RL78/G14

Firmware Update over LoRaWAN® Sample Application

Introduction

This application note describes a sample application to update a firmware over LoRaWAN®. The process to update the firmware over LoRaWAN is called as FUOTA (Firmware Updates Over The Air) and application layer protocols used for the update are standardized in the LoRa Alliance®.

This sample application supports the FUOTA process, which is intended for the end device in the LoRaWAN network.

Feature

- FUOTA sample application
  - API functions to handle the FUOTA related application layer protocols:
    Clock synchronization message
    Remote multicast setup
    Fragmented data block transport
  - Multicast sessions in Class B and Class C operation.
  - User command interface based on AT commands format.
- Firmware update sample application
  - Firmware update of the internal code flash memory.
- Tool
  - Converter tool to generate a firmware image data file from an object file.

Target Device

MCU: Renesas RL78/G14 (R5F104ML)

Transceiver: Semtech SX1261 or SX1262

Contents

1. Overview

FUOTA (Firmware Updates Over The Air) provides a function to remotely update a firmware over the wireless communication. This function is a key feature for IoT applications deployed widely in the field and required long term operation.

The LoRa Alliance standardized the FUOTA process utilizing the application layer protocols on top of the LoRaWAN protocol, such as the clock synchronization message protocol, the remote multicast setup protocol, and the fragmented data block transport protocol. These protocols can realize to deliver a firmware image (thereafter referred to as “F/W image”) to multiple devices at the time specified by an application server.

Figure 1-1 shows the overview of the FUOTA in the LoRaWAN network architecture. The application server requests the LoRaWAN network server to deliver the F/W image to an end device or a group of end devices with the time of the delivery. The LoRaWAN network server delivers the F/W image to end device(s) via the LoRaWAN wireless network according to the request.

The application layer protocols are utilized for the delivery from the LoRaWAN network server to the end device(s). The fragmented data block transport protocol provides the functions to divide the F/W image into the size less than the maximum size of a message that can be transmitted in the LoRaWAN network and reconstruct them into the F/W image. The remote multicast protocol provides the functions to simultaneously deliver the fragmented F/W image to a group of end devices. The clock synchronization protocol provides the functions to synchronize the end device’s clock to the LoRaWAN network’s GPS clock so that the end devices can prepare for the delivery and receive the fragmented F/W image.

Figure 1-2 shows the message exchange between LoRaWAN network server and an end device. First, the parameters required for the delivery are set to the end device using the application layer protocols. After that, the F/W image is delivered to the end device via the data fragment message of the fragmented data block transport protocol. The end device reconstructs the fragmented data into the F/W image, updates the internal firmware with the F/W image and reboots itself.

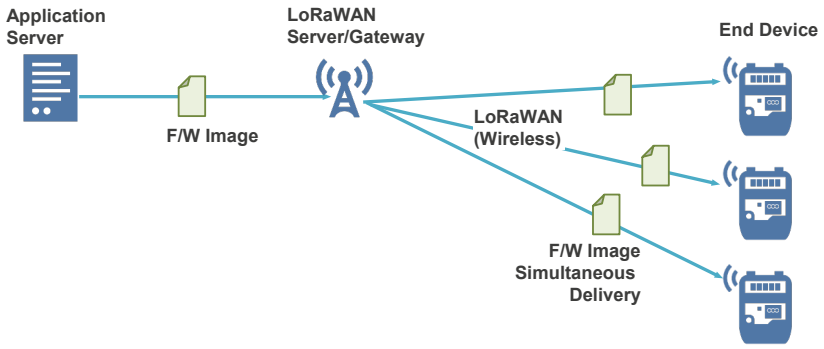


Figure 1-1 Overview of FUOTA in LoRaWAN network architecture

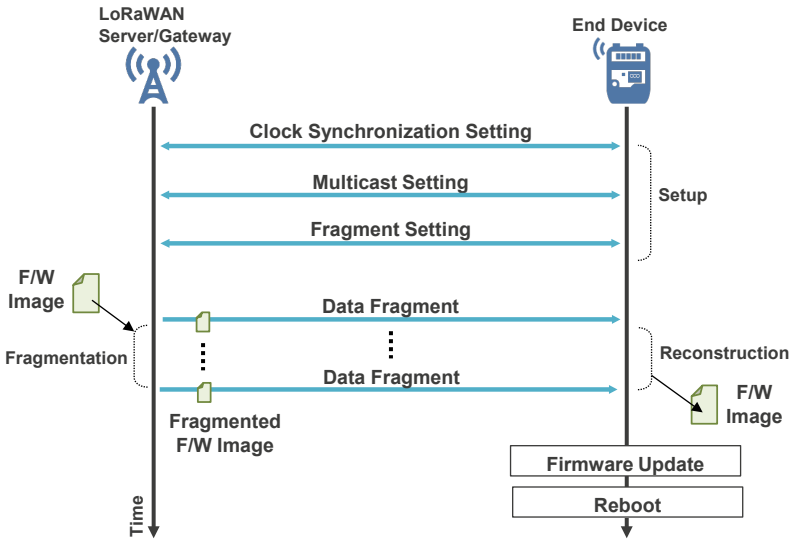


Figure 1-2 FUOTA message exchange between LoRaWAN network server and end device

This application note provides the sample application for FUOTA targeting for the end device based on RL78/G14 and the Semtech SX1261/62 transceiver for LoRa.

Figure 1-3 shows overview of the FUOTA process of this sample software for the end device. The FUOTA process can be achieved by two sample applications: The FUOTA sample application and the F/W update sample application.

The FUOTA sample application is to receive a F/W image over the LoRaWAN and its application layer protocols related to FUOTA, and to store it in the internal code flash memory. Once the F/W image is received, it switches to the F/W update sample application by the RL78 boot swap function.

The F/W update sample application is to update an end device's firmware using the F/W image. Once the update is completed, it switches to the updated FUOTA sample application by the RL78 boot swap function.

For details, the FUOTA sample application is described in chapter 2, the F/W update sample application is described in chapter 3 and the example operation of the end device is described in chapter 4.

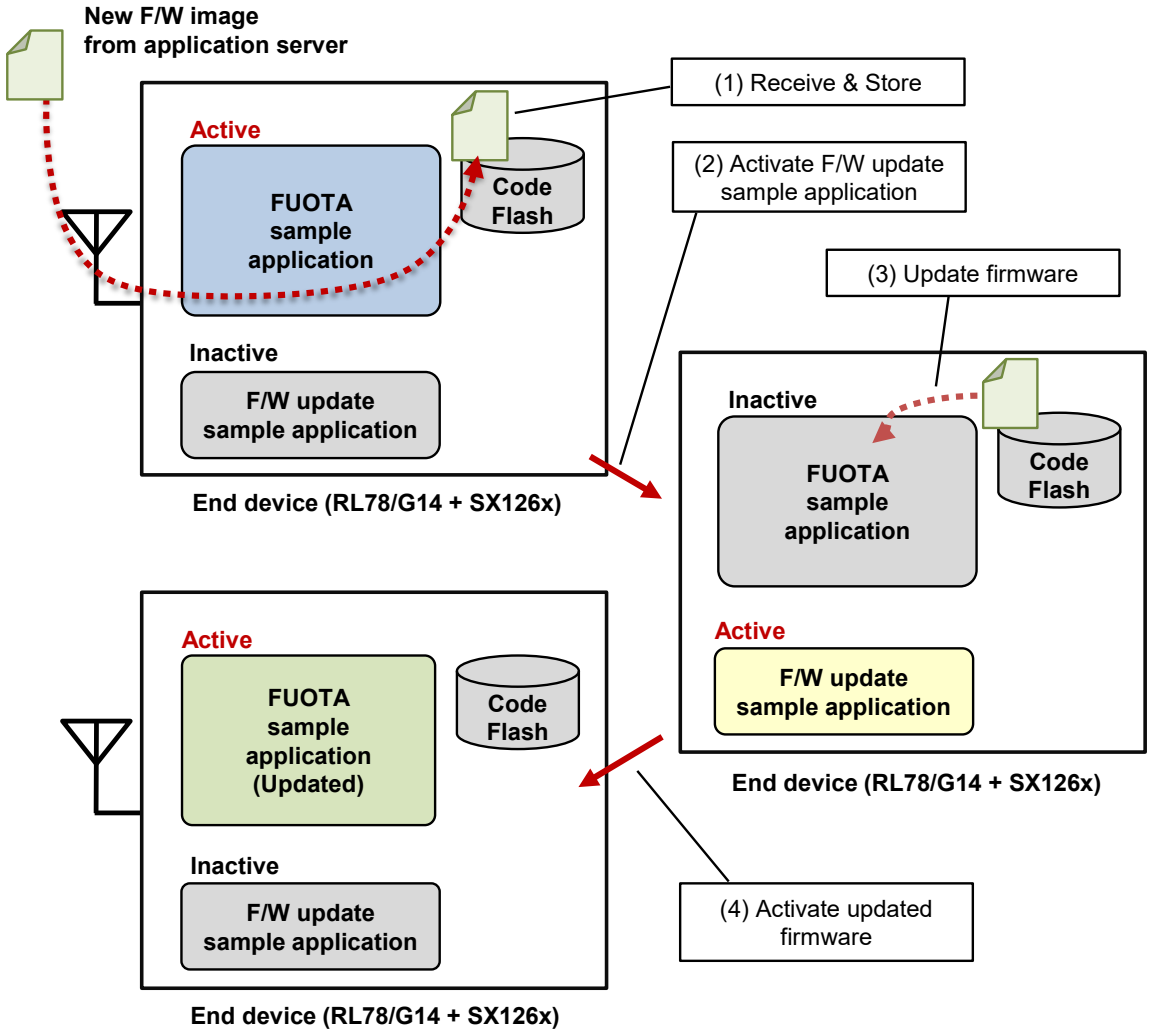


Figure 1-3 Overview of FUOTA process of this sample application for end device

1.1 Acronyms and abbreviations

Table 1-1 Acronyms and abbreviations

Acronyms	Description
FUOTA	Firmware Update Over-The-Air

1.2 Related Documentation

Table 1-2 Related Documentation

	Document No.	Title	Author	Language
[1]	R11AN0228EJ	LoRaWAN® stack reference guide	Renesas Electronics	English
[2]	R11AN0231EJ	LoRaWAN® Stack Sample Application	Renesas Electronics	English
[3]	R11QS0033EJ	LoRa®-based Wireless Software Package Quick Start Guide	Renesas Electronics	English

1.3 Directories

Table 1-3 shows a folder structure and what kind of codes are included in each folder

Table 1-3 Directories

Directories	Description
apps/LoRaFuotaSample	FUOTA sample application
apps/LoRaFuotaSample/add-in-fuota/r_cflash/FSL (*1)	Code Flash Library (*1)
apps/FWUpdateSample	FW update sample application codes
boards	Board specific codes
boards/mcu	MCU drivers
mac	LoRaWAN MAC stack
radio	Radio driver for LoRa®
peripherals	Security related codes
system	Utility codes
sytem/d_flash (*2)	EEPROM Emulation Library (*2)

CAUTION:
(*1) This folder is for containing the Code Flash Library to write to Code Flash memory. The library for testing, ‘Flash Self Programming Library Type01 Package Ver.3.00 for the RL78 Family [for the CA78K0R/CC-RL Compiler], is contained in advance. But when you start to develop a product, it is necessary to copy the newest Code Flash Library corresponding to your development environment to this folder.
The Code Flash Library is downloaded from Renesas website.

(*2) This folder is for containing the EEPROM Emulation Library and Data Flash Access Library to access Data Flash memory. The libraries for testing, ‘EEPROM Emulation Library Pack02 Package Ver.2.00(for CA78K0R/CC-RL Compiler) for RL78 Family’, are contained in advance. But when you start to develop a product, it is necessary to copy the newest library corresponding to your development environment to this folder.
The EEPROM Emulation Library and Data Flash Access Library are downloaded from Renesas website.

2. FUOTA sample application

This chapter describes the FUOTA sample application.

2.1 Overview of FUOTA sample application

2.1.1 FUOTA sample application block diagram

Figure 2-1 shows a block diagram of the FUOTA sample application. This sample application consists of the LoRaWAN stack, FUOTA and application layers. The FUOTA includes the application layer messaging packages over LoRaWAN shown in Table 2-1.

The clock synchronization package is used to synchronize an end device's clock to the LoRaWAN network's GPS clock. The remote multicast setup package is used to setup Class B or Class C multicast session for a group of end devices. The fragment data block transport package is used to receive fragmented data and reconstruct them into the original data.

These packages can realize an application server simultaneously send fragments of a firmware image to a group of end devices at the time notified from the server.

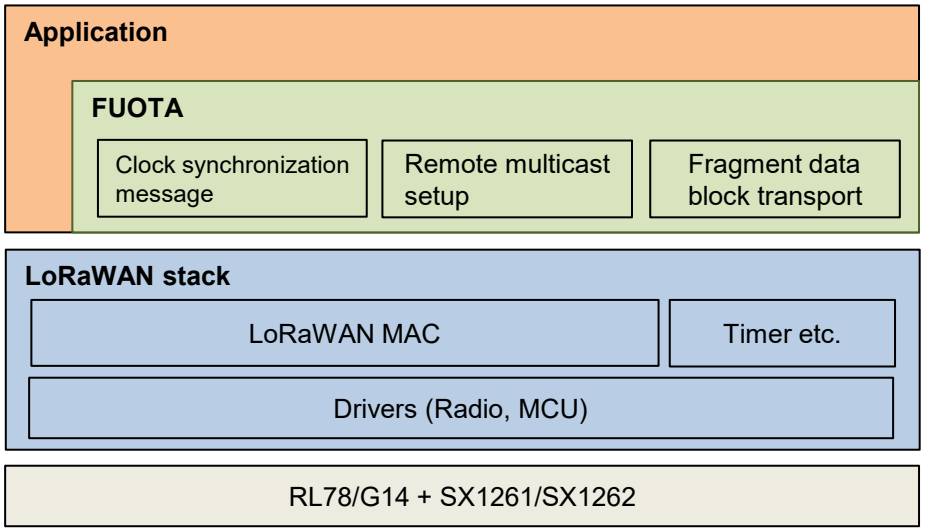


Figure 2-1 FUOTA sample application block diagram

Table 2-1 Application layer messaging package list

Package name	Version	Package ID	Package version	FPort
Clock Synchronization Message Package (*1)	v1.0.0	1	1	202
Remote Multicast Setup Package (*2)	v1.0.0	2	1	200
Fragmented Data Block Transport Package (*3)	v1.0.0	3	1	201

(*1) https://lora-alliance.org/resource-hub/lorawanr-application-layer-clock-synchronization-specification-v100
(*2) https://lora-alliance.org/resource-hub/lorawanr-remote-multicast-setup-specification-v100
(*3) https://lora-alliance.org/resource-hub/lorawanr-fragmented-data-block-transport-specification-v100

2.1.2 Resource usage (informative)

Table 2-2 shows the resources used by the FUOTA sample application including the LoRaWAN stack and drivers.

Table 2-2 Resources

MCU Peripherals
[LoRaWAN stack] - Real Time Clock (RTC) - Timer RJ - 12 bit Interval Timer - Serial Array Unit (SAU) - unit 3 channel 1 (CSI31) - I/O port - P26, P52, P53, P54, P02, P75, P03, P77 [FUOTA and application] - Serial Array Unit (SAU) - unit 0, channel 0 and 1 (UART0) - I/O port - P50, P51 (UART0 RXD0, TXD0)

2.1.3 Software architecture

Figure 2-2 shows a software architecture of the FUOTA layer.

The application can request to the FUOTA by the API functions of the FUOTA and receive the notification from the FUOTA by the callback functions of the FUOTA.

The FUOTA processes the received messages of the application layer protocols such as the clock synchronization, the fragment data block and the remote multicast setup according to the frame port number (FPort) when those are notified via the callback functions of the LoRaWAN stack. The FUOTA sends back the messages in response to the received messages if necessary via the API functions of the LoRaWAN stack.

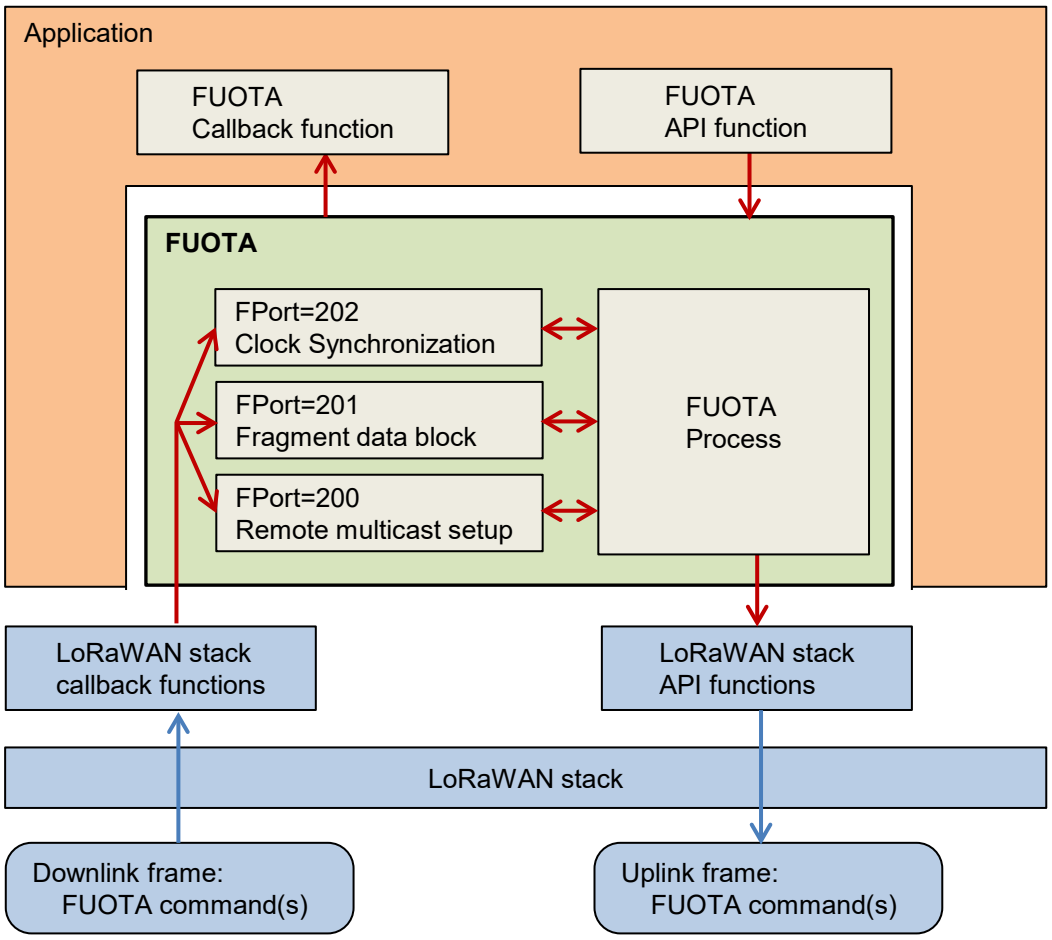


Figure 2-2 FUOTA software architecture

2.2 Macros

2.2.1 Configuration

Table 2-3 shows about the FUOTA configuration parameters.

These parameters can be defined in the file “LoRaFuotaConfig.h”.

Table 2-3 Macros for the FUOTA configuration

Macro	Description * [ ] indicates related application layer protocol	
FUOTA_CONFIG_RMTMC_MAX_MC_SESSION	**Type:** uint8_t (1 - 4)	**Default:** 1
	[Remote multicast setup] Maximum number of multicast sessions which can be used simultaneously.	
FUOTA_CONFIG_FRGMNT_MAX_FRAG_SESSION	**Type:** uint8_t (1 - 4)	**Default:** 1
	[Fragment data block] Maximum number of fragment sessions which can be used simultaneously.	
FUOTA_CONFIG_FRGMNT_MAX_DATABLK_SIZE	**Type:** uint32_t (depends on RAM)	**Default:** 16384
	[Fragment data block] Maximum size of a data block to be transported via a fragment session.	
FUOTA_CONFIG_FRGMNT_MAX_NBFRAG	**Type:** uint16_t (1 - 16383) (depends on RAM)	**Default:** 200
	[Fragment data block] Maximum number of fragments of a data block to be transported via a fragment session. Note: Maximum number of fragments (NbFrag) can be determined by possible minimum fragment size (fragSize): NbFrag = (Data block size + fragSize - 1) / fragSize The fragSize is determined by LoRaWAN network server. Please take this into account when setting this configuration.	
FUOTA_CONFIG_FRGMNT_MAX_NBLOST	**Type:** uint16_t (1 - 400) (depends on RAM)	**Default:** 50
	[Fragment data block] Maximum acceptable number of missed fragments of a data block to be transported via a fragment session. If the number of missed fragments exceeds this value, the data block cannot be reassembled.	

2.2.2 Information base (IB)

Table 2-4 shows the FUOTA information base (IB).

These are the parameters of FUOTA which can be get or set by the FUOTA API functions. See section 2.4.4 and 2.4.5.

Table 2-4 Macros for the FUOTA information base (IB)

Macro	Description * [ ] indicate related application layer protocol		
FUOTA_IB_CLKSNC_TIMEREQ_PERIOD_SEC	**Type:** uint32_t (0, 10 - 0x418390)	**Default:** 256	Read/Write
	[Clock synchronization] Periodicity of AppTimeReq command transmission in seconds. If it is set to 0, AppTimeReq is not transmitted. If it is set to other than 0, AppTimeReq is periodically transmitted according to the periodicity. Note: It could be updated when an application server requests to change the periodicity by DeviceAppTimePeriodicityReq command.		
FUOTA_IB_CLKSNC_TIMEREQ_MIN_PERIODICITY	**Type:** uint8_t (0 - 15)	**Default:** 0	Read/Write
	[Clock synchronization] Acceptable minimum periodicity of AppTimeReq transmission (*1).		
FUOTA_IB_CLKSNC_TIMEREQ_MAX_PERIODICITY	**Type:** uint8_t (0 - 15)	**Default:** 15	Read/Write
	[Clock synchronization] Acceptable maximum periodicity of AppTimeReq transmission (*1).		
FUOTA_IB_CLKSNC_FORCESYNC_PERIOD_SEC	**Type:** uint8_t (10 - 255)	**Default:** 60	Read/Write
	[Clock synchronization] Periodicity of AppTimeReq transmission in seconds when the specified number of transmissions are requested by ForceDeviceResyncReq command.		
FUOTA_IB_CLKSNC_TIMEANS_REQUIRED	**Type:** uint8_t (0, 1)	**Default:** 0	Read/Write
	[Clock synchronization] AnsRequired field of AppTimeReq to be transmitted from the end device It indicates whether to request answer in response to AppTimeReq. 1: Answer required 0: Answer not required		
FUOTA_IB_RMTMC_GENAPPKEY	**Type:** uint8_t [16]	**Default:** all 0	Read/Write
	[Remote multicast setup] GenAppKey used to derive multicast session keys, McNwkSKey and McAppSKey.		
FUOTA_IB_PROC_POLLING_PERIOD_SEC	**Type:** uint32_t (0, 10 - 0x418930)	**Default:** 0	Read/Write
	Periodicity of uplink frame transmission to receive downlink frame in seconds. If it is set to 0, uplink frame is not transmitted. If it is set to other than 0, uplink frame is periodically transmitted according to the period. Note: Periodical uplink frame transmission is suspended during multicast session is active.		
FUOTA_IB_PROC_POLLING_FPORT	**Type:** uint8_t (1 - 223)	**Default:** 223	Read/Write
	Frame port (FPort) value used for the uplink frame transmitted according to the setting of FUOTA_IB_PROC_POLLING_PERIOD_SEC.		

(*1) Actual periodicity of AppTimeReq transmission in seconds is 128 * (2 ^ Period) +/- rand(30).

2.3 Enumerations

2.3.1 FuotaStatus_t

This type is an enumeration containing the status of the operation of a FUOTA service.

Table 2-5 FuotaStatus_t

Enumerator	Description
FUOTA_STATUS_OK	Service processed successfully
FUOTA_STATUS_ERROR	Error - FUOA process was failed
FUOTA_STATUS_BUSY	Error - FUOTA and/or LoRaWAN stack is busy for other operations
FUOTA_STATUS_SERVICE_UNKNOWN	Error - Unknown request
FUOTA_STATUS_PARAMETER_INVALID	Error - Invalid parameter
FUOTA_STATUS_IB_READONLY	Error - IB is read only

2.4 FUOTA APIs

This section describes the API functions of FUOTA shown in Table 2 6.

Table 2-6 FUOTA APIs

Function	Description
FuotaInit	Initialize FUOTA.
FuotaStart	Start FUOTA
FuotaStop	Stop FUOTA
FuotaIbGetRequest	Information Base service to get attribute of FUOTA.
FuotaIbSetRequest	Information Base service to set attribute of FUOTA.
FuotaProcess	Process FUOTA interruption.
FuotaMcpsConfirm	Process MCPS-Confirm related to FUOTA.
FuotaMcpsIndication	Process MCPS-Indication related to FUOTA.
FuotaMlmeConfirm	Process MLME-Confirm related to FUOTA.
FuotaMlmeIndication	Process MLME-Indication related to FUOTA.

2.4.1 FuotaInit

FuotaStatus_t FuotaInit(FuotaEventCb_t *p_fuotaEventCb)		
This function initializes FUOTA. Event handler functions shall be specified in ‘p_fuotaEventCb’. Please call it before calling other FUOTA API functions.		
Parameters:		
P_fuotaEventCb	input	Pointer to the structure to set the FUOTA event handler functions. See section 2.5 for details.

Return:	
FUOTA_STATUS_OK	Request is finished successfully.
FUOTA_STATUS_PARAMETER_INVALID	Requested parameter is invalid.

2.4.2 FuotaStart

void FuotaStart(void)
This function starts FUOTA.
Parameters:
(none)
Return:
(none)

2.4.3 FuotaStop

void FuotaStop(void)		
This function stops FUOTA and initializes the information inside FUOTA except for the information base. It can be used to prevent RL78 boot swap process from being interrupted by FUOTA.		
Parameters:		
(none)		
Return:		
(none)		

2.4.4 FuotaIbGetRequest

FuotaStatus_t FuotaIbGetRequest(uint8_t ib, void *vpVal)		
This function is the FUOTA information base (IB) service to get attributes of the FUOTA. See section 2.2.2 for the IDs and types of IB.		
Parameters:		
ib	Input	ID of the information base
*vpVal	output	Destination of the attribute value

Return:	
FUOTA_STATUS_OK	Request is finished successfully.
FUOTA_STATUS_ERROR	Request cannot be accepted.
FUOTA_STATUS_PARAMETER_INVALID	Requested parameter is invalid.
FUOTA_STATUS_SERVICE_UNKNOWN	Requested IB is unknown.

2.4.5 FuotaIbSetRequest

FuotaStatus_t FuotaIbSetRequest(uint8_t ib, void *vpVal)		
This function is the FUOTA information base (IB) service to set attributes of the FUOTA. See section 2.2.2 for the IDs and types of IB.		
Parameters:		
ib	Input	ID of the information base
*vpVal	input	Source of the attribute value

Return:	
FUOTA_STATUS_OK	Request is finished successfully.
FUOTA_STATUS_ERROR	Request cannot be accepted.
FUOTA_STATUS_PARAMETER_INVALID	Requested parameter is invalid.
FUOTA_STATUS_SERVICE_UNKNOWN	Requested IB is unknown.
FUOTA_STATUS_IB_READONLY	Requested IB is read-only.
FUOTA_STATUS_BUSY	MAC is busy. Another service is running.

2.4.6 FuotaProcess

void FuotaProcess(void)		
This function processes pending events of FUOTA. Application shall periodically call this function in its main loop as short an interval as possible. Please call this function right after the LoRaMacProcess() function. (LoRaMacProcess() is an API function of LoRaWAN stack. See [1])		
Parameters:		
(none)		
Return:		
(none)		

2.4.7 FuotaMcpsConfirm

void FuotaMcpsConfirm(McpsConfirm_t *p_mcpsConfirm)		
This function processes the MCPS-Confirm message if it is related to FUOTA. Please call this function at the beginning of the MCPS-Confirm callback function. (See [1] about MCPS-Confirm callback function.)		
Parameters:		
p_mcpsConfirm	input	Pointer to MCPS-Confirm message, which is an argument of MCPS-Confirm callback function.
Return:		
(none)		

2.4.8 FuotaMcpsIndication

FuotaStatus_t FuotaMcpsIndication(McpsIndication_t *p_mcpsIndication)	
This function processes the MCPS-Indication message if it is related to FUOTA. Please call this function at the beginning of the MCPS-Indication callback function. (See [1] about MCPS-Indication callback function.)	
Parameters:	
p_mcpsIndication / input / Pointer to MCPS-Indication message, which is an argument of MCPS-Indication callback function.	
Return:	
FUOTA_STATUS_OK	The request is finished successfully.
FUOTA_STATUS_ERROR	Request cannot be accepted.
FUOTA_STATUS_COMMAND_ERROR	Command processing failed.
FUOTA_STATUS_BUSY	FUOTA is busy. Another service is running.

2.4.9 FuotaMlmeConfirm

void FuotaMlmeConfirm(MlmeConfirm_t *p_mlmeConfirm)		
This function processes the MLME-Confirm message if it is related to FUOTA. Please call this function at the beginning of the MLME-Confirm callback function. (See [1] about MLME-Confirm callback function.)		
Parameters:		
p_mlmeConfirm	input	Pointer to MLME-Confirm message, which is an argument of MLME-Confirm callback function.
Return:		
(none)		

2.4.10 FuotaMlmeIndication

void FuotaMlmeIndication(MlmeIndication_t *p_mlmeIndication)		
This function processes the MLME-Indication message if it is related to FUOTA. Please call this function at the beginning of the MLME-Indication callback function. (See [1] about MLME-Indication callback function.)		
Parameters:		
p_mlmeIndication	input	Pointer to MLME-Indication message, which is an argument of MLME-Indication callback function.
Return:		
(none)		

2.5 Callback handler functions (FuotaEventCb_t)

FuotaEventCb_t is a structure containing FUOTA event handler functions to notify application layers of the events.

Table 2-7 FuotaEventCb_t

Member (callback handler functions)	Description
void (*FuotaRmtMcSessionSetupIndication)(DeviceClass_t sessionClass, uint8_t mcGroupId, uint32_t timeToStartSec, uint32_t timeoutSec)	Pointer to callback function to be called when the time to start/end of the multicast session is scheduled.
void (*FuotaRmtMcSessionStartIndication)(DeviceClass_t sessionClass, uint8_t mcGroupId, uint32_t timeoutSec)	Pointer to callback function to be called when a multicast session is started.
void (*FuotaRmtMcSessionEndIndication)(DeviceClass_t sessionClass, uint8_t mcGroupId)	Pointer to callback function to be called when a multicast session end is ended.
FuotaStatus_t (*FuotaFrgmntSessionSetupIndication)(uint8_t fragIndex, uint32_t descriptor)	Pointer to callback function to be called before starting a fragment session.
void (*FuotaFrgmntDataBlockIndication)(uint8_t fragIndex, uint8_t *p_dataBlk, uint32_t dataBlkSize)	Pointer to callback function to be called when a data block is received.
void (*FuotaFrgmntSessionEndIndication)(uint8_t fragIndex)	Pointer to callback function to be called when a fragment session is ended.

2.5.1 FuotaRmtMcSessionSetupIndication

void (*FuotaRmtMcSessionSetupIndication)(DeviceClass_t sessionClass, uint8_t mcGroupId, uint32_t timeToStartSec, uint32_t timeoutSec)

This function will be called when the time to start/end of the multicast session is scheduled on reception of ‘MulticastClassCSessionReq’ or ‘MulticastClassBSessionReq’ command.
FUOTA will switch the device class to Class B or Class C when the multicast session is started. So, the application needs to prepare especially in case that the device class will be switched to Class B. If the application operates in Class A, it has to request the beacon acquisition to the LoRaWAN stack and start beacon tracking until multicast session is started.

Parameters:

sessionClass	input	Class of multicast session; CLASS_C or CLASS_B.
mcGroupId	input	Group ID
timeToStartSec	input	Time to start multicast session in seconds.
timeoutSec	input	Timeout of the session from start in seconds.

Return (Output):

(none)

2.5.2 FuotaRmtMcSessionStartIndication

void (*FuotaRmtMcSessionStartIndication)(DeviceClass_t sessionClass, uint8_t mcGroupId, uint32_t timeoutSec)		
This function notifies when the multicast session is started, and the device class is changed to the class of multicast session.		
Parameters:		
sessionClass	input	Class of multicast session; CLASS_C or CLASS_B.
mcGroupId	input	Group ID
timeoutSec	input	Timeout of the session in seconds.
Return (Output):		
(none)		

2.5.3 FuotaRmtMcSessionEndIndication

void (*FuotaRmtMcSessionEndIndication)(DeviceClass_t sessionClass, uint8_t mcGroupId)		
This function notifies when multicast session is ended, and the device class is returned to the class before the multicast session is started.		
Parameters:		
sessionClass	input	Class of multicast session; CLASS_C or CLASS_B.
mcGroupId	input	Group ID
Return (Output):		
(none)		

2.5.4 FuotaFrgmntSessionSetupIndication

FuotaStatus_t (*FuotaFrgmntSessionSetupIndication)(uint8_t fragIndex, uint32_t descriptor)		
This function will be called when the parameters used for the fragment session are notified on reception of 'FragSessionSetupReq' command by FUOTA. Application has to check parameters and decide if the fragment session can be started. The result of the decision is set to the status parameter of 'FragSessionSetupAns' command.		
Parameters:		
fragIndex	input	Index of fragment session.
descriptor	input	Descriptor; parameter in FragSessionSetupReq command. As for this FUOTA sample application, the information related to the firmware image is supposed to be set in the descriptor. So, application needs to check the value of the descriptor. See section 2.7.2 for details.
Return (Output):		
FUOTA_STATUS_OK		Fragment session can be started.
FUOTA_STATUS_ERROR		Fragment session cannot be started.

2.5.5 FuotaFrgmntDataBlockIndication

void (*FuotaFrgmntDataBlockIndication)(uint8_t fragIndex, uint8_t *p_dataBlk, uint32_t dataBlkSize)		
This function notifies the reception of a data block. Application can store the data block to the internal code flash memory. See section 2.7.2 for details.		
Parameters:		
fragIndex	input	Index of fragment session.
p_dataBlk	input	Pointer to the received data block.
dataBlkSize	input	Size of the received data block.
Return (Output):		
(none)		

2.5.6 FuotaFrgmntSessionEndIndication

void (*FuotaFrgmntSessionEndIndication)(uint8_t fragIndex)		
This function notifies when the fragment session is ended and deleted.		
Parameters:		
fragIndex	input	Index of fragment session.
Return (Output):		
(none)		

2.6 FUOTA related commands sequence, usage of API and callback functions

2.6.1 Flow of FUOTA processing

Figure 2-3 shows a basic flow diagram of FUOTA process.

After the FUOTA is initialized and started, the FUOTA related commands can be processed by passing the MCPS indication (downlink data) notified from the LoRaWAN stack.

Application needs to call FuotaProcess() function periodically for FUOTA to process its events; the FUOTA related command transmissions and timer interruptions.

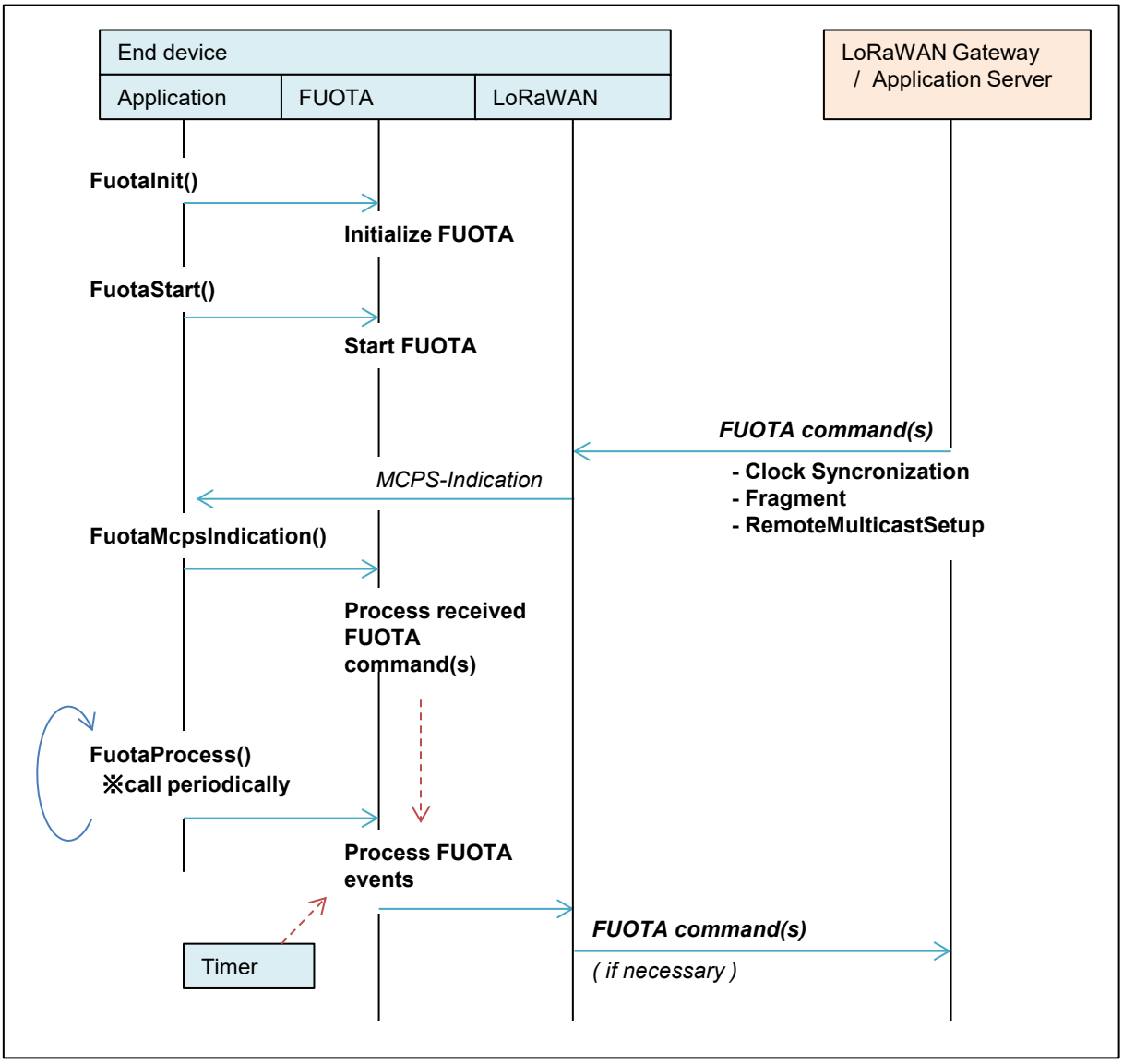


Figure 2-3 Flow of FUOTA processing

2.6.2 Clock Synchronization

Figure 2-4 shows a flow diagram of the clock synchronization between an end-device's clock and the LoRaWAN network's GPS based clock.

When the FUOTA is started, FUOTA starts to send AppTimeReq command periodically according to the IB, FUOTA_IB_CLKSNC_TIMEREQ_PERIOD_SEC. See section 2.2.2.

The FUOTA controls the process of the clock synchronization and no event is notified to the application.

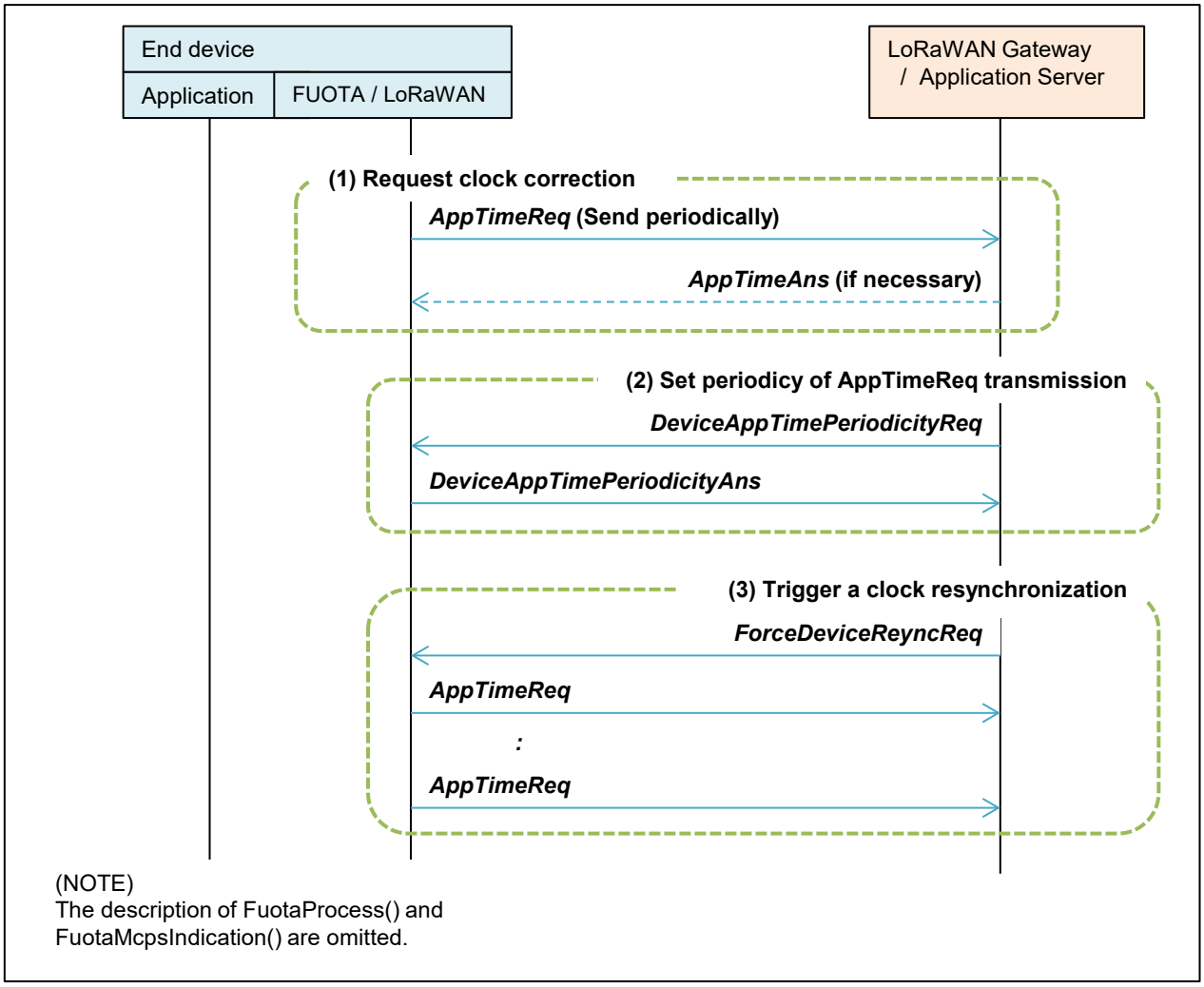


Figure 2-4 Clock synchronization

2.6.3 Remote multicast setup

Figure 2-5 shows a flow diagram of the remote multicast setup.

The FUOTA notifies the application of the start and the end time of the multicast session when the application server requests the end device to schedule the start and end of a multicast session.

There are two type of multicast sessions: Class C and Class B. When a multicast session is started, FUOTA switches the device class to Class C or Class B. So, the application needs to prepare to start the multicast session; especially in case of Class B session, application which operates in Class A has to request the beacon acquisition to the LoRaWAN stack and start beacon tracking before the multicast session is started.

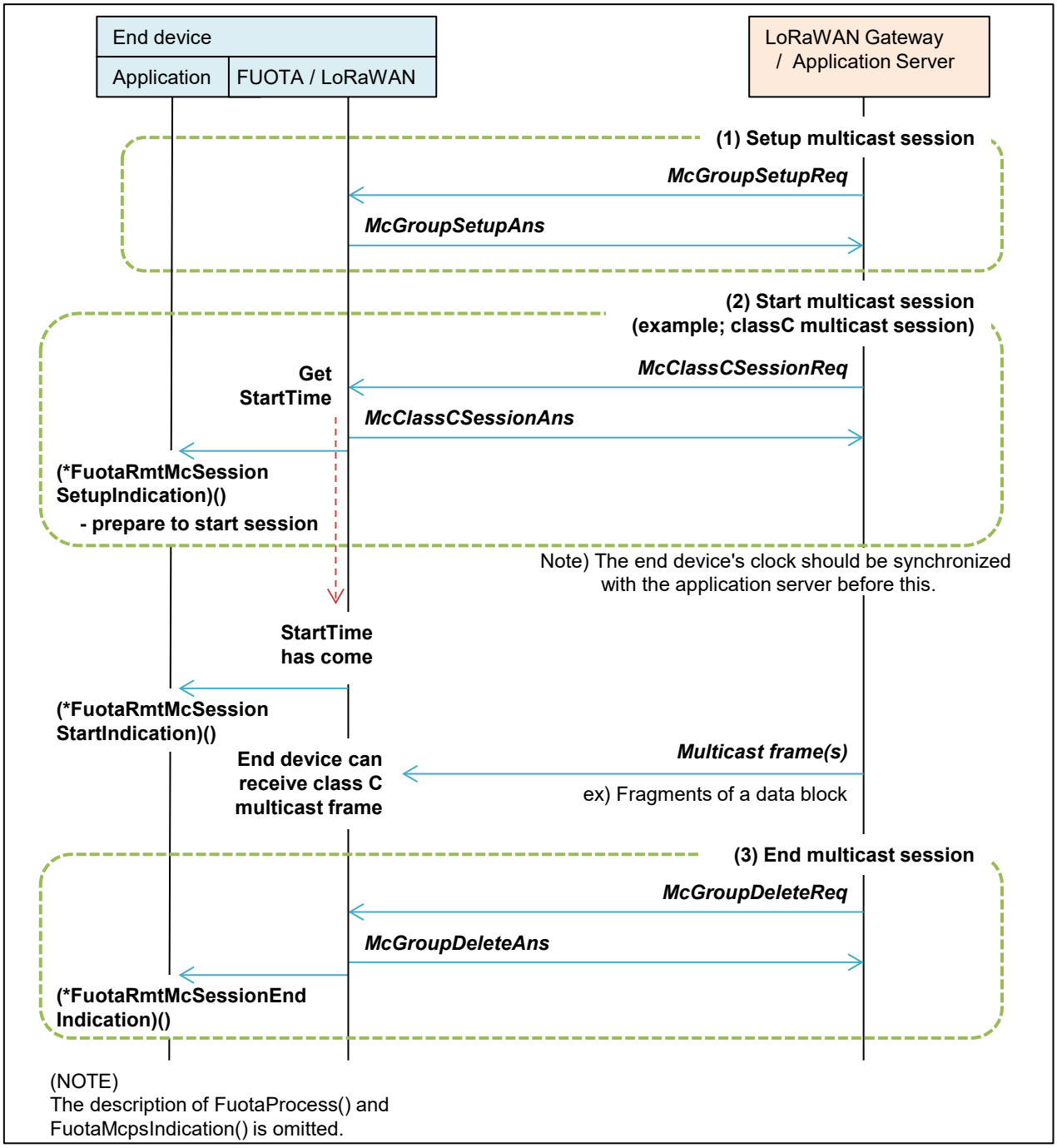


Figure 2-5 Remote multicast setup

2.6.4 Fragment data block transport

Figure 2-6 shows a flow diagram of the fragment data block transport.

The FUOTA notifies the application of the start and end of the fragment session when the application server requests the end device to start and end a fragment session. Also, FUOTA notifies the application of the reception of a data block when it is reconstructed by the fragments during the fragment session.

The application needs to store the data block to the code flash memory to update the firmware later. See section 2.7 for details.

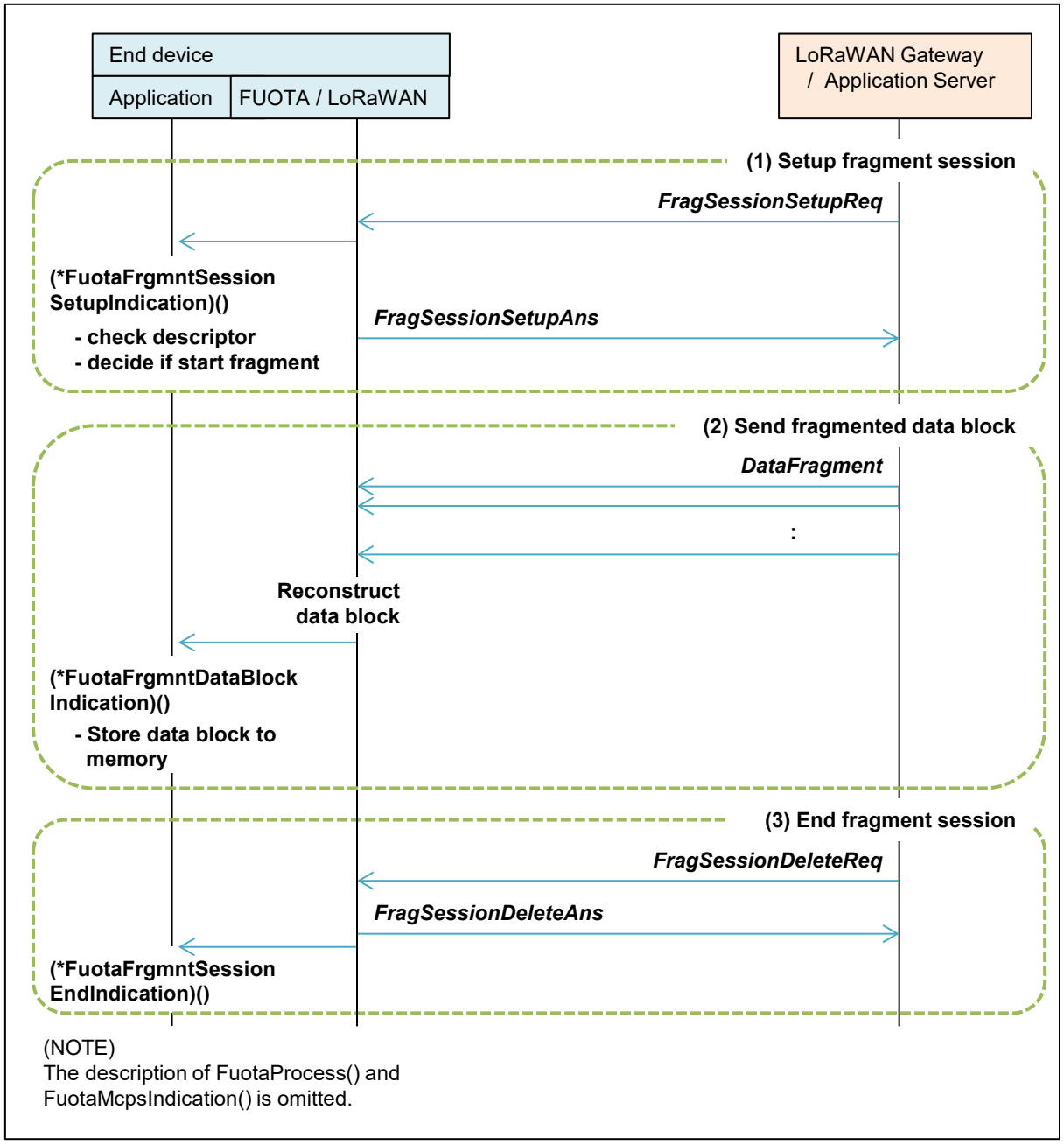


Figure 2-6 Fragment data block

2.7 FUOTA sample application

This section describes about the application layer specification of the FUOTA sample application software.

2.7.1 Overview

Figure 2-7 shows the sequence of the firmware update process according to the following (Step 1) to (Step 4).

The FUOTA sample application supports the following (Step 1) and (Step 2), and the F/W update sample application supports the following (Step 3) and (Step 4). There are functions prepared to write the data blocks to the code flash memory for (Step 1), and to activate the F/W update sample application for (Step 2). Refer to the section 2.7.2 and 2.7.3 respectively.

The FUOTA sample application can be controlled by the AT commands defined in [2] and additional FUOTA related AT commands. Refer to the section 2.7.4 for details.

(Step 1) Receives the new F/W image and stores it to the code flash memory [See section 2.7.2]

The FUOTA sample application starts to receive the new F/W image from the application server. The FUOTA layer processes the received F/W image, which could consist of some data blocks.

The application layer stores the data blocks notified from the FUOTA to the internal code flash memory.

(Step 2) Validates the new F/W image and activates the F/W update sample application [See section 2.7.3]

After the validation of the stored new F/W image, the FUOTA sample application activates the F/W update sample application by the RL78 boot swap function. The F/W update sample application is supposed to be pre-programed in the code flash memory.

(Step 3) Update the firmware using the new F/W image [See chapter 3]

The F/W update sample application validates the stored new F/W image and updates the internal firmware using the F/W image.

(Step 4) Activates the updated firmware [See chapter 3]

After the updates of the firmware, the F/W update sample application activates the updated FUOTA sample application by the RL78 boot swap function.

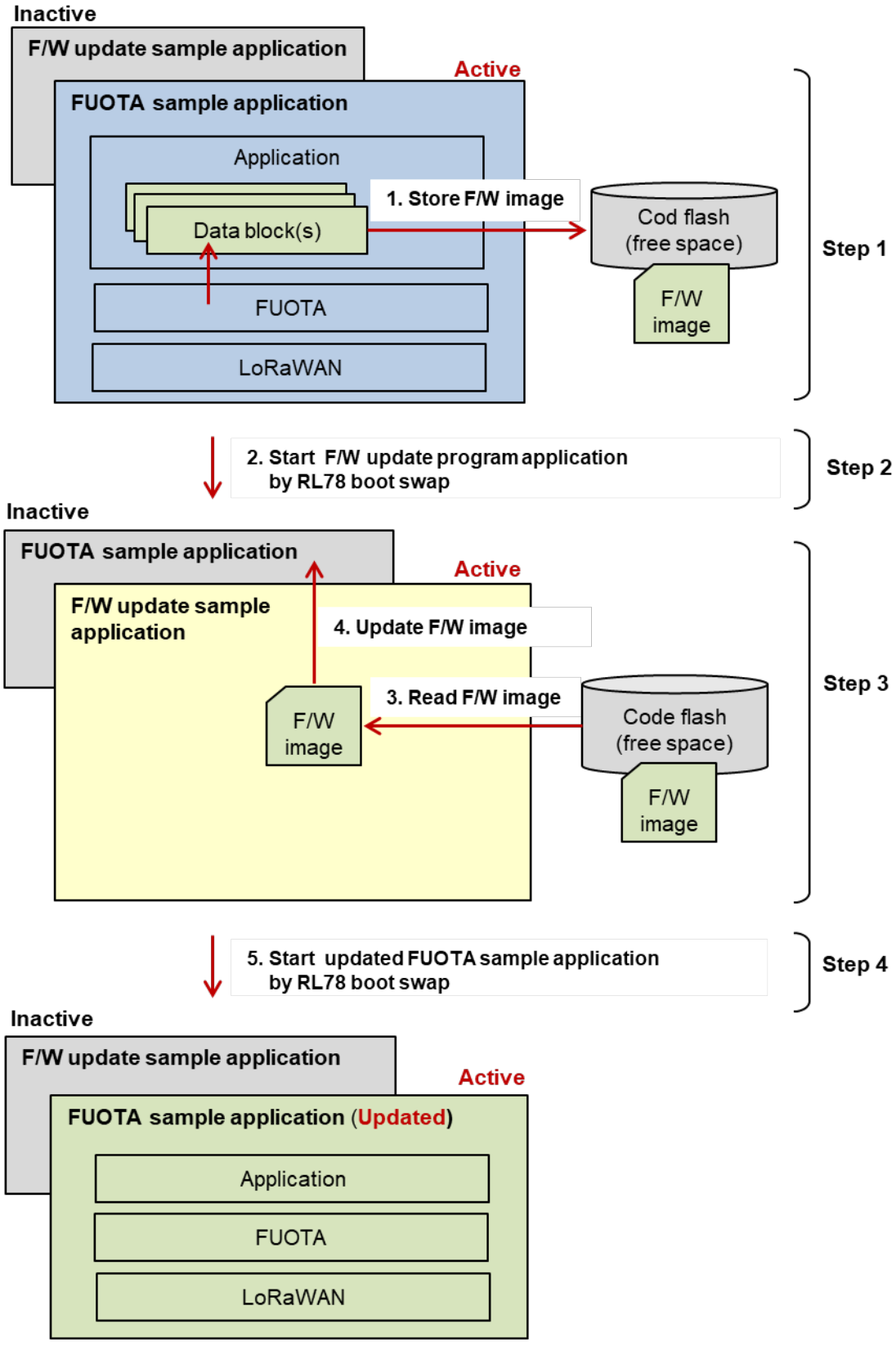


Figure 2-7 Sequence of the firmware update

2.7.2 Functions to write F/W image to code flash memory

The FUOTA sample application receives the data blocks divided from the new F/W image via the fragment sessions. After that, it needs to write the data blocks to the internal code flash memory.

There are two functions prepared for the application to write the data blocks. Figure 2-8 shows the usage of the functions.

FuotaUpdateStatus_t AppFuotaUpdateStartFragment(uint32_t fragmentDescriptor)		
This function checks the fragmentDescriptor that is the descriptor field of a received FragSessionSetupReq command and prepares to write a data block to the code flash memory. Please call this function within the callback function 'FuotaFrgmntSessionSetupIndication'. See section 2.5.4.		
Parameters:		
fragmentDescriptor	input	Specify the descriptor, which is the 2nd argument of the callback function 'FuotaFrgmntSessionSetupIndication'. Note: The descriptor is the parameter of the "FragmentSessionSetupReq" command sent by the application server. The F/W image could be divided into several data blocks. So, the FUOTA sample application supposes that the descriptor indicates the index number starting from 0 and is incremented for each fragment session.
Return:		
FUOTAUPDT_STATUS_OK (= FUOTA_STATUS_OK)		Processed successfully.
FUOTAUPDT_STATUS_ERROR (= FUOTA_STATUS_ERROR)		Process was failed.

FuotaUpdateStatus_t AppFuotaUpdateStoreFwImage(uint8_t *p_dataBlk, uint16_t dataSize)		
This function writes a data block indicated by the callback function 'FuotaFrgmntDataBlockIndication' to the code flash memory. Please call this function within the callback function. See section 2.5.5.		
Parameters:		
p_dataBlk	input	Specify the pointer to the received data block, which is the 2nd argument of the callback function 'FuotaFrgmntDataBlockIndication'.
dataSize	input	Specify the size of the received data block, which is the 3rd argument of the callback function 'FuotaFrgmntDataBlockIndication'.
Return:		
FUOTAUPDT_STATUS_OK (= FUOTA_STATUS_OK)		Processed successfully.
FUOTAUPDT_STATUS_ERROR (= FUOTA_STATUS_ERROR)		Process was failed.

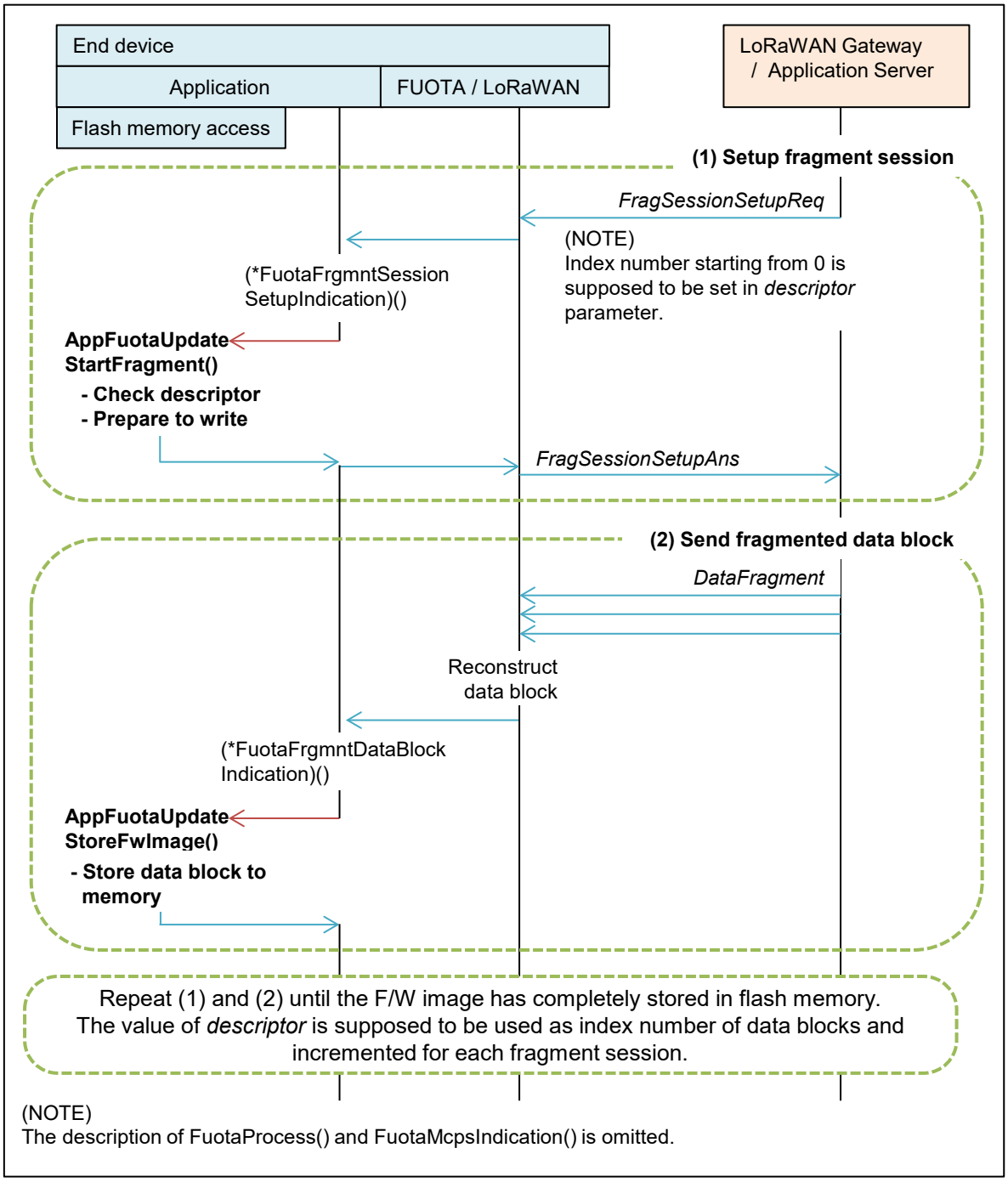


Figure 2-8 Example usage of functions to write F/W image to the code flash memory

2.7.3 Functions to activate F/W update sample application

There are two functions for application to get the status of the FUOTA process and activate the F/W update sample application. Figure 2-9 shows the example of the function usage.

uint8_t AppFuotaUpdateGetStatus(void)	
This function gets the status of storing F/W image	
Parameters:	
(none)	
Return:	
FUOTAUPDT_STATE_SUCCESS	Complete F/W image is written to the code flash memory.
FUOTAUPDT_STATE_NONE	No F/W image.
FUOTAUPDT_STATE_INITIAL	No F/W image. A fragment session is requested to be setup.
FUOTAUPDT_STATE_RUNNING	F/W image is not yet completed. Some data blocks of F/W image are written to the code flash memory.
FUOTAUPDT_STATE_FAILED	- Failed to write F/W image to the code flash memory. - Address of the F/W image storage area is invalid. - Succeeding index number set in *descriptor* is not sequential. - Information of headers in F/W image is invalid.

FuotaUpdateStatus_t AppFuotaUpdateStartFwUpdate(AppFuotaPreUpdate_t p_preUpdateCbFunc)		
This function activates the F/W update sample application if the complete F/W image is stored in the code flash memory and it is validated successfully. This function is returned only if the RL 78 boot swap to activate the F/W update sample application could not be performed.		
Parameters:		
p_preUpdate	input	Callback function that is called before starting F/W update sample application by the RL78 boot swap. If NULL is specified, the callback function is not called.
Return:		
FUOTAUPDT_STATUS_BUSY (= FUOTA_STATUS_BUSY)		LoRaWAN stack is busy for other processing. Note: Validation of F/W image was successfully.
FUOTAUPDT_STATUS_ERROR (= FUOTA_STATUS_ERROR)		Failed to activate the F/W update sample application. Note: The detail reason for the failure can be retrieved with AppFuotaUpdateGetStatus() function. If this return value is FUOTAUPDT_STATE_SUCCESS, it indicates the validation of the F/W image was failed.

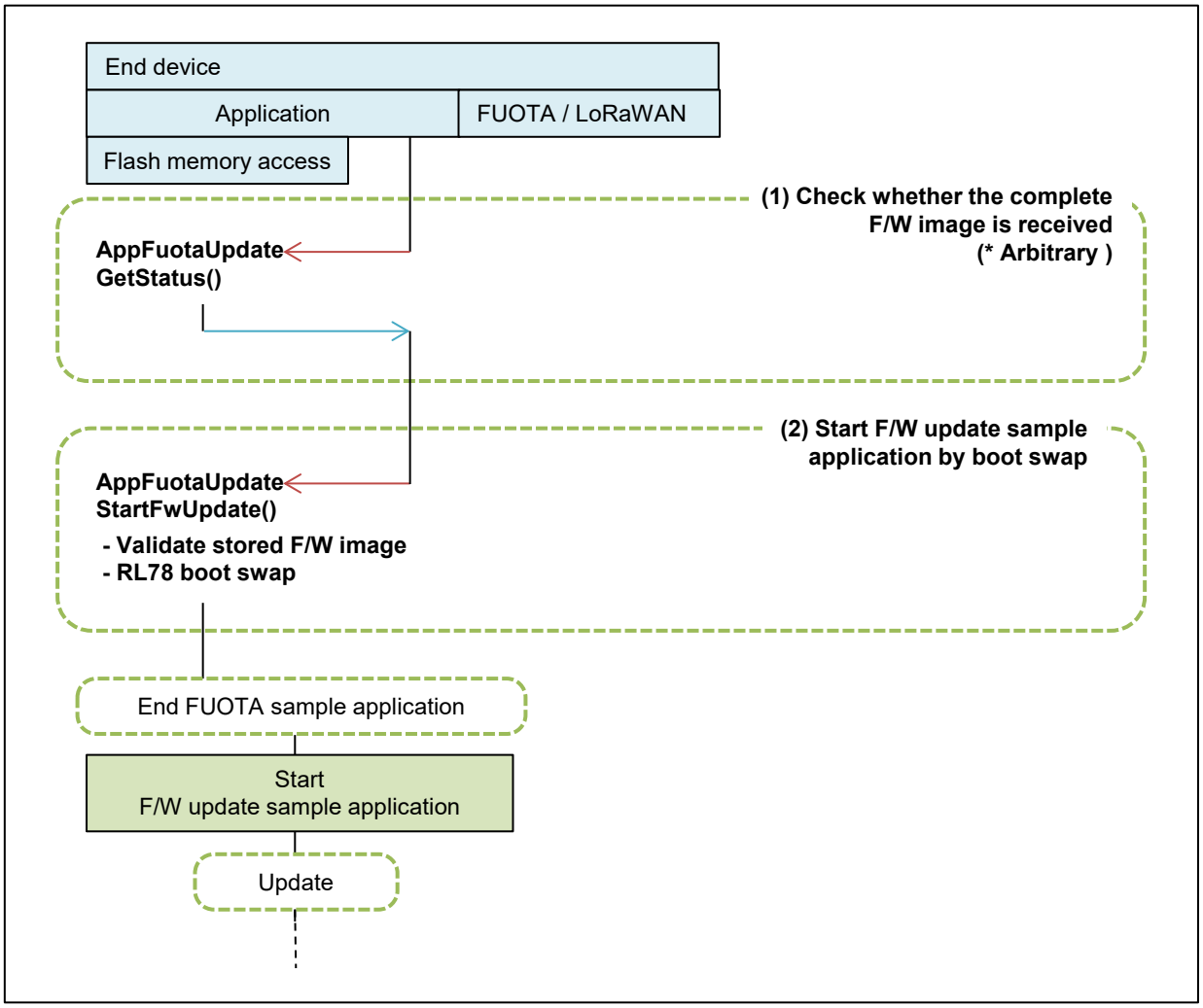


Figure 2-9 Example usage of functions to activate F/W update sample application

2.7.4 AT commands for the FUOTA sample application

This section describes about AT commands for the FUOTA sample application.

The FUOTA sample application can be controlled by the AT commands defined in [2] and additional FUOTA related AT commands specified as follows.

(1) AT commands request for FUOTA

AT Command	(Top) Syntax / (Bottom) Description
AT+FUOTASTART	AT+FUOTASTART - Start FUOTA process by calling API function, FuotaStart(). - This command shall be issued after network join.
AT+FUOTASET	ATFUOTASET=<IB>,<Val> <IB> = ID of IB (Hexadecimal without prefix) <Val>= Value of IB (Decimal or 16-byte Hexadecimal without prefix) - Set a FUOTA related IB by calling API function, FuotaIbSetRequest().
AT+FUOTAGET	ATFUOTAGET=<IB> <IB> = IB of ID (Hexadecimal without prefix) - Get a FUOTA related IB by calling API function, FuotaIbGetRequest().
AT+FUOTAUPDT	AT+FUOTAUPDT - Activate the F/W update sample application by the RL78 boot swap function and update the FUOTA sample application using the F/W image stored in the code flash memory. - After the update, activate the updated FUOTA sample application by the RL78 boot swap function.

ID of IB to set/get by AT+FUOTASET and AT+FUOTAGET

IB	ID	Example
FUOTA_IB_CLKSNC_TIMEREQ_PERIOD_SEC	0x10	AT+FUOTASET=10,100 Set to 100 seconds as transmission interval of AppTimeReq
FUOTA_IB_CLKSNC_TIMEREQ_MIN_PERIODICITY	0x12	AT+FUOTASET=12,1 Set to 1 as the minimum periodicity of AppTimeReq transmission
FUOTA_IB_CLKSNC_TIMEREQ_MAX_PERIODICITY	0x13	AT+FUOTASET=13,14 Set to 14 as the minimum periodicity of AppTimeReq transmission
FUOTA_IB_CLKSNC_FORCESYNC_PERIOD_SEC	0x14	AT+FUOTASET=14,200 Set 200 seconds as transmission interval of AppTimeReq when ForceDeviceResyncReq is requested.
FUOTA_IB_RMTMC_GENAPPKEY	0x20	AT+FUOTASET=20,1111....11 Set GenAppKey to 1111...11 (16 byte) for multicast setup.
FUOTA_IB_PROC_POLLING_PERIOD_SEC	0xF0	AT+FUOTASET=F0,300 Set 300 seconds as transmission interval of polling uplink to get downlink
FUOTA_IB_PROC_POLLING_FPORT	0xF1	AT+FUOTASET=F1,55 Set FPort to 55 for the polling uplink to get downlink

(2) AT command indication from FUOTA

AT command	(Top) Description / (Bottom) Parameter
+FUOTAIND:<indication>,<param1>,<param2>,<param3>,<param4> <indication> FUOTA evet indication type <param1> - <param4> Depends on indication value.	Event indication from FUOTA layer - When <indication> = 0: Indicate an event of the RemoteMulticast session setup <param1> Session Class. 1=ClassB / 2=ClassC <param2> Multicast Group ID <param3> Seconds to start session <param4> Seconds to timeout session - When <indication> = 1: Indicate an event of the RemoteMulticast session start. <param1> Session Class. 1=ClassB / 2=ClassC <param2> Multicast Group ID <param3> Seconds to timeout session - When <indication> = 2: Indicate an event of the RemoteMulticast session end. <param1> Session Class. 1=ClassB / 2=ClassC <param2> Multicast Group ID - When <indication> = 128: Indicate an event that F/W image is ready. <param1> - <param4> (Omitted)

3. F/W update sample application

3.1 Overview of F/W update sample application

Figure 3-1 shows the sequence of the firmware update.

The F/W update sample application is activated after the F/W image is received and stored in the internal flash by the FUOTA sample application.

The F/W update sample application processes the following steps.

(Step 1) Update the firmware of the FUOAT sample application by referring to the information contained in the F/W image. [See section 3.2 and 3.3]

(Step 2) Activate the updated firmware by the RL78 boot swap.

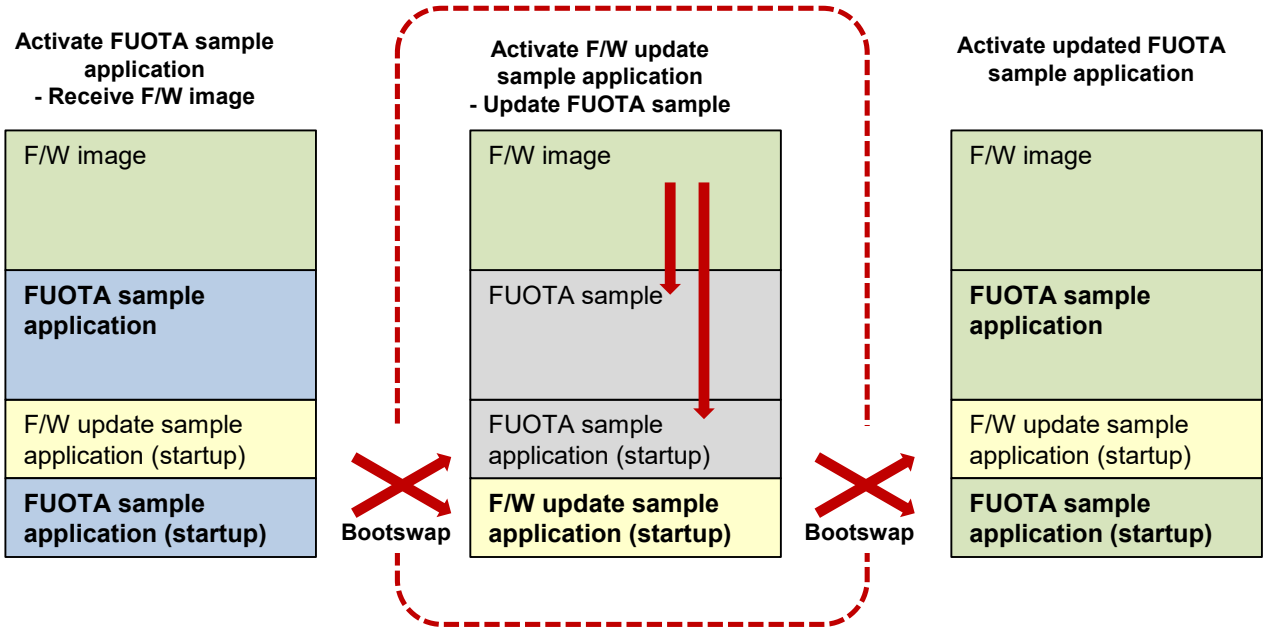


Figure 3-1 Overview of F/W update sample application

3.2 F/W image format

Figure 3-2 shows the F/W image format that the F/W update sample application supports.

The F/W image is a binary data of the firmware and consists of some image blocks. Each block includes the information such as the start address to write the program code, the code size, and the code data.

Table 3-1 shows the detail format of the F/W image.

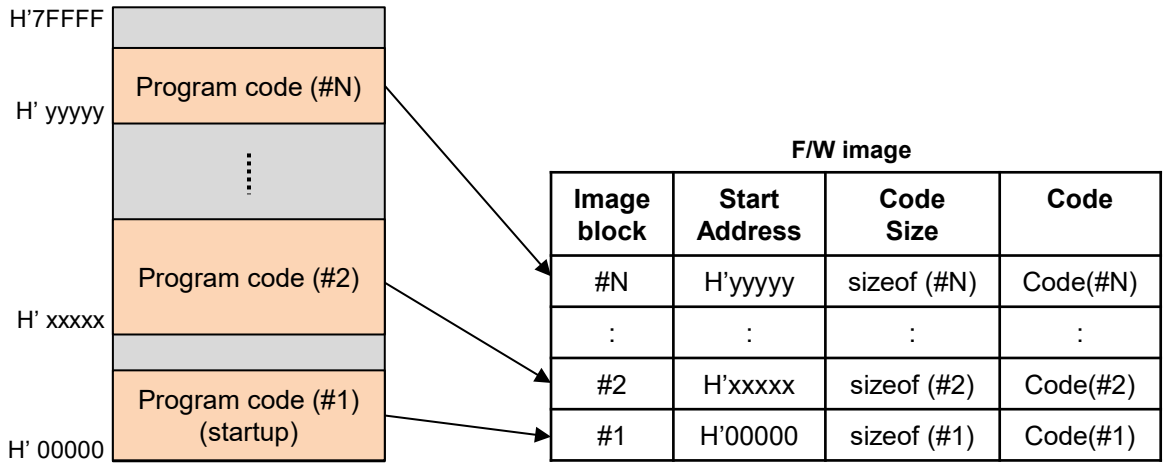


Figure 3-2 F/W image

Table 3-1 Format of F/W image

Contents		Size (Byte)	Description
F/W image Header	ImageBlockNum	1	Total number of image blocks (=N)
	ImageBlockIndex	1	Index of image bock; here set 0.
	ImageVersion	4	Version of F/W image
	ImageSize	4	Total size of F/W image
	ImagePriority	1	Priority (Arbitrary use)
	_reserved	1	(Reserved to adjust alignment)
	ImageVerify	32	ImageVerify is used to check F/W image validity. Upper 4 byte of ImageVerify is a checksum. Lower 28 byte of ImageVerify is reserved for future extension.
Image block #1	ImageBlockNum	1	Total number of image blocks (=N)
	ImageBlockIndex	1	Index of image bock (=1)
	CodeAddress	4	Address to write code #1.
	CodeSize	4	Size of code #1.
	Code	(CodeSize)	Code #1. Note) If CodeSize is odd number, 0x00 padding for alignment.
:	:	:	:
Image block #N	ImageBlockNum	1	Total number of image blocks (=N)
	ImageBlockIndex	1	Index of image bock (=N)
	CodeAddress	4	Address to write code #N.
	CodeSize	4	Size of code #N
	Code	(CodeSize)	Code #N (see note above.)

3.3 Firmware update using F/W image

The F/W update sample application updates the firmware according to the information of the image blocks in the F/W image except the case the code needs to be written in the boot cluster 0 area where the address range is from H'000000 to H'000FFF, as shown in Figure 3-3. The F/W update sample application writes the code to the boot cluster 1 area where the address range is from H'001000 to H'001FFF so that the code is mapped to the boot cluster 0 area when the new firmware is activated by the RL78 boot swap. (Figure 3-3)

After the update is finished, the F/W update sample application activates the new firmware by the RL78 boot swap.

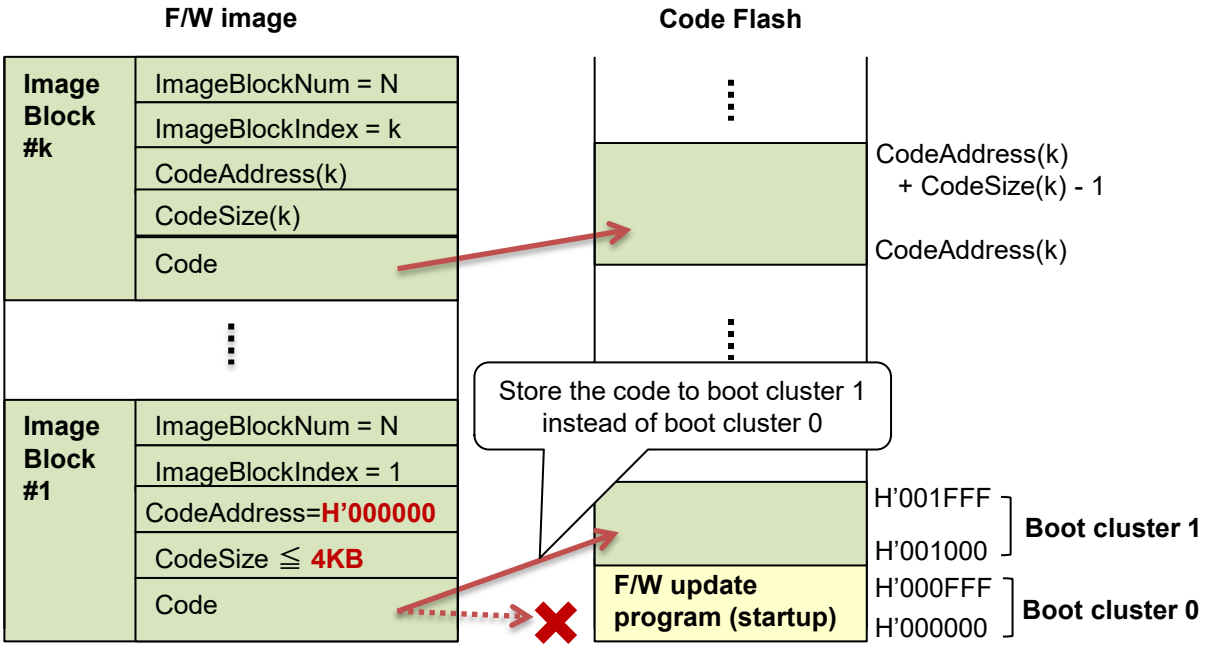


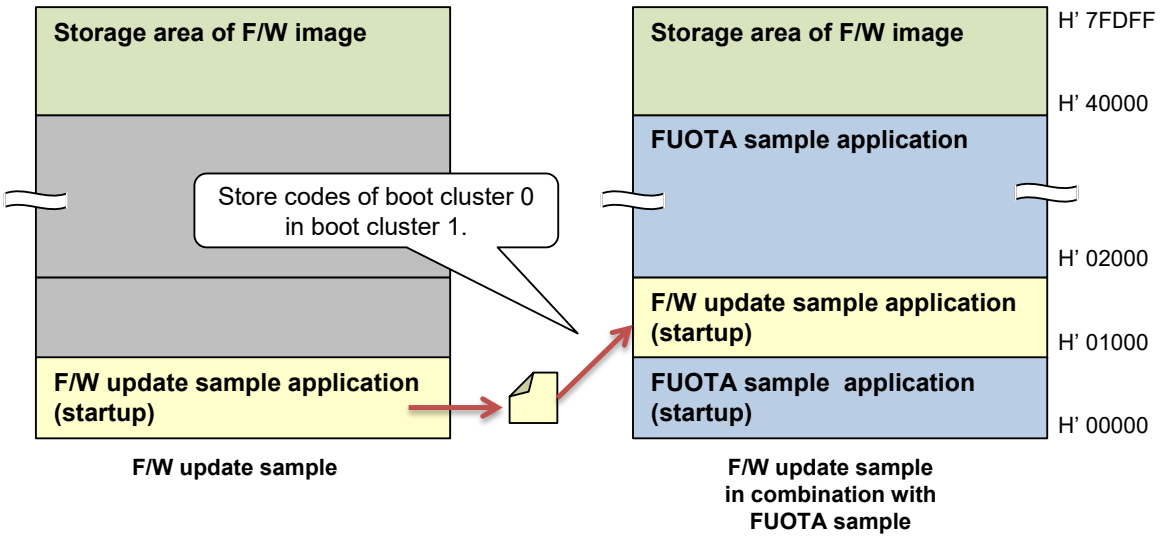
Figure 3-3 Firmware update using F/W image

3.4 ROM mapping

Figure 3-4 shows the ROM mapping.

The left side of the figure shows the ROM mapping in case the F/W update sample application is activated.

The right side of the figure shows the ROM mapping in case the FUOTA sample application is activated, and the F/W update sample application is inactivated. In order for the FUOTA sample application to activate the F/W update sample application by the RL78 boot swap, the startup code of the F/W update sample application must be stored at the boot cluster 1 (H'1000 – H'1FFF). So, the code of the FUOTA sample application cannot be allocated to the boot cluster 1.



NOTE: H'7FE00- H'7FFFF are reserved for the debug monitor area

Figure 3-4 ROM mapping

4. Example operations of FUOTA sample application

This section describes the example operations of FUOTA sample application.

The section 4.1 describes preparation required for the end device. The section 4.2 describes preparation required for the LoRaWAN network server. The section 4.2.2 describes the example operations using the AT commands to setup/run FUOTA operations and update the firmware by the received new F/W image.

4.1 Preparation for end device

Two sample applications, the FUOTA sample application and the F/W update sample application, needs to be built and programmed.

4.1.1 Building of FUOTA sample application

The FUOTA sample application needs to be built using one of the following project files. The object file of the program image will be made as ‘LoRaFuotaSample.mot’. The object file of the program and the symbols will also be made as ‘LoRaFuotaSample.x’ in case of e2studio, and ‘LoRaFuotaSample.abs’ in case of CS+.

[Project file]

(1) e2studio

samples\project\e2studio\rl78g14fpb_sx126x\LoRaFuotaSample\

(2) CS+

samples\project\csplus\rl78g14fpb_sx126x\LoRaFuotaSample\LoRaFuotaSample.mtpj

4.1.2 Building of F/W update sample application

The FW update sample application needs to be built using one of the following project files. The object file of the program image will be made as ‘FWUpdateSample_dlAddr01000.mot’. The program is mapped from the address of H’01000 to be programmed to the boot cluster 1 area. Refer to the section 3.4 for details.

[Project file]

(1) e2studio

samples\project\e2studio\rl78g14fpb_sx126x\FWUpdateSample\

(2) CS+

samples\project\csplus\rl78g14fpb_sx126x\FWUpdateSample\FWUpdateSample.mtpj

4.1.3 Programing of object files to code flash memory

The two object files built in the section 4.1.1 and 4.1.2 need to write to the code flash memory of RL78/G14. The operations for the flash programing are shown in the following (1) and (2)

(1) When Renesas Flash Programmer (RFP) is used for the flash programing

If only the flash programing is necessary, RFP can be used. Figure 4-1 shows the configuration of RFP. This configuration is necessary before writing the object files. After that, the object file of only program image (.mot) of the FUOTA sample application needs to be download first, and the object file of only program image (.mot) of the F/W update sample application needs to be download secondly.

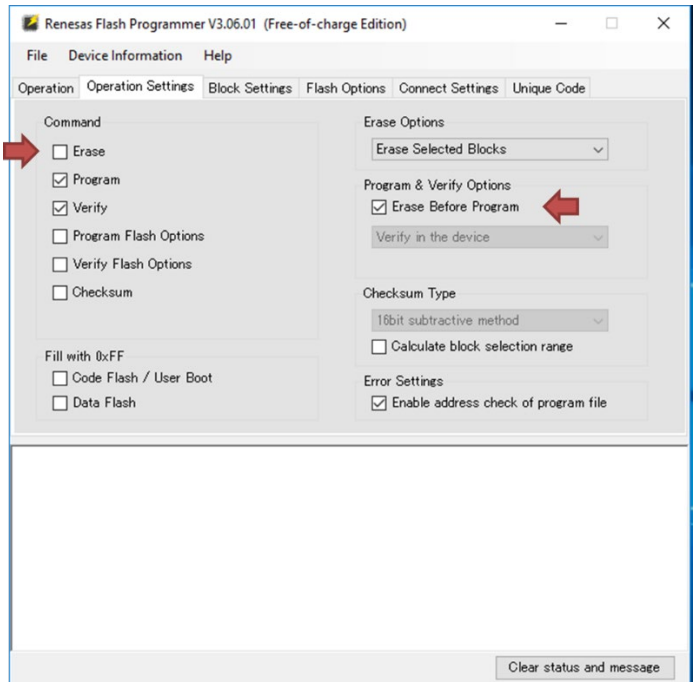


Figure 4-1 The configuration of the object files to be download in case of RFP

(2) When e2studio/CS+ is used for the flash programing and debug

If not only the flash programing but also the debugging is necessary, e2studio or CS+ needs to be used. Figure 4-2 and Figure 4-3 show the example configurations of the object files to be download in case of e2studio and CS+ respectively.

When e2studio is used, the object file with the image and symbols (.x) of the FUOTA sample application needs to be download first, and the object file with only image (.mot) of the F/W update sample application needs to be download secondly.

When CS+ is used, the object file with the image and symbols (.abs) of the FUOTA sample application needs to be download first, and the object file with only image (.mot) of the F/W update sample application needs to be download secondly.

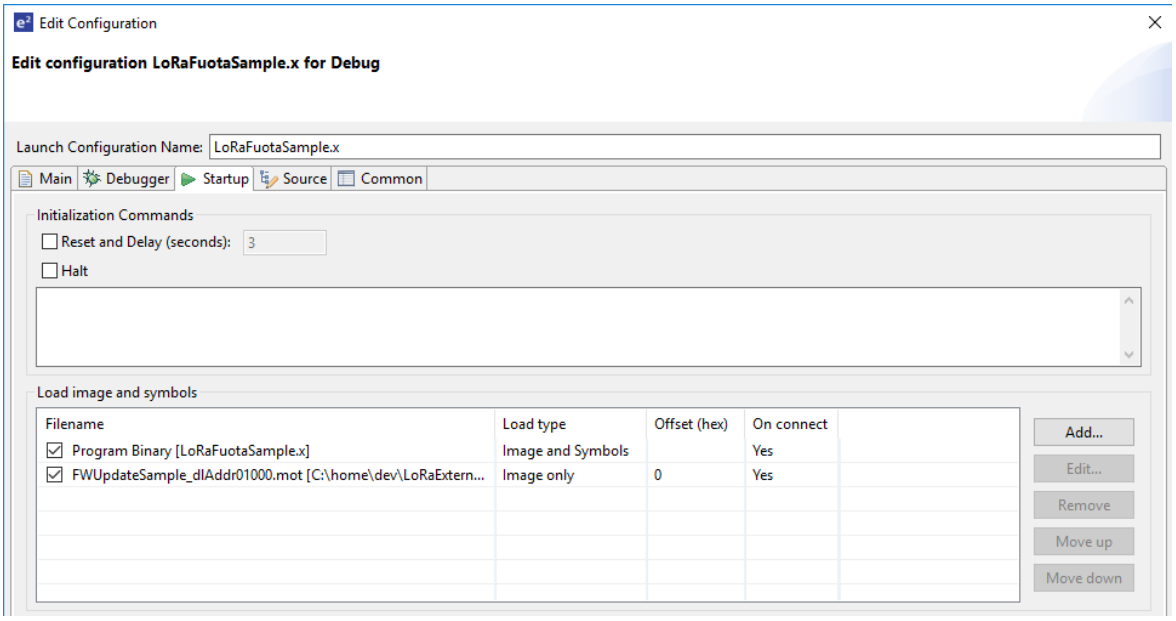


Figure 4-2 The configuration of the object files to be download in case of e2studio

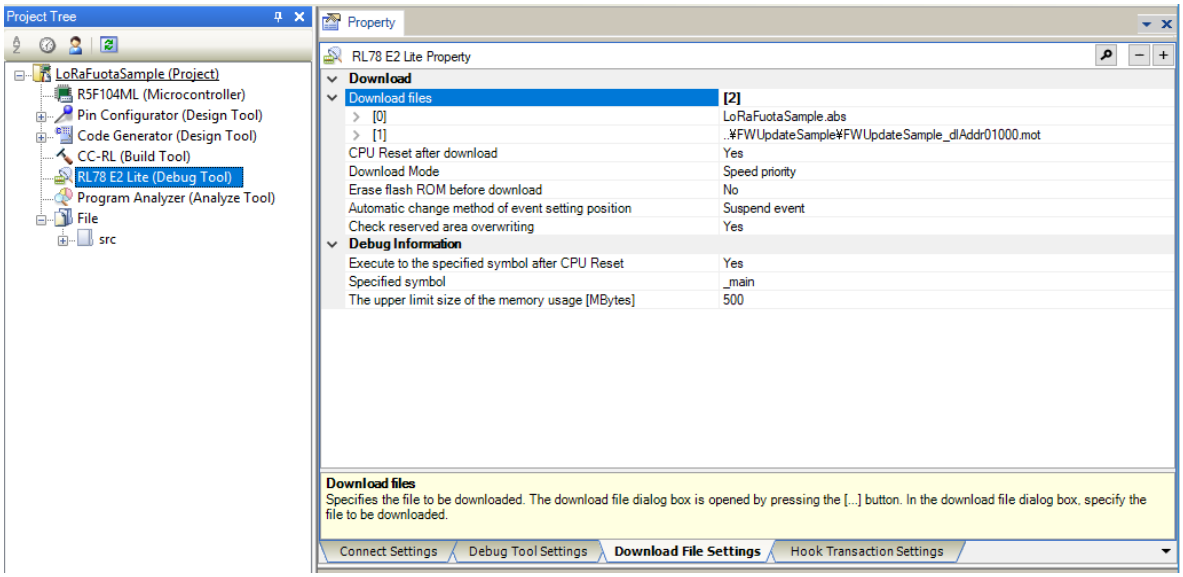


Figure 4-3 The configuration of the object files to be download in case of CS+

4.2 Preparation for LoRaWAN network server

4.2.1 Make F/W image files (binary)

The F/W image files (binary) need to be converted from the object file (.mot) of the new firmware to deliver the files from the LoRaWAN network server. A batch file, ‘make_fwimage.bat’, is prepared for the conversion.

Name	make_fwimage.bat	
Description	Make the F/W image from an object file (.mot) and output F/W image file(s) from the F/W image, divided by the specified size	
Syntax	make_fwimage.bat [MotFile] [FWVersion] [DividingSize] [OutputFile]	
Folder	samples\tools\FUOTA	
Argument	MotFile	Object file (.mot)
	FWVersion	Version to be set to F/W image Four bytes HEX number without prefix.
	DividingSize	Dividing size of the F/W image. Need to specify the dividing size less than to the data block size that the end device can receive, which can bet set to FUOTA_CONFIG_FRGMNT_MAX_DATABLK_SIZE. (See section 2.2.2) If 0 is specified, the F/W image will not be divided.
	OutputFile	The base file name for the F/W image. See below.
Example	**Argument:** [MotFile] = Application.mot [FWVersion] = 0x00000100 (4 byte) [DividingSize] = 8192 byte [OutputFile] = FWImage make_fwimage.bat Application.mot 00000100 8192 FWImage **Output file:** - FWImage_desc0x00000000.bin (Image file to be sent first, 8192 byte) - FWImage_desc0x00000001.bin (Image file to be sent second, 8192 byte) - FWImage_desc0x00000002.bin (Image file to be sent third, 1024 byte) Where the total size of the FW image is 17408 bytes	

4.2.2 Setup to deliver F/W image

The F/W image file(s) made in the section 4.2.1 need to be specified to the LoRaWAN network server so that the files will be delivered from the server via the fragmentation session(s). The descriptor corresponding to the F/W image file(s) also need to be specified to indicate the index of the F/W image file(s), which starts from 0 and incremented for each fragment session.

F/W image files and corresponding descriptors in case of the example shown in section 4.2.1

Data block to send	Descriptor
FWImage_desc0x00000000.bin	0x00000000
FWImage_desc0x00000001.bin	0x00000001
FWImage_desc0x00000002.bin	0x00000002

In addition, the setup such as the following items might be additionally necessary. For more details, refer to the specification of the LoRaWAN server.

Example of the setup:

- Fragmentation

  Transmission interval of the fragments to be sent

- Multicast (if multicast is used for the delivery)

Device class used for the multicast session

GenAppKey to share the multicast session key

- Time configuration

Time to start/end of the delivery

4.3 Example operations of end device

This section describes the example operation of end device. In this example, it is supposed that the sample F/W image file included in the software package is used. The file is in the following folders. The size of the file is 566 bytes and the size of the code included in the file is 512 bytes.

Please note that the sample F/W image can be used if the source code and the build setting of the FUOTA sample application are not changed from the original ones included in the software package because the sample F/W image updates the code in the specific address area.

[Sample F/W image file]

(1) e2studio

samples\project\e2studio\rl78g14fpb_sx126x\LoRaFuotaSample\sample_fwimage
sample_fwimage_desc0x00000000.bin

(2) CS+

samples\project\csplus\rl78g14fpb_sx126x\LoRaFuotaSample\sample_fwimage
sample_fwimage_desc0x00000000.bin

When the sample F/W image file is applied via the FUOTA process, the version of the FUOAT sample application is changed from Ver.03.01 to Ver.09.00. The change can be confirmed using the AT command, “AT+VER?”.

Before the sample F/W image is applied	AT+VER? +VER: LoRa Sample App Ver.03.01 OK
After the sample F/W image is applied	AT+VER? +VER: LoRa Sample App Ver.09.00 OK

The following is the sample operation of the end device for the FUOTA. The value with under line should be change according to the setting of the LoRaWAN server and the information of the end device.

Example operation and notification for end device	Description
	Confirm the version of the current FUOTA sample application is 'Ver.03.01'.
AT+VER? +VER: LoRa Sample App Ver.03.01 OK	Show version
	Set parameters such as region, device class, activation mode, AppKey, AppEUI, DevEUI required in case of OTAA. Refer to [2] for details.
AT+REGION=6 OK	Set AS923 for region
AT+CLASS=0 OK	Set Class A for device class
AT+ACTMODE=1 OK	Set OTAA for activation mode
AT+DEVEUI=90F OK	Set 000000000000090F for DevEUI
AT+APPEUI=10E OK	Set 000000000000010E for AppEUI
AT+APPKEY=F0E OK	Set 00000000000000000000000000000F0E for AppKey
AT+JOIN OK +JOIN: JOIN_ACCEPTED	Request to join the network
	Start FUOTA and set related parameters
AT+FUOTASTART OK	Start FUOTA
AT+FUOTASET=10,0 OK	Disable to send AppTimeReq periodically
AT+FUOTASET=F0,30 OK	Set 30 seconds to sending interval of uplink messages to receive downlink message

AT+FUOTASET=20,00000000000000000000000000000000 OK	Set 00000000000000000000000000000000 to GenAppKey to receive the multicast session key from the server. Need to change the key value according to the server setting/specification.
	Indication of multicast when the multicast is used.
+FUOTAIND: 0,2,0,54,4096	Indication of setup of multicast session, which is indicated when the multicast is used Session class: Class C Multicast group ID: 0 Seconds to start session: 54 seconds Seconds to timeout session: 4096 seconds
+FUOTAIND: 1,2,0,4096	Indication of start of multicast session Session class: Class C Multicast group ID: 0 Seconds to timeout session: 4096 seconds
	Wait until complete F/W image is received and update the FUOTA sample application
+FUOTAIND: 128	Indication of the F/W image is ready If the F/W image consists of multiple data blocks, this command will be indicated when all data blocks are received.
AT+FUOTAUPDT OK	Update the firmware, the FUOAT sample application, using the F/W image after its validation. After the update, the updated FUOTA sample application will be activated.
	Confirm the version of the updated FUOTA sample application is changed to ‘Ver.09.00’. It indicates the firmware update is successful.
AT+VER? +VER: LoRa Sample App Ver.09.00 OK	Show version.

Revision History

Rev.	Date	Description	
		Section	Summary
01.00	Oct. 9, 2020	-	Initial release
03.00	Mar. 26, 2021	4.3	Changed the versions from V03.00 to V09.00, and from V02.10 to V03.00 in the example operations
03.01	June 10, 2021	4.3	Changed the version form V3.00 to V3.01 No functional changes.

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Semtech, the Semtech logo, LoRa, LoRaWAN and LoRa Alliance are registered trademarks or service marks, or trademarks or service marks, of Semtech Corporation and/or its affiliates.

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.