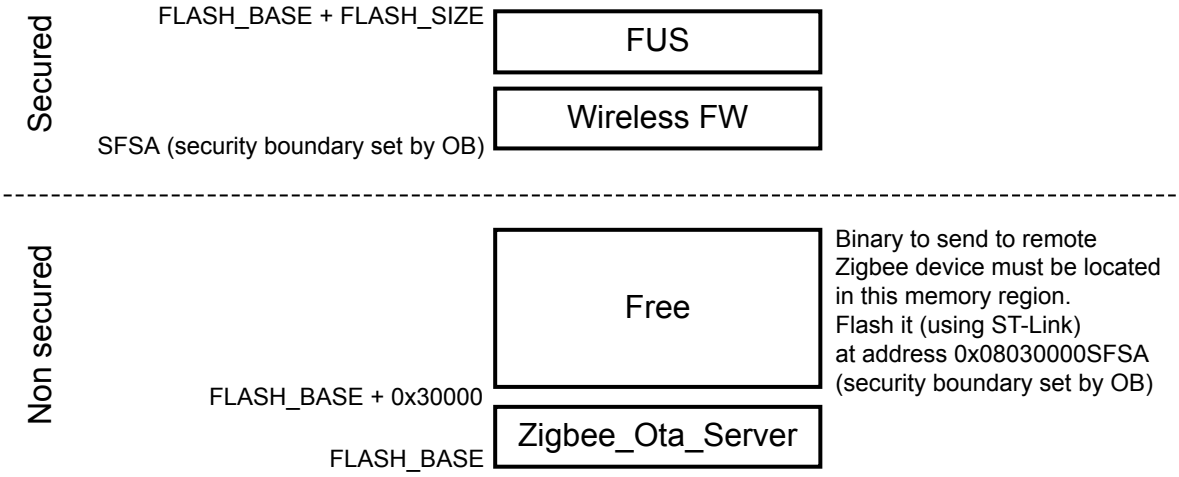
Server side

The binary file to be installed (either for CPU1 or for CPU2 update) on remote device has to be flashed first on the “FREE” memory region on the Server side (see Figure 15).

Maximum size of the binary to be transferred is equal to:

```
FREE region size = SFSA Address - (FLASH_BASE - 0x8030000)
```

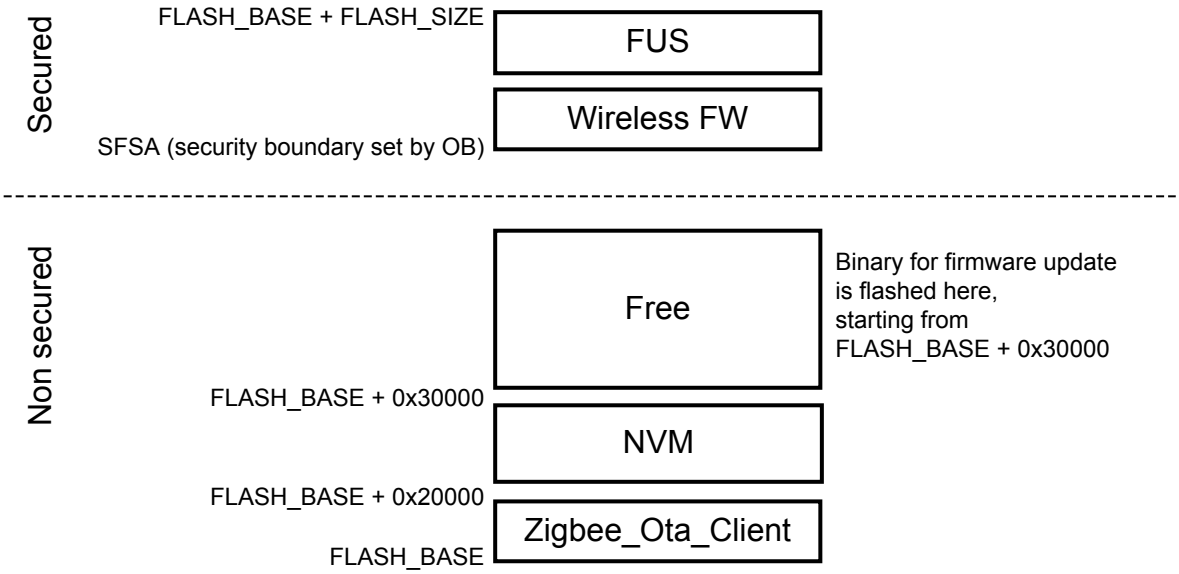
Figure 15. **OTA server (ZigBee_Ota_Server) Flash memory mapping**



Client side

On the client side, before receiving the binary from the server, the Flash memory is as shown in Figure 16.

Figure 16. **FUOTA client Flash memory mapping initial state**



After receiving the binary data from server side, the Flash memory is updated as shown in Figure 17 and Figure 18, respectively, for CPU1 binary transfer and CPU2 binary transfer.

Figure 17. **FUOTA client Flash memory mapping after CPU1 binary transfer**

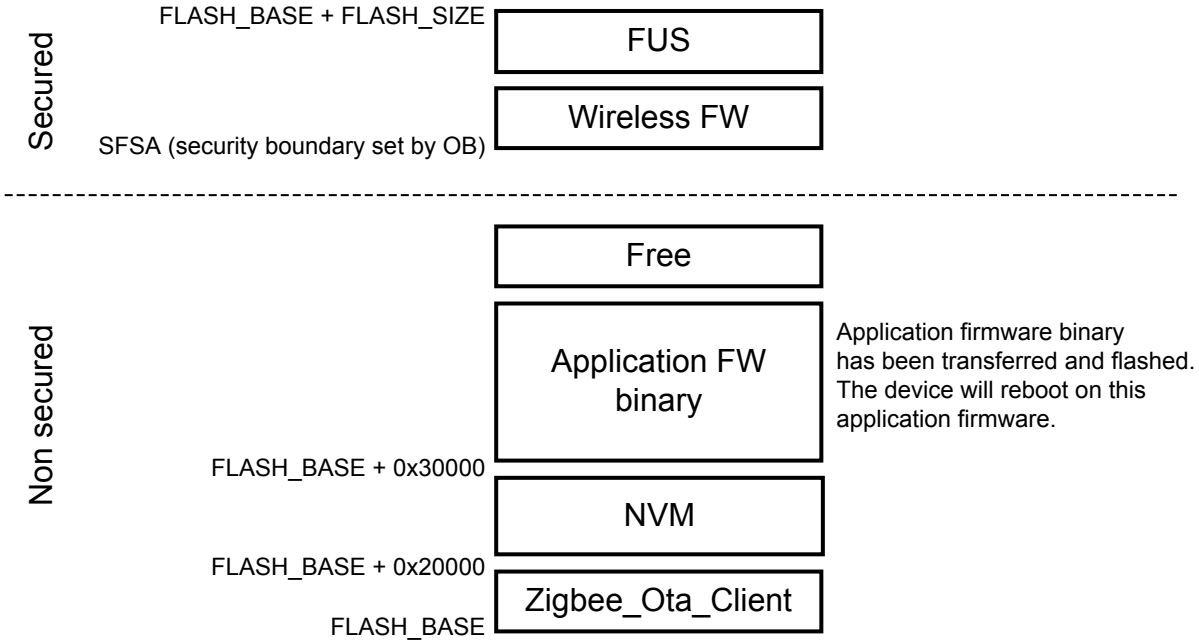
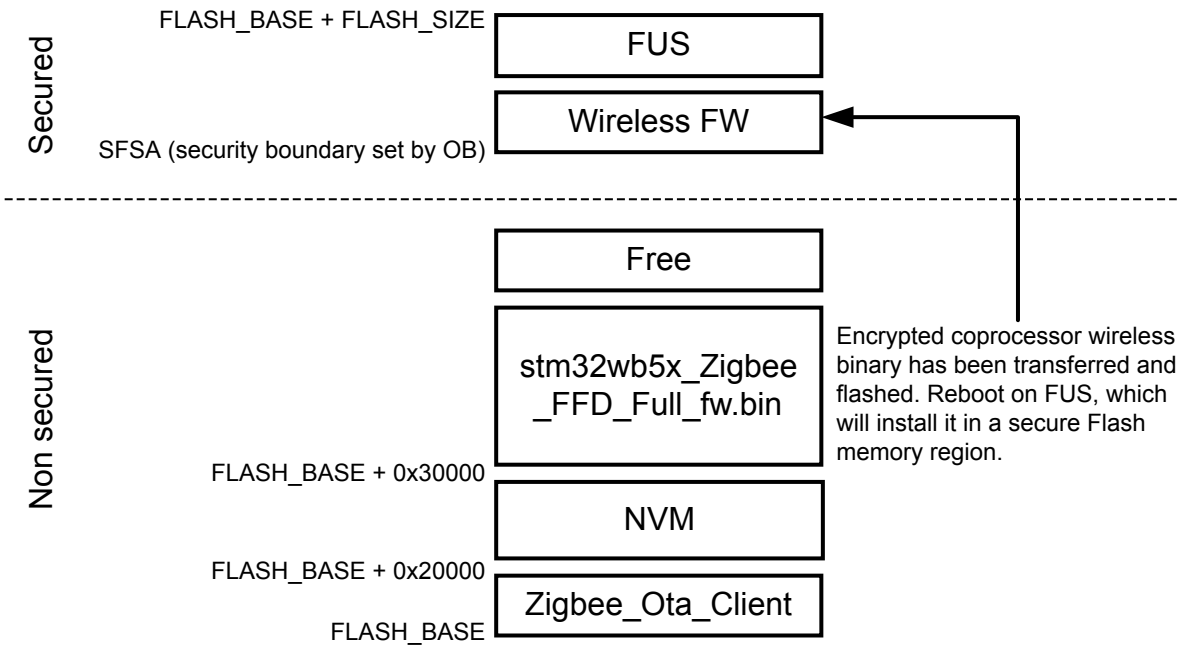


Figure 18. **FUOTA client Flash memory mapping after CPU2 binary transfer**



5.4.3 Zigbee FUOTA protocol

This is a STMicroelectronics proprietary protocol to update CPU2 wireless coprocessor binary or CPU1 FW application using Zigbee, based on ZCL OTA cluster.

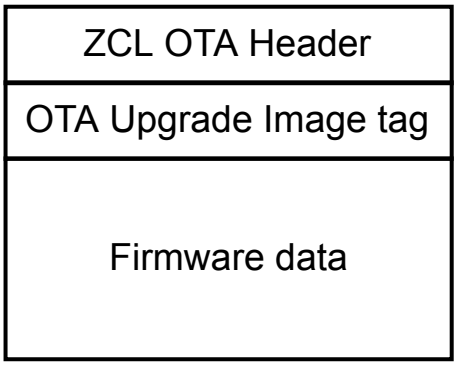
OTA file format

The ZCL OTA upgrade is based on a specific OTA file format made of:

- An OTA header
- Addition of OTA sub-elements

The figure below shows the OTA file format for the Zigbee FUOTA upgrade process

Figure 19. OTA file format for Zigbee FUOTA



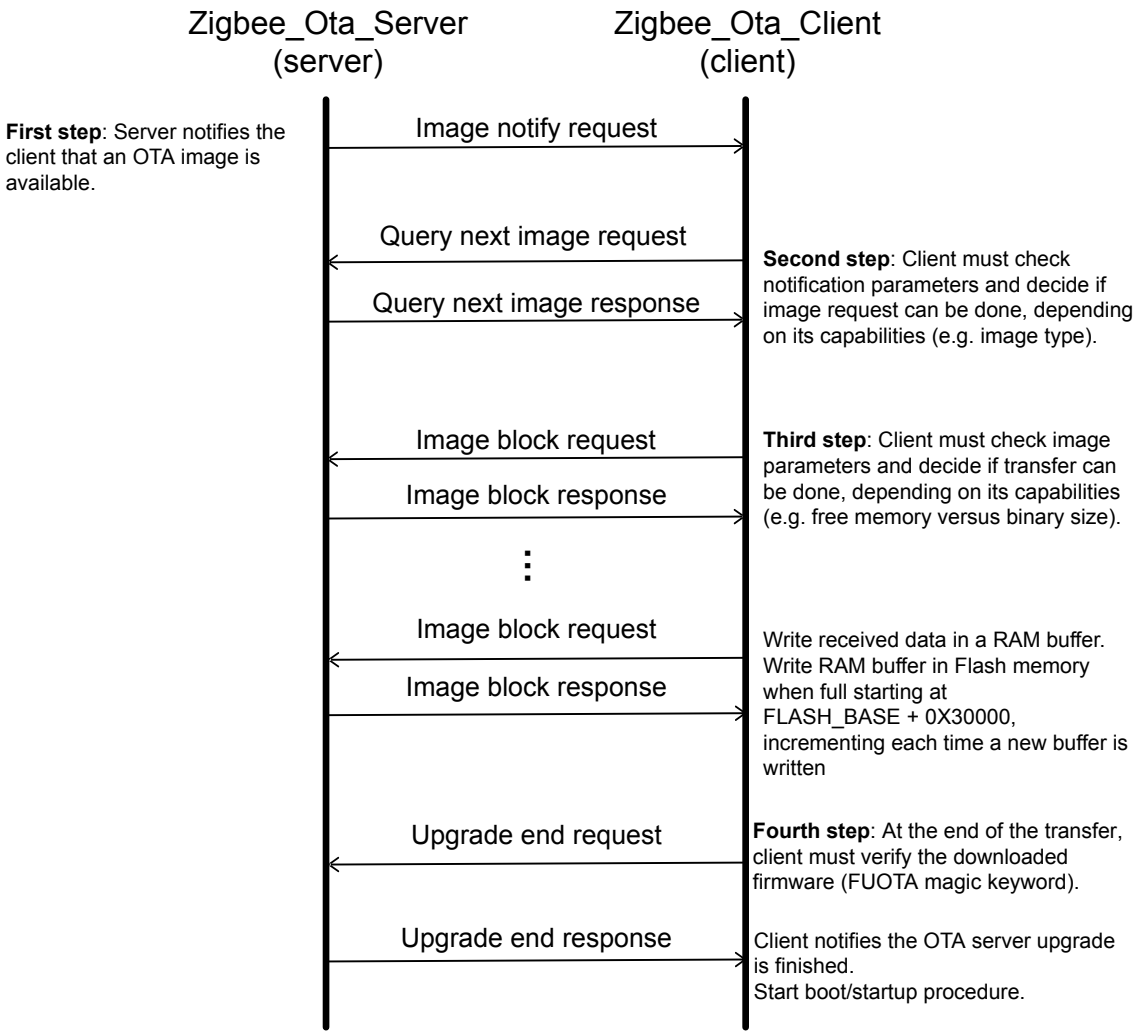
This OTA file format contains one sub-element (Upgrade image tag and associated binary firmware data).

For any OTA firmware stored in the OTA server Flash memory (see Section 5.4.2 Memory mapping for further details), the OTA file format is dynamically generated (OTA header, Upgrade Image tag information).

Zigbee FUOTA process

The figure below shows the steps to perform the firmware update transfer.

Figure 20. Zigbee FUOTA protocol



1. Server sends an Image Notify request to inform the client an image is available. It contains:
   - manufacturer ID
   - image type
   - new file version
2. Client decides to request this image (Query Next Image request) regarding notification parameters.
3. Query Next Image response is sent to the client. It contains:
   - manufacturer ID
   - image type
   - new file version
   - OTA image full size

Client checks parameters and decides if transfer can be done depending on its capabilities (for instance free space from image size).

1. During the OTA block transfer, the client store received firmware data blocks in a RAM cache and in Flash when it is full.
2. When transfer is finished (client has received all the necessary blocks regarding OTA image size), client validates the downloaded with the magic keyword.
3. The client notifies the server.
4. The upgrade finishes with the Upgrade End request and starts boot/startup procedure (regarding Upgrade End response parameters).

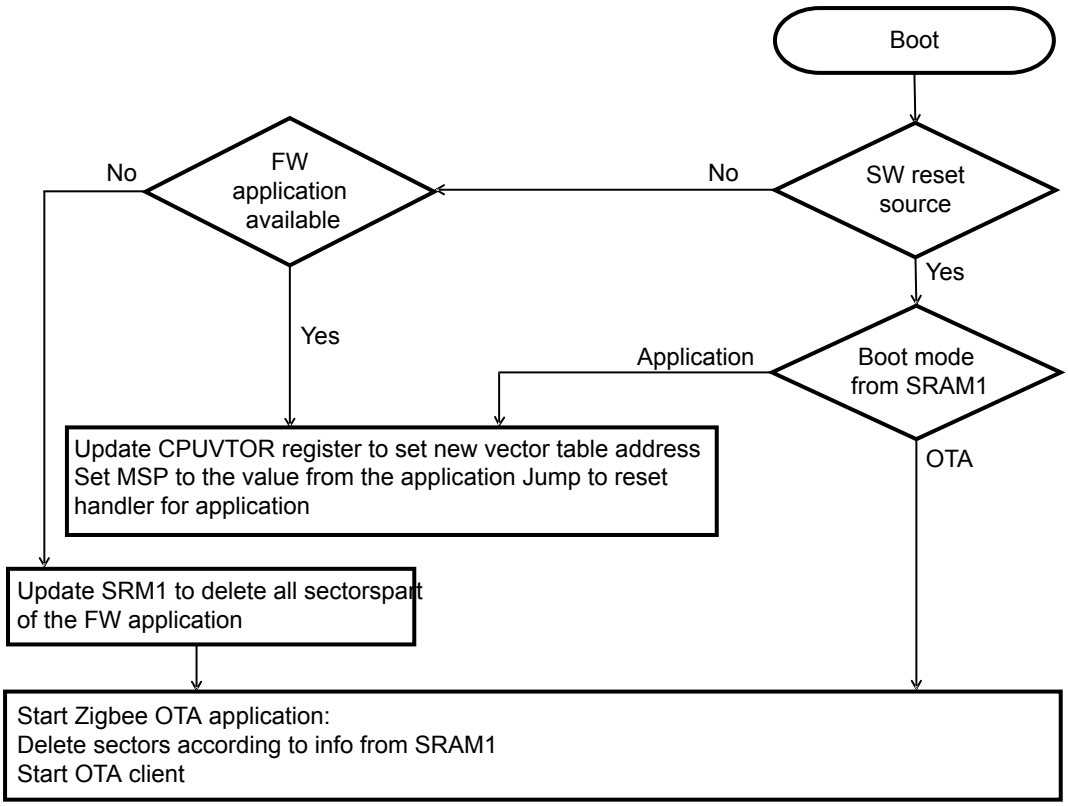
5.4.4 FUOTA application startup procedure

Once binary data has been transferred to the remote device (Zigbee FUOTA client), the startup procedure is different for the update of a CPU1 application or of CPU2 coprocessor wireless binary.

FUOTA for CPU1

On the client side after the binary transfer is completed, the process shown in Figure 21 takes place to jump on OTA specific application (for instance Zigbee_OnOff_Client_Router_Ota).

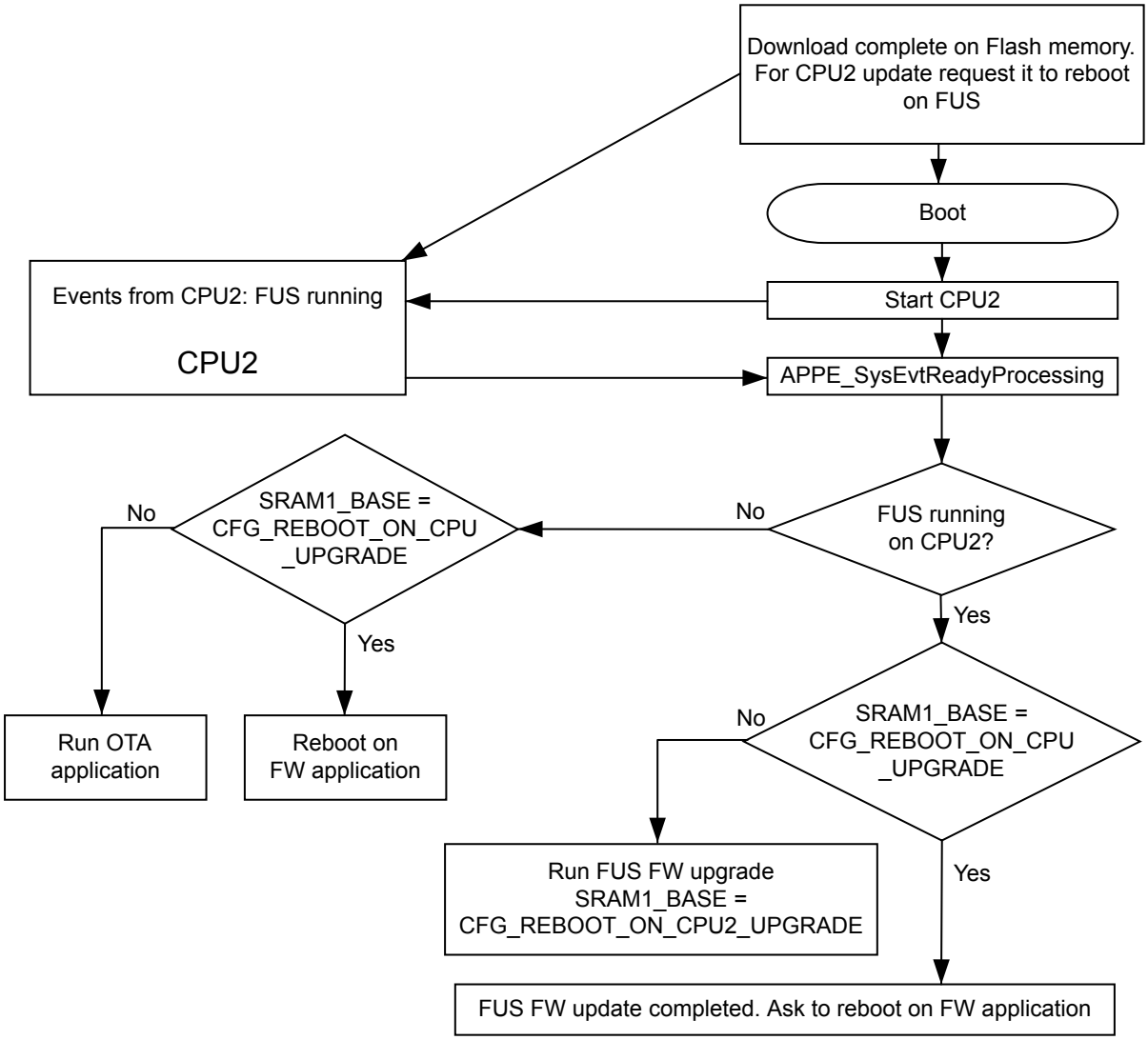
Figure 21. FUOTA startup procedure



FUOTA for CPU2

CPU2 update involves FUS (firmware upgrade service) software component, which is responsible to decrypt and install secure binary. Figure 22 describes the process.

Figure 22. **Update procedure**



5.4.5 Applications

ZigBee_OTA_Server_Coord

This application must be loaded on STM32WB 1Nucleo board acting as FUOTA server.

ZigBee_OTA_Client_Router

This application must be loaded on STM32WB Nucleo board acting as FUOTA client. In order for the client to be able to reconnect to the Zigbee network after an upgrade procedure, the Zigbee stack parameters have to be persisted. This is why persistent data are used with this application with both RAM cache and Flash storage.

ZigBee_OnOff_Client_Router_Ota

This application is almost identical to ZigBee_OnOff_Client_Router, the differences are:

- Persistent data with Flash storage are used. The NVM configuration is the same as in ZigBee_OTA_Client_Router application. So, after a reboot with this new application firmware, the previous Zigbee stack configuration can be restored. The updated device is able to reconnect to the coordinator's network.

- Use special tags (to manage end data transfer and data consistency):
  - TAG_OTA_END : The Magic Keyword value is checked in the ZigBee_Ota_Client_Router Application.
  - TAG_OTA_START : The Magic Keyword address shall be mapped at 0x140 from start of the binary image.

Therefore, by reading memory content at 0x140 it must be equal to Magic keyword value.

- Scatter file must be updated to place the sections above

Example for IAR:

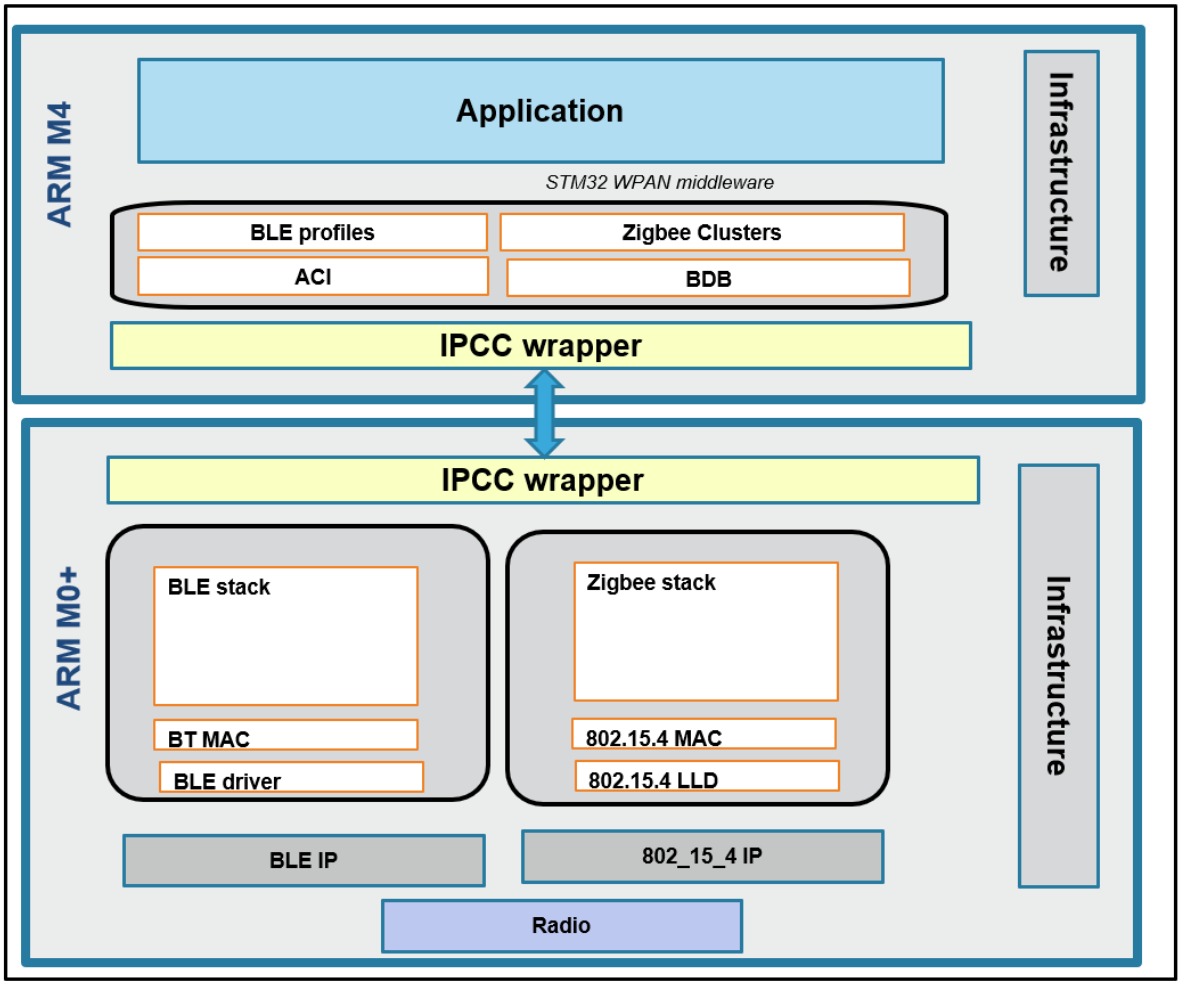
```
Vector table and ROM start @ moved to 0x08030000:
define symbol __ICFEDIT_intvec_start__ = 0x08030000;
define symbol __ICFEDIT_region_ROM_start__ = 0x08030000;
define region OTA_TAG_region = mem:[from
(__ICFEDIT_region_ROM_start__ + 0x140) to
(__ICFEDIT_region_ROM_start__ + 0x140 + 4)];
keep { section TAG_OTA_START};
keep { section TAG_OTA_END };
place in OTA_TAG_region { section TAG_OTA_START };
place in ROM_region { readonly, last section TAG_OTA_END };
```

5.5 Static concurrent mode

An example of static concurrent mode (BLE/Zigbee) is provided in the STM32WB firmware package.

This application is located under Projects\P-NUCLEO-WB55.Nucleo\Applications\BLE_Zigbee directory. When running this use case, the ‘static concurrent mode’ device can switch from BLE to Zigbee and vice-versa. This device can be connected through BLE to a smartphone running the "ST BLE Sensor" application and once the BLE activity is stopped. It can join a Zigbee network. Then, once the Zigbee application has been fully stopped, it is possible to go back to BLE again.

Figure 23. **Static concurrent mode on STM32WB**



Revision history

Table 7. **Revision history**

Date	Revision	Changes
23-Jul-2020	1	Initial release

Contents

List of tables

List of figures