Software Release Note

Z-Wave 700 SDK 7.11.0.GA

Document No.:	SRN13181
Version:	2
Description:	-
Written By:	JFR;AYURTTAS
Date:	2019-03-13
Reviewed By:	JFR;SCBROWNI;JOPEDERSEN;HAKRONER;JRM;NTJ;AYURTTAS;BBR;AMUNKHAUS;PSH;JBU
Restrictions:	Partners Only

Approved by:

Date	CET	Initials	Name	Justification
2019-03-13	05:12:22	NTJ	Niels Johansen	

REVISION RECORD				
Doc. Rev	**Date**	**By**	**Pages Affected**	**Brief Description of Changes**
1	20181107	JFR	All	Initial draft based on SRN13926-8 – Z-Wave 500 Series SDK v6.81.03
2	20190118	MLEDESMA	ALL	Grammar and structure (consistent format) modification

Table of Contents

List of Figures

1 INTRODUCTION

This chapter introduces the SDK as well the Z-Wave technology.

1.1 Introduction to the SDK

The Z-Wave 700 Software Development Kit (SDK) on the Silicon Labs Zen Gecko platform, enables developers to design low power, long range Z-Wave products. The Z-Wave 700 SDK building blocks and tools ensures fast time to market while staying compliant with the Z-Wave Plus V2 certification.

The software consists of Z-Wave libraries supporting controller and slave devices and a Z-Wave Plus V2 Application Framework, as well as code for a broad range of smart home applications. To realize a specific application, it is recommended to modify one of the existing pre-certified apps having the wanted Role Type (AOS, RSS & LSS) [13].

The Z-Wave 700 SDK enables support of the Z-Wave 700 Module and Z-Wave 700 SoC. It is a mature release (GA) having similar functionality as version 6.81.0x. This release is intended for Z-Wave certified 700 based products entering volume production. All the Z-Wave Plus V2 Applications are Z-Wave certified. Chapter 2 list all programs and version numbers included on this SDK. For details regarding functionality, refer to Chapter 3 and 4. Finally, refer to [9] for a detailed description of contents.

1.1 Abbreviations

Abbreviation	Explanation
ACK	Acknowledge
AOS	Always On Slave Role Type. Intended for mains powered devices that are always reachable from a RF point of view.
API	Application Programming Interface
C	Command
CC	Command Class
DUT	Device Under Test
ID	Identifier
FLiRS	Frequently Listening Routing Slave having the LSS Role Type. Communication to a FLiRS node can be established by a wakeup beam
LSS	Listening Sleeping Slave Role Type. Intended for battery powered devices that can be reached even though they are sleeping thanks to a wakeup beam (FLiRS device).
NIF	Node Information Frame
NWI	Network Wide Inclusion (add node out of direct range to Z-Wave network)
NWE	Network Wide Exclusion (remove node out of direct range from Z-Wave network)
OTA	Over The Air (e.g. making a firmware update wireless)
OTW	Over The Wire (e.g. making a firmware update via the serial API interface)
RSS	Reporting Sleeping Slave Role Type. Intended for battery powered devices that only wake up and communicates when a local event has occurred.
S0	Security 0 Command Class
S2	Security 2 Command Class
SDK	Software Development Kit
SSA	Server-Side Authentication

1.2 Introduction to the Z-Wave Technology

Z-Wave is a wireless mesh protocol oriented to the residential control and automation market but may also be used in light commercial environments. The technology provides a simple yet reliable method to wirelessly control lights and A/V equipment in your house. Z-Wave works in the unlicensed industrial, scientific, and medical (ISM) bands around 900MHz. Regional frequencies vary slightly. Each Z-Wave network may comprise up to 232 nodes. Nodes may retransmit a message to guarantee delivery. The typical communication range between two nodes is 100 feet.

The Z-Wave ecosystem offers a routing protocol stack and a complete Z-Wave Plus Application Framework of device types and command classes for interoperable deployments. Interoperability is ensured between all device types thanks to the Z-Wave certification program. The Z-Wave logo is only granted to products passing certification.

1.2.1 Protocol Stack Overview

Z-Wave offers a routing protocol that reliably transfers messages up to 5 hops away; i.e., up to 500 feet. The protocol stack comprises a PHY/MAC layer to control access to RF media, a transport layer to handle frame integrity and retransmissions, and a network layer with all its routing magic and application interfaces.

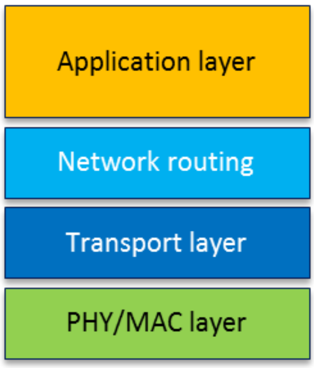

Figure 1. Z-Wave Protocol Stack

The maximum size of payload data is 46 bytes when routing is used. The Z-Wave protocol uses standard collision-avoidance methods; postponing a transmission a random number of milliseconds when media is busy. The Z-Wave transport layer controls the transfer of data between two nodes including acknowledgement and optional retransmission.

Multicast and broadcast may only be used in direct range. Broadcast and multicast may be used to reach more than one destination address. In case of multicast, the same payload will be delivered to selected nodes only.

The Z-Wave application layer is responsible for handling application commands. Commands are divided into two classes: Z Wave protocol and application-specific. Most protocol-related operations are just address assignment logic, but commands that are more complex are defined for advanced network management operations.

Each Z-Wave network has a unique 32-bit identifier called Home ID. Every new node joining the network inherits the same Home ID from the primary controller. Individual nodes in the network are addressed using an 8-bit Node ID that is unique within the network.

1.2.2 Classic Z-Wave

The following text makes references to classic nodes. In short, the term “Classic Z-Wave node” covers previous generations of Z-Wave nodes which do not implement recently introduced features such as Network-Wide Inclusion (NWI), Dynamic Route Resolution and FLiRS communication.

1.2.3 Node Types

There are two main types of devices: controllers and slaves. Controllers can handle network management and communication to classic nodes. Slaves provide no network management capability.

1.2.3.1 Controllers

A controller node maintains a routing table for all operational links in the network. This table allows the controller to calculate routes between any two nodes in the network. The primary controller may refresh the routing table and distribute updated routing tables to other controllers.

Controllers come in three variants, portable, static and bridge. However, SDK 7.11.0 contains only the bridge controller because static controller is discontinued.

The portable controller is optimized for battery operation. It is typically used for remote control devices.

The bridge controller is intended for mains-powered control panels, gateways, or network managers. The bridge controller may also act as repeater for other nodes.

1.2.3.2 Slaves

A slave device has simpler functionality than a controller has. It may repeat a message for other nodes.

Reporting Sleeping Slave (RSS) Role Type node is intended for battery powered devices that only wake up and communicates when a local event has occurred. RSS device may be used for sensor-style devices such as alarms and sensors.

Listening Sleeping Slave (LSS) Role Type node is intended for battery powered devices that can be reached even though they are sleeping thanks to a wakeup beam (FLiRS device). Referred to as duty cycling in the literature, the Frequently Listening Routing Slave (FLiRS) wakes up in fixed intervals to listen very shortly for a preamble pattern. This enables the design of products with battery lifetimes measured in years. Yet, it is possible to reach such devices on short notice.

RSS and LSS nodes cannot operate as repeaters as they are sleeping most of the time to conserve battery.

1.2.4 Network Operation

Management of Z-Wave nodes constitutes two main operations: inclusion/exclusion and association. Inclusion adds a new node to the network. Exclusion removes a node. Only primary controllers can include and exclude nodes.

Association is the creation of a logical connection between applications. In other words, it defines what controls what. Association is handled by the application layer.

1.2.5 Routing Principles

Z-Wave uses source routing to reach a destination. Source routing allows implementation of a lightweight protocol; avoiding distributed routing tables in all repeaters. This puts a limit to the length of routes. Real-world deployments indicate that residential networks rarely have more than 2-hop routes. Z-Wave's support for 5-hop routes is a sufficient and efficient compromise.

The route is carried in the routing header and every repeater forwards the frame according to the routing header. Only always-listening nodes can participate in routing, but routing may also be used to reach FLiRS nodes.

Network Wide Inclusion (NWI) allows a user to include a new node even though the new node is not within range of the primary controller. Dynamic route resolution allows a node to repair broken routes during normal operation. Classic nodes do not support NWI and dynamic route resolution.

Network Wide Exclusion (NWE) uses the same explorer strategy as Network Wide Inclusion (NWI) to accomplish an out-of-range exclusion of nodes from the network. It is also possible to remove a specific node from the network by specifying the Node ID.

1.2.6 Application Development

Depending on node type functionality (such as controller vs. slave), developers may choose from a selection of libraries. On top of the chosen library, an application designer may choose from a wide range of Command Classes; including light control, sensors, garage port control, and many others. Command Classes are a collection of functionally related commands. A device may implement several functions and therefore support more Command Classes.

Z-Wave applications are designed as a state machine periodically polled from the Z-Wave library. This allows for the design of products with fewer CPU resources than typically required for OS'es with threads, tasks, priorities, etc. This again translates into inexpensive products suited for mass production.

Depending on the actual product, an application may interface to the Z-Wave protocol stack in three ways:

- Most constrained devices, like a light dimmer with one button, may have its application running in the on-chip MCU. In this configuration, the Z-Wave API is used directly via function calls provided by the binary image implementing the Z-Wave library.
- Larger devices, like a remote control with display, may have its own host processor. The application designer may prefer to implement all application logic in the host processor; only

running the Z-Wave protocol stack in the on-chip MCU. The Z-Wave Serial API provides an abstracted version of the Z-Wave API that is accessed via an on-chip serial port. The application design principle for the Z-Wave part should still be a state machine that reacts to incoming events, callback functions, and timeouts.

- Most advanced devices like IP gateways and PC-based light control servers may use an even more abstracted API provided via the Network Management Command Class. In this model, all communication is carried in IP packets. The Z/IP Gateway library provides this mapping.

1.2.7 Managing Interoperability

Interoperability is a key part of the Z-Wave ecosystem. Every product must pass certification to be granted the Z-Wave logo. The Z-Wave Alliance manages the Z-Wave certification program but certification testing is performed by independent test houses. Certification makes sure that a product correctly implements all device and command classes that it claims supporting.

2 RELEASED VERSIONS

2.1 SDK 7.11.0

```
Z-Wave Framework and Certified Applications.............. v10.11.0
Z-Wave Protocol and Serial API Applications.............. v7.11.0
Z-Wave Serial API Application Interface.................. v8
```

3 FEATURE OVERVIEW

The Z-Wave 700 SDK version 7.11.x contains the following major enhancements compared to SDK version 6.8x:

- New integrated development environment Simplicity Studio [10]:
  - Z-Wave support of ZGM130S SIP for end devices and EFR32ZG14 SoC for gateways.
  - New button/LED board BRD8029 with separate definition file.
  - Debugger and trace (ETM) via Serial Wire Debug (SWD).
  - Energy Profiler enabling optimization of power consumption including source code trace.
  - Integrated programmer Simplicity Commander.
  - Z-Wave sniffer functionality in Simplicity Studio known as Network Analyzer is not supported. Use instead a standalone 500 Series based Z-Wave sniffer to log RF traffic. Both Z-Wave sniffer and Z-Wave PC Controller is distributed in Simplicity Studio.
  - Distribute Z/IP Gateway and Z-Ware source code and binaries.
  - No license fees.
- Event driven architecture based on FreeRTOS optimizing power consumption.
- Larger memory for applications including Z-Wave Plus V2 Framework reducing the need for host processor-based applications using the serial API interface. Application code space is restricted to 8KB RAM data and 64KB FLASH code including framework.
- Simplified power management handling.
- Easy migration from 500 Series when using the Z-Wave Framework.
- Reliable File System handling application data without an external non-volatile memory.
- Bootloader supporting signature, encryption, and compression.
- Flexible peripheral driver (EMDRV and EMLIB) installation and customization.
- Supporting Z-Wave Plus v2 Framework and applications certified accordingly.
- Serial API-based applications only distributed as binaries due to Z-Wave Plus v2 Framework compliance.
- Serial API Static Controller and Bridge Controller merged to one Serial API Controller. The Serial API Controller available on both EFR32ZG14 SoC for gateways and ZGM130S SIP for end devices.
- Serial API enhanced 232 slave without an external non-volatile memory obsoleting Serial API routing slave. The Serial API enhanced 232 slave available only on ZGM130S SIP for end devices.

- Discontinued MyProductPlus because only enhanced 232 slave is available for application development in end devices. To realize a specific application, it is recommended to modify one of the existing pre-certified apps having the wanted Role Type or similar (AOS, RSS, and LSS).
- OTA/OTW firmware update without an external non-volatile memory. Support dual firmware on end devices for robust and safe firmware update.
- On chip private/public keys and QR Code generation to increase security and streamline production.
- API call ZW_RFPowerLevelSet and Low Power option obsoleted in Z-Wave protocol preventing change of RF power level on the fly. RF power level is now predefined in the header file config_rf.h.
- RAILTest is a standalone test application, which is used for testing radio functionality, as well as peripherals, deep sleep states, etc. HW Configurator is not supported in RAILTest. RAILTest replaces uRFLinkX and Production Test Gen/DUT tools.

4 NEW FEATURES DESCRIPTION

4.1 SmartStart (6.81.0x+)

Z-Wave SmartStart aims to shift the tasks related to inclusion of an end device into a Z-Wave network away from the end device itself, and towards the more user-friendly interface of the gateway.

Z-Wave SmartStart removes the need for initiating the end device to start inclusion. Inclusion is initiated automatically on power-ON and repeated at dynamic intervals for as long as the device is not included into a Z-Wave network. As the new device announces itself on power-ON, the protocol will provide notifications and the gateway can initiate the inclusion process in the background, without the need for user interaction or any interruption of normal operation. This improvement also removes the possibility of other devices being included, as the SmartStart inclusion process only includes authenticated devices.

By moving the device authentication process into the manufacturing and distribution phase or service provider domain, the end user is no longer required to do anything but to power on the devices. This enables a simplified user experience where the device is genuinely ready to use, right out of the box . The device manufacturer or service provider can now prepare inclusion prior to the devices ending up at the end user's house.

Building on the elements introduced by S2 security, the Z-Wave SmartStart is not only easy for the end user, but also secure. Z-Wave SmartStart uses the same device specific keys (DSK) that form the foundation of the secure inclusion process of S2. Only authorized and intended devices are included in the Z-Wave network.

SmartStart introduces a number of new API's for using learn mode and adding nodes to the network. For end nodes, the ZW_NetworkLearnModeStart() is now used for controlling learn mode. For controller nodes, ZW_AddNodeToNetwork() is used.

4.2 SmartStart and S2 QR Code Generation

Z-Wave devices supporting the Security 2 (S2) Command Class or SmartStart provisioning must provide a QR code physically on the device as well as on packaging. The actual marking and layout requirements are documented in [14] while the data string encoded in the QR code is specified in [22].

Both QR code and S2 DSK are generated in the SmartStart device itself and Simplicity Commander facilitate readout of QR code for printing.

The current SDK release contains two software utilities [23] to assist developers in creating and to verify the contents of a QR code:

- QrCodeEncoder.xlsm
    - Encoding of QR code fields
    - Single-sample generation of QR codes for prototyping
    - Generation of dynamic string for Production Control File with fields to be replaced during production
- QrCodeDecoder.xlsm
    - Decodes the string contained in a QR code using an arbitrary smart phone QR code scanner application

All gray fields should be left untouched.

The utilities are implemented using Excel sheets, incorporating several macro functions for SHA-1 checksum calculation, QR code rendering, and control file generation. The utilities must therefore be stored in a folder that is not write protected.

4.3 Improved Z-Wave Plus V2 Framework

The Z-Wave Plus V2 Framework is an extension of the well-known Z-Wave Plus certified solutions featuring a selected set of extended features and capabilities that enhance the end user experience and make Z-Wave installations even faster and easier to install and set up.

The Z-Wave Plus V2 requirements are as follows:

1. SmartStart is mandatory.
2. OTA Firmware update is mandatory.
3. Extended CC support for root devices and Multi-Channel End Points. All actuator Device Types must support Basic CC.
4. Indicator to identify device such as a visible LED etc.
5. Dynamic capabilities and node discovery. Capabilities may change due to user interaction.
6. New controller requirements to strengthen interoperability; for instance, blocking or forced exclusion of non-preferred devices is not allowed anymore.
7. Minimum CC to be controlled by a controller extended. This applies also for bridging devices interfacing to another technology.
8. Detection Z-Wave Plus V2 compliant nodes using Z-Wave Plus Info CC.

For a detailed description of application development using the Z-Wave Plus V2 Framework, refer to [17].

5 Z-WAVE PROTOCOL

The Z-Wave Protocol (Z-Wave API library) is a low bandwidth half-duplex protocol designed for reliable and robust wireless communication in a low-cost control mesh network. This version supports the 700 single chips in various configurations. For an overview, refer to [1]. The API consists of a Bridge Controller library and an Enhanced 232 Slave library. Only Enhanced 232 Slave library supports S0 and S2. The type of library used, and associated Role Type depends on the application features needed.

5.1 Serial API

The Serial Applications Programming Interface (Serial API) allows a host to communicate with a Z-Wave chip. The host may be a PC or a less powerful embedded host CPU, e.g., in a remote control or in a gateway device. This solution is typically used when the whole application cannot reside on the Z-Wave chip itself. The Serial API code contains an example of how a serial UART interface to the Z-Wave protocol can be implemented. The Serial API supports both controller and slave applications. For detailed information, refer to [9].

6 Z-WAVE FRAMEWORK AND EMBEDDED APPLICATIONS

The SDK contains code as well as compiled code for Z-Wave embedded applications according to devices in [13]-[14], and command classes in [4]-[7]. The Z-Wave Plus V2 embedded applications supports both non-secure and secure S0/S2 in one target.

Associations must be configured to examine all the features in the Z-Wave embedded applications. Setting up associations is fully supported by the Z-Wave PC based Controller v5 and not older versions of the Z-Wave PC based Controller.

The code can be used as is to become familiar with Z-Wave or changed according to the needs of the application programmer.

A Z-Wave application based on earlier Z-Wave Single Chips requires porting of the source code to the 700 Single Chip. For details about porting, refer to [24], [11], and [12].

A Z-Wave application based on earlier SDKs may also require porting to a newer version of the Z-Wave Plus V2 Application Framework. For details, refer to [17].

6.1 Door Lock with Key Pad

The Door Lock with Key Pad application shows a lock implementation that has a built-in keypad. It will support user codes to open a door and thereby eliminate the need for traditional keys. Typically, it is possible to both lock and unlock the door remotely through the Z-Wave protocol. The Door Lock with Key Pad implementation is built upon the Z-Wave Plus Application Framework [17]. The Door Lock with Key Pad is based on Door Lock Keypad Device Type with Listening Sleeping Slave (LSS) as the Role Type and enhanced 232 slave. For detailed information, refer to [9].

6.2 On/Off Switch

The On/Off Power Switch application shows a switch implementation to turn on any device that is connected to power. Examples include lights, appliances, etc. The On/Off Switch implementation built upon the Z-Wave Plus V2 Framework [17]. The On/off Switch is based on On/Off Power Switch Device Type with Always On Slave (AOS) as the Role Type and enhanced 232 slave. For detailed information, refer to [9].

6.3 PIR Sensor

The PIR Sensor application shows a presence/movement detector implementation for controlling other devices and for sending notifications. The PIR Sensor implementation is built upon the Z-Wave Plus V2 Framework [17]. The PIR Sensor is based on Sensor – Notification Device Type with Reporting Sleeping Slave (RSS) as the Role Type and enhanced 232 slave. For detailed information, refer to [9].

6.4 Power Strip

The Power Strip application shows an extension block implementation to turn on several devices that are connected to power. Examples include lights, appliances, etc. The Power Strip implementation is built upon the Z-Wave Plus V2 Framework [17]. The Power Strip is based on Power Strip Device Type with Always On Slave (AOS) as the Role Type and enhanced 232 slave. The Power Strip show also how to implement a Multi-Channel device. For detailed information, refer to [9].

6.5 Wall Controller

The Wall Controller application shows a push button switch panel implementation to control devices in the Z-Wave network from push buttons (physical or virtual) on a device that is meant to be mounted on a wall. Examples include scene and zone controller and wall mounted AV controllers. Device of this type can both be battery operated or main powered. The Wall Controller implementation is built upon the Z-Wave Plus V2 Framework [17]. The Power Strip is based on Wall Controller Device Type with Always On Slave (AOS) as the Role Type and enhanced 232 slave. The Wall Controller shows how to implement an interrupt service routine (ISR) on application level. For detailed information, refer to [9].

6.6 Z-Wave Plus V2 Application Framework

The Z-Wave Plus V2 Application Framework simplifies implementation of robust Z-Wave Plus compliant and interoperable products. Many Z-Wave certification requirements are also handled by the Z-Wave Plus Application Framework making it a lot easier to pass certification. The Z-Wave logo is only granted to products passing certification.

6.6.1 Application Command Handlers

The Application Command Handlers code contains an implementation of various command classes used by Z-Wave Plus applications. For detailed information, refer to [17].

6.6.2 Application Utilities

The Application Utilities code contains an implementation of various general-purpose functions used by Z-Wave Plus applications. A large part of the S0/S2 security solution resides now in the Z-Wave Protocol. For detailed information, refer to [17].

REFERENCES

[1] Silicon Labs, INS10243, Instruction, Z-Wave Protocol Overview.
[2] Silicon Labs, SDS10242, Software Design Specification, Z-Wave Device Class Specification.
[3] Silicon Labs, SDS10865, Software Design Specification, Z-Wave Security Application Layer.
[4] Silicon Labs, SDS13781, Software Design Specification, Z-Wave Application Command Class Specification.
[5] Silicon Labs, SDS13782, Software Design Specification, Z-Wave Management Command Class Specification.
[6] Silicon Labs, SDS13783, Software Design Specification, Z-Wave Transport-Encapsulation Command Class Specification.
[7] Silicon Labs, SDS13784, Software Design Specification, Z-Wave Network-Protocol Command Class Specification.
[8] Silicon Labs, SDS13548, Software Design Specification, List of defined Z-Wave Command Classes.
[9] Silicon Labs, INS14453, Instruction, How to use Certified Apps in Development Kit 7.11.x
[10] Silicon Labs, AN0822, Simplicity Studio™ User Guide.
[11] Silicon Labs, APL12444, Application Note, Porting Z-Wave Appl. SW from ZW0301 to 500 Series.
[12] Silicon Labs, APL12445, Application Note, Porting Z-Wave Appl. SW from 400 to 500 Series.
[13] Silicon Labs, SDS11846, Software Design Specification, Z-Wave Plus Role Types Specification.
[14] Silicon Labs, SDS11847, Software Design Specification, Z-Wave Plus Device Types Specification.
[15] Silicon Labs, SDS14306, Software Design Specification, Z-Wave 700 Lock Bits and User Data Page Contents.
[16] Silicon Labs, INS12350, Instruction, Serial API Host Appl. Prg. Guide.
[17] Silicon Labs, INS14259, Instruction, Z-Wave Plus V2 Application Framework v7.11.X.
[18] Silicon Labs, INS14265, Instruction, 700 Integration Guide.
[19] Silicon Labs, INS13474, Instruction, Z-Wave Security Whitepaper.
[20] Silicon Labs, SDS13349, Software Design Specification, Security considerations in Home Control installations.
[21] Silicon Labs, APL13434, Application Note, FAQ: On the use of S0, S2 and Supervision CC in product design and deployments.
[22] Silicon Labs, SDS13937, Software Design Specification, Node Provisioning QR Code Format.
[23] Silicon Labs, INS13975, Instruction, Smart Start Production control.
[24] Silicon Labs, APL14440, Application Note, Porting Z-Wave Appl. SW from 500 Series to 700.

INDEX