Software Release Note

Z-Wave 500 Series SDK v6.71.03

Document No.:	SRN13389
Version:	9
Description:	-
Written By:	JFR;COLSEN;PSH;BBR
Date:	2018-03-02
Reviewed By:	BBR;NTJ;JFR;TMORTENSEN;COLSEN;LTHOMSEN
Restrictions:	Public

Approved by:

Date	CET	Initials	Name	Justification
2018-03-02	15:47:54	JFR	Jørgen Franck	on behalf of NTJ

REVISION RECORD

Doc. Rev	Date	By	Pages affected	Brief description of changes
1	20160120	JFR	All	Initial draft based on SRN12911-5 – Z-Wave 500 Series SDK v6.60.00 Beta
2	20160701	JFR	Section 1, 3 & 4 2 and 1.1	Described new features in SDK 6.70.01 Updated released versions to SDK 6.70.01 and use of Keil PK51 v9.54A
3	20160901	JFR	Section 1.1	Must use Keil PK51 v9.54A
4	20160909	JFR	Section 1, 2 & 4 Section 6 -	Updated to SDK 6.71.00 Removed On/Off Switch, PIR Sensor, Power Strip and Wall Controller because these Z-Wave Plus Apps are not certified yet. Removed Z-Wave PC based Controller application v5 because it is now available on ZTS as an individual program.
5	20170124	JFR	Section 6	Added On/Off Switch, PIR Sensor, Power Strip and Wall Controller. Notice that applications are not certified yet.
6	20170211	JFR	Section 1, 2 & 4	Updated to SDK 6.71.01
7	20170712	PSH	Section 1, 2 & 4	Updated to SDK 6.71.02
7	20170719	JFR	Section 6.1 Section 4.5	Updated command class requirement numbers in table MY frequency obsoleted
8	20171018	JFR	Section 1, 2 & 4	Updated to SDK 6.71.03
9	20180302	BBR	All	Added Silicon Labs template

Table of Contents

List of Figures

1 INTRODUCTION

This chapter provides an introduction to the SDK as well the Z-Wave technology.

1.1 Introduction to the SDK

The Z-Wave 500 Series Software Development Kit (SDK) intended to help developers creating Z-Wave Plus compliant products in a fast and cost effective manner. The software consists of Z-Wave libraries supporting controller/slave devices and a Z-Wave Plus Application Framework, as well as code for a broad range of home automation applications. To realize a specific application it is maybe easier to modify an existing sample app instead of using MyProduct app as a starting point.

The Z-Wave 500 Series SDK version 6.71.03 is a mature version of 6.70.xx intended for 500 Series based products entering volume production. The Z-Wave Plus Applications Door Lock with Key Pad, On/Off Switch, PIR Sensor, Power Strip and Wall Controller are certified. For details regarding functionality, refer to Chapter 3 and 4. Finally, refer to [10] for a detailed description of contents.

Notice: Be aware that Z-Wave SDK 6.71.xx requires Keil PK51 v9.54A.

1.2 Abbreviations

Abbreviation	Explanation
ACK	Acknowledge
API	Application Programming Interface
C	Command
CC	Command Class
CSA	Client-Side Authentication
DUT	Device Under Test
ID	Identifier
FLiRS	Frequently Listening Routing Slave. Communication to a FLiRS node can be established by a wakeup beam
NIF	Node Information Frame
NWI	Network Wide Inclusion (add node out of direct range)
NWE	Network Wide Exclusion (remove node out of direct range)
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)
S0	Security 0 Command Class
S2	Security 2 Command Class
SDK	Software Development Kit
SSA	Server-Side Authentication
TO	Test Observation (bug)

1.3 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 in order to guarantee delivery. The typical communication range between two nodes is 100 feet.

The Z-Wave eco system 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.

1.3.1 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.

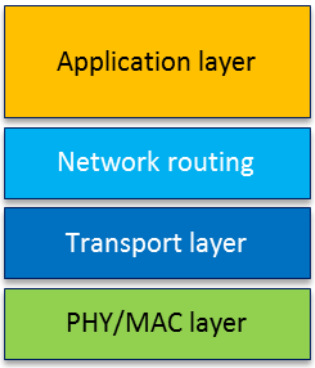


Figure 1. Z-Wave protocol stack

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.

1.3.2 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.

1.3.3 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.

1.3.3.1 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.

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

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

1.3.3.2 Slaves

A slave device has simpler functionality than a controller has and can be implemented without any external non-volatile storage. It may repeat a message for other nodes.

A special variant, the WakeUp slave may be used for sensor-style devices such as alarms and sensors.

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.

WakeUp slaves and FLiRS nodes cannot operate as repeaters as they are sleeping most of the time to conserve battery.

1.3.4 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.

1.3.5 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.

1.3.6 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 8051 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 8051 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.

1.3.7 Managing interoperability

Interoperability is a key part of the Z-Wave eco system. 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 6.71.03

```
Z-Wave Plus Framework and Embedded Applications.................. v3_01_01
Z-Wave Serial API Interface...................................... v7
Z-Wave Protocol.................................................. v5_03_00

Tools
=====
IMA Tool......................................................... v0_99
uVision Project Generator........................................ v1_15
```

3 OVERVIEW

The Z-Wave 500 Series SDK version 6.7x contains the following major enhancements compared to SDK version 6.6x:

- New improved second-generation security solution (S2 based on Security 2 Command Class) for a routing slave and an enhanced 232 slave with respect to authentication levels, initial key exchange, communication overhead, multicast support etc. However, the routing slave has some limitations with respect to the new security solution because it is without an external NVM.

- Introduction of a new Inclusion Controller Command Class allowing inclusion controller to inform the SIS that a new node has been added, and that the SIS will have to perform any additional required setup operation. Examples of such setup operations could be Z-Wave Plus Lifeline configuration or Security 2 bootstrapping.

- Introduction of a new client side authentication enabling OTA firmware update from a S0 device to a S2 device. This method makes it possible to enter the QR code of a controller onto joining device (client side authentication). Specifically targeted door locks having a keypad.

- An improved Z-Wave Plus Application Framework resulting in a simpler application development due to flexible NIF configuration, extended task and event handler, simpler command class handling, integrated multicast handling, integrated multichannel handling etc.

- The Sensor PIR application is now based on a routing slave instead of an enhanced 232 slave.

- The enhanced 232 slave still supports applications having 128KB external NVM. This enables migration to SDK 6.7x based applications on existing products.

- The production test improved for a final product.

- Network key saved into zlf log file generated by Zniffer enabling support of customers developing security 2 applications.

- New Micro RF Link features such as the RxSweep writing output to a UART terminal for documentation of measurements.

- New Z-Wave PC Controller features such as inclusion controller support using SIS as trust center for security 2 key exchange, client side authentication support, configuration of Application Priority Route (APR) and Z/IP Gateway client application.

4 DETAILED DESCRIPTION

4.1 Z-Wave Plus applications are now Z-Wave certified (SDK 6.71.03+)

The Z-Wave Plus Apps are now Z-Wave certified according to the latest specifications [12]-[13] and [5]-[9].

4.2 Intermediate applications removed (SDK 6.71.02+)

The intermediate applications are now moved to SDK 6.61.01.

4.3 Z-Wave Plus applications (SDK 6.71.02+)

The Z-Wave Plus Apps could not pass Z-Wave certification.

4.4 Support for NVM ultra-deep sleep (SDK 6.71.02+)

Z-Wave modules using Adesto AT45DBxxxE chips as external NVM can now use the ultra-deep sleep mode in sleep mode.

4.5 MY frequency obsoleted (SDK 6. 71.02+)

All the MY frequency targets are obsoleted. Use ANZ frequency instead of MY frequency in Malaysia.

4.6 Z-Wave Plus applications are now Z-Wave certified (SDK 6.71.01+)

The Z-Wave Plus Apps are now Z-Wave certified. The Z-Wave Plus Apps in question are On/Off Switch, PIR Sensor, Power Strip and Wall Controller. In addition, the latest specification [5] requires Central Scene CC v3 support in Wall Controller. The Central Scene CC v3 is now implementation in this release.

4.7 Door Lock Key Pad supporting SSA and CSA (SDK 6.71.01+)

The Door Lock Key Pad now supports both SSA and CSA. SSA used as key exchange (Security Class ‘Access Control’ or ‘Authenticated’) in products supporting S2 security solution. CSA is a similar but weaker key exchange solution used when migrating from SDK 6.51.xx or SDK 6.61.xx devices having a key pad to a S2 security solution.

4.8 Intermediate serial API apps (SDK 6.71.01+)

Beside Door Lock Key Pad are all the serial API applications with the new bootloader supporting the compressed hex file format added to the SDK. The binaries are also available allowing migrating from SDK 6.51.xx or SDK 6.61.xx serial API based devices to a S2 security solution without access to Keil PK51 build environment. Furthermore, a new serial API call are introduced to adjust RF power level

runtime because the RF power level parameters do not have fixed positions anymore in the compressed hex file format. For further details, refer to section 4.18

4.9 Second-generation security solution (SDK 6.71.00+)

Notice that the mature version of the second-generation security solution (S2) is not backward compatible with former beta releases (SDK 6.70.0x). All S2 related specifications have passed final approval. For details, refer to [7].

4.10 Z-Wave Plus applications not Z-Wave certified yet (SDK 6.71.00)

Some of the Z-Wave Plus Apps are not Z-Wave certified yet. The Z-Wave Plus Apps in question are On/Off Switch, PIR Sensor, Power Strip and Wall Controller. In addition, the latest specification [6] requires Central Scene CC v3 support in Wall Controller. However, Central Scene CC v3 implementation is missing in this release.

4.11 Breakdown of command class specifications (SDK 6.71.00+)

The command class specifications are now organized in the following four documents depending on the functionality level supported:

- Z-Wave Application Command Class Specification [5] lists commands classes providing application level functionalities, which can typically be supported by any node, regardless of its Device or Role Type. This document is intended to be the “getting started” document and also describes Command Class rules, such as fields, versioning or the Node Information Frame.

- Z-Wave Management Command Class Specification [6] describes commands classes providing functionalities to support nodes’ administration and management. Those Command Classes are often mandated by the Z-Wave Plus Role Type [12] and/or Device Type [13] and provide functionalities like battery support, firmware update or associations.

- Z-Wave Transport-Encapsulation Command Class Specification [7] lists command classes used for encapsulating other command classes. They are namely: Transport, Security S0/S2, Multi Channel, CRC-16, Multi Command and Supervision.

- Z-Wave Network-Protocol Command Class Specification [8] describes all the command classes providing functionalities related to RF or Z-Wave/IP networks. Controller specific command classes are also listed in this document.

A document listing command classes and their corresponding identifier, status, category and newest version is also distributed [9]. This enables quick way to identify where to find information about a Command Class.

4.12 Serial API versioning (SDK 6.71.00+)

The serial API version is incremented from 6 to 7 enabling S2 host applications verification of the serial API used. For details, refer to [2] and [15].

```
#define SERIAL_API_VER 7
```

4.13 Supporting promiscuous mode (SDK 6.71.00+)

The static and bridge controller supports now promiscuous mode. The Serial API based static and bridge controller supports the new functionality. Only portable controller supported promiscuous mode in earlier releases. For details, refer to [2].

4.14 Obsoleted PC Applications (SDK 6.71.00+)

All PC applications substituted by Z-Wave PC based Controller v5. Furthermore, the Z-Wave PC based Controller v5 is available on ZTS as an individual program release.

4.15 Extended Triac Controller API (SDK 6.71.00+)

The Triac Controller API is extended with respect to FET leading edge mode. Notice that extensions may have an impact on current implementations. For details, refer to [2].

4.16 Enhancement of second-generation security solution (SDK 6.70.01+)

The second-generation security solution (S2) supports also a routing slave in addition to the enhanced 232 slave. However, the routing slave has some limitations with respect to the new S2 security solution because it is without an external NVM.

4.16.1 Routing Slave

The number of destinations, routing capabilities etc. are unchanged in the routing slave compared to SDK 6.61.00. However, a number of limitations are necessary with respect to Security S2 due to lack of external NVM:

- All key classes supported but only one can be active after inclusion.
- Slave routing based devices will save the “Most Recently Used” (MRU) S2 SPAN entry in critical RAM when entering Sleep mode. The MRU S2 SPAN entry is restored on power up if a valid S2 SPAN entry resides in critical RAM.
- S2 Public-Private key pair resides in Protocol part of NVR and written to NVR at production. Refer to [14] for details.
- Cannot send S2 multicast but do support S2 multicast receive.

The table below show the NVM memory budget of the full routing slave with S0/S2 functionality included is as follows:

#	Functionality	Bytes
1	Home ID, Node ID, magic bytes and SUC return routes.	25
2	Return routes: 5 destinations having 4 full hop routes each 5 destinations x (1 destination byte + 4 x (4 hop bytes + 1 aux. byte))	105
3	Keyclass byte – Which security keyclass is active – Only ONE keyclass can be active at any time, can be either S0, S2 (3 keyclasses)	1
4	S0/S2 Network key	16
5	Critical SPAN nodeID – Identifies the SPAN, which is saved in Critical RAM when going into Sleepmode and reloaded on wakeup from Sleepmode – This means that no resync (S2) is needed for the Critical SPAN nodeID after wakeup from Sleepmode. If equal to ZERO the MRU SPAN entry will be stored/restored.	1

Figure 2. Z-Wave protocol critical RAM requirements for a routing slave.

The routing slave NVM is placed in the MTP (Total of 255 Bytes) resulting in 107 Bytes available for the application.

No SPAN/MPAN are saved in NVM, which for a FLiRS node would mean that the node needs to sync every time it wakes up in case it want to communicate with another node. But to minimize the resync after Sleepmode we do save ONE SPAN in Critical RAM (retention RAM).

The Critical RAM is 128 Bytes retention memory – Currently 32 Bytes are allocated for Application and 96 bytes are allocated for protocol. The table below shows the protocol Critical RAM requirement for a Routing Slave with one SPAN entry:

#	Functionality	Bytes
1	Protocol usage: phyRfData, ResponseRoutes, FLiRS, NodeID, HomeID etc.	51
2	1 S2 SPAN	39

Figure 3. Z-Wave protocol critical RAM requirements for a routing slave (one span entry).

The ECDH keypair for routing slaves is stored in the NVR [14] and must be generated and pre-programmed during production. The updated NVR layout must be used when producing a SDK 6.70 based routing slaves. The routing slave cannot detect if the keypair is missing.

4.16.2 Inclusion controller signaling to SIS

A new Inclusion Controller Command Class allow an inclusion controller to inform the SIS that a new node has been added, and that the SIS now can perform any additional required setup operations. Examples of such additional setup operations could be:

- S2 capable devices: SIS offloads Security Bootstrapping and Z-Wave Plus configuration from Inclusion Controller.

- Non-S2 capable devices: Security S0 part if relevant is performed by the Inclusion Controller and SIS offloads Z-Wave Plus configuration.

4.16.3 Client-side authentication

New client-side authentications (CSA) feature enabling OTA firmware update from a S0 device to a S2 device. This method makes it possible to enter the QR code of a controller onto joining device (client side authentication). This feature is especially targeted door locks having a keypad.

4.17 Enhanced 232 Slave supporting 128KB external NVM (SDK 6.70.01+)

The enhanced 232 slave based applications has now external NVM space usage reduced from 256 KB in SDK 6.70.00 beta release to 128 KB in this release. The reduction is achieved by using a new compressed hex file format (*.otz) compared to the current S-Record based hex files. This enables migration to SDK 6.7x based applications from existing products based on SDK 6.51.xx. The migration requires an extra step via a modified SDK 6.61.00 based application to get the new bootloader installed.

4.18 Migrating from existing SDK 6.5x/6.6x based products (SDK 6.70.01+)

Migration from an existing slave product using 128KB external NVM and based on SDK 6.5x/6.6x to SDK 6.70.0x is now possible using OTA firmware update. However, the migration process consists of a two-stage process as shown on the figure below:

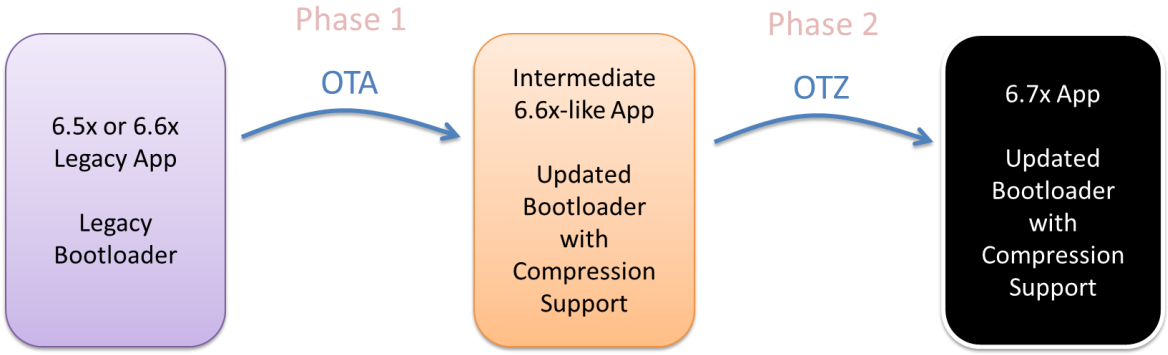


Figure 4. Migration process.

The migration process relies on a new bootloader that support a new compressed hex file format (*.otz). Due to memory shortage in external NVM when using SDK 6.7x the bootloader update must be done in an SDK 6.61.00 environment. Therefore, this release also contains a modified SDK 6.61.00 capable of building the Door Lock Key Pad application with the new bootloader as an example. This intermediate application must also be Z-Wave Plus certified to ensure interoperability in case skipping last stage.

4.19 External NVM (SDK 6.70.01+)

The generic external non-volatile memory (NVM) driver supports now additional FLASH chips as alternatives due to EOL announcements on current NVM chips. Regarding a full list of recommended EEPROM/FLASH components as external NVM, refer to [17].

4.20 Sensor PIR now based on a Routing Slave (SDK 6.70.01+)

The Sensor PIR application uses now a routing slave instead of an enhanced 232 slave. The application allow only S2 authentication when exchanging security keys. However, it is possible to configure authentication level in the source code. For details, refer to [10].

4.21 Application test interface (SDK 6.70.01+)

The Z-Wave Plus Application Framework now offers a Test Interface [16] enabling the developer to send to a device via UART using a terminal.

4.22 Second-generation security solution (SDK 6.70.00+)

A new second-generation Z-Wave transport security, the Security 2 Command Class (S2) introduced in this release to address development of devices specifically for the Internet of Things (IoT) market.

S2 is both stronger from a security standpoint, faster due to reduced communication overhead and supports multicast applications compared to the old Security Command Class (S0). S2 uses the industry-wide accepted secure key exchange using Elliptic Curve Diffie-Hellman (ECDH) and authentication via QR or PIN code. An un-authenticated solution exists also for low-cost devices and simple systems such as keyfobs etc. S2 uses rolling code avoiding negotiation of a new nonce for each image fragment. Today, gateways use Get/Report Command Classes to monitor application progress. With S2, the Supervision Command Class advertises when the transition is complete. The rolling code and Supervision Command Class functionality reduce communication overhead considerably compared to S0. S2 Multicast uses Z-Wave Broadcast primed via S2 singlecast follow-up. For further details regarding S2, refer to [7], [18] and [19].

The following applications support now both non-secure and secure (S0/S2):

- Door Lock Key Pad (enhanced 232 slave)
- Power Strip (enhanced 232 slave)
- Sensor PIR (enhanced 232 slave – changed to routing slave in SDK 6.70.01+)
- Switch On/Off (enhanced 232 slave)
- Wall Controller (enhanced 232 slave)
- PC based Controller (serial API static/bridge controller)

Controller libraries require an external NVM equal to 256 KB when using OTA/OTW firmware update as announced in SRN for SDK 6.60.00. However, the enhanced 232 slave based applications requires also an external NVM equal to 256 KB in this beta release.

Both the S0 and S2 solution is now an integrated part of the enhanced 232 slave library and the following new/modified API calls are introduced:

Security API – only enhanced 232 slave

- ZW_SetNetworkKeyS0 – Move S0 key into protocol (OTA firmware update)
- ZW_GetSecurityKeys – Returns security keys the node possess (outgoing frames)
- ApplicationSecureKeysRequested – Including controller grant all or a subset of keys.
- ApplicationSecureCommandsSupported – Supported CCs for a given key

Z-Wave Transport API – only enhanced 232 slave

- ZW_SendDataEx – Non-secure or secure S0/S2
- ZW_SendDataMultiEx – Secure S2 multicast
- ZW_Transport_CommandClassVersionGet – Supported internally by protocol

Z-Wave Basis API

- ApplicationInitSW(ZW_NVM_STATUS) – External NVM rearrange if necessary.

A new serial API command Application Node Information Command Classes Command introduced in enhanced 232 slave, which can set the command classes to be supported in the following inclusion scenarios:

- Not included
- Non-securely included
- Securely S0/S2 included

The Z-Wave Programmer and Zniffer also support S2 with respect to keys, decryption of S2 frames etc.

4.23 Improved Z-Wave Plus Framework (SDK 6.70.00+)

The Z-Wave Plus Application Framework is also improved in a number of places resulting in a simpler application development [16].

An application now has support for non-secure and secure inclusion based on security levels and the device command class lists. The benefit is one binary file handling all security levels.

Improved command class API for simpler application implementation. Command classes do now have built-in functionality to find nodes in the association database. This gives a simpler application design for sending unsolicited events. As a result, the application now only needs to handle AGI and association in initialization process.

S2 Multicast is automatically used if there is more than one destination node with the same S2 security level (S0 do not have support for multicast).

The Z-Wave Plus Application Framework has support for Supervision Command Class and automatically handles this as an integrated part of the Security 2 Command Class. If the application also requires Supervision, it is easy to setup and use it.

Handling multi-channel and endpoints is now an integrated part of the Z-Wave Plus Application Framework. This results in a simpler build and application development environment.

The task pool provides a pluggable architecture to make multitasking applications easier to write and more flexible.

The improve event scheduler provides a general event queue that handles events for the application state event machine.

4.24 External NVM (SDK 6.70.00+)

The generic external non-volatile memory (NVM) driver supports now additional FLASH chips. Regarding a full list of recommended EEPROM/FLASH components as external NVM, refer to [17].

4.25 Bridge Controller supporting repeater functionality (SDK 6.70.00+)

The bridge controller library now has the repeater functionality enabled. Notice that filename is unchanged because default was without repeater functionality.

4.26 Final product testing of RF communication (SDK 6.70.00+)

The final product test of RF communication requires only activation of the inclusion process and the new Production Test Generator. A new command combines capture of HomeID in NIF from a device not included in a Z-Wave Network and sending 10 test frames to the same HomeID/NodeID afterwards.

5 Z-WAVE API LIBRARY

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 500 Series single chips in various configurations. For a detailed description of the API calls refer to [2]. The API consists of five different libraries; a Portable Controller library, a Static Controller library, a Bridge Controller library, a Routing Slave library, and an Enhanced 232 Slave library. The type of library used depends on the application features needed.

5.1 New features

Refer to Chapter 3 and 4.

6 Z-WAVE PLUS EMBEDDED APPLICATIONS

The SDK contains code as well as compiled code for Z-Wave Plus embedded applications according to devices in [12]-[13], and command classes in [5]-[8]. The Z-Wave Plus 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 Plus embedded applications. Setting up associations is supported fully 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 500 Series Single Chip. For details about porting, refer to [2].

A Z-Wave Plus application based on earlier SDKs may also require porting to a newer version of the Z-Wave Plus Application Framework. For details, refer to [16].

6.1 Z-Wave Plus Application Certification Guide Lines

Introduction of the S2 security solution mandates additional requirements to the certification program. A large part of the S0/S2 security solution resides in the Z-Wave Protocol but parts are also located in the Z-Wave Plus Application Framework and Z-Wave Plus Applications. The application developer must therefore be aware of below list of 22 command class requirement numbers [7] applicable for slave based devices during development:

Command Class Requirement Numbers (slaves)		
CC:009F.01.0E.11.008	CC:009F.01.05.11.018	CC:009F.01.00.11.0AB
CC:009F.01.0E.11.003	CC:009F.01.05.11.017	CC:009F.01.00.11.034
CC:009F.01.0D.11.007	CC:009F.01.00.11.070	CC:009F.01.00.21.00B
CC:009F.01.0D.11.004	CC:009F.01.00.11.06E	CC:009F.01.00.21.00C
CC:009F.01.0D.11.003	CC:009F.01.00.11.061	CC:009F.01.00.21.008
CC:009F.01.0A.11.002	CC:009F.01.00.11.05E	CC:009F.01.00.21.007
CC:009F.01.08.11.007	CC:009F.01.00.11.092	
CC:009F.01.06.11.003	CC:009F.01.00.11.0A6	

Figure 5. Command class requirement numbers for slaves.

Not all listed requirements may be relevant depending on the application in question.

6.2 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 [16]. 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 [10].

6.3 My Product Plus

As an alternative to the Device Type code applications use My Product Plus, this application contains the minimum framework for developing a Z-Wave Plus application. The My Product Plus implementation is built upon the Z-Wave Plus Application Framework [16]. For detailed information, refer to [10].

6.4 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 Framework [16]. 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 [10].

6.5 PIR Sensor

The PIR Sensor application shows a presence/movement detector implementation for controlling other devices and for sending notifications. The PIR Sensor implementation built upon the Z-Wave Plus Framework [16]. The PIR Sensor is based on Sensor – Notification Device Type with Reporting Sleeping Slave (RSS) as the Role Type and routing slave. For detailed information, refer to [10].

6.6 Power Strip

The Power Strip application shows an extension block implementation to turn on a number of devices that is connected to power. Examples include lights, appliances etc. The Power Strip implementation built upon the Z-Wave Plus Framework [16]. 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 [10].

6.7 Production Test DUT

The Production Test DUT code for a device under test contains an example of how the basic tasks of testing devices in a Z-Wave network can be implemented using the Z-Wave API. Detailed information regarding the Production Test DUT code can be found in [10].

6.8 Production Test Generator

The Production Test Generator code contains an example of how the basic tasks of testing devices in a Z-Wave network can be accomplished using the Z-Wave API. The Z-Wave generator is used in conjunction with the Production Test DUT to verify the TX / RX circuits on Z-Wave enabled products. Detailed information regarding the Production Test Generator code can be found in [10].

6.9 Serial API Plus

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 [10]. In addition, detailed information about the Serial API code and how to interface to the Serial API can be found in [15] and [2].

6.10 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, wall mounted AV controllers. Device of this type can both be battery operated or mains powered. The Wall Controller implementation built upon the Z-Wave Plus Framework [16]. 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 [10].

6.11 Z-Wave Plus Application Framework

The Z-Wave Plus 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.

6.11.1 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 [16].

6.11.2 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 [16].

7 TOOLS

The SDK contains various tools for helping SW developers writing and debugging code.

NOTICE: Some of the tools are not bundled together with the SDK anymore but are available on ZTS as individual programs:

7.1 IMA Tool Box

The IMA Tool Box supports an installation and maintenance procedure, which can ensure an easy installation and provide an operational qualification of the installation. Use this tool in combination with the Serial API based hex targets, that default incorporates IMA functionality. The source code of IMA Tool Box is also included. For detailed information, refer to [1].

7.2 uVision4 Project Generator

The Keil uVision4 Project Generator makes uVision projects for a sample application. The makefile system can generate uVision projects for the Keil uVision4 IDE by calling the project generator.

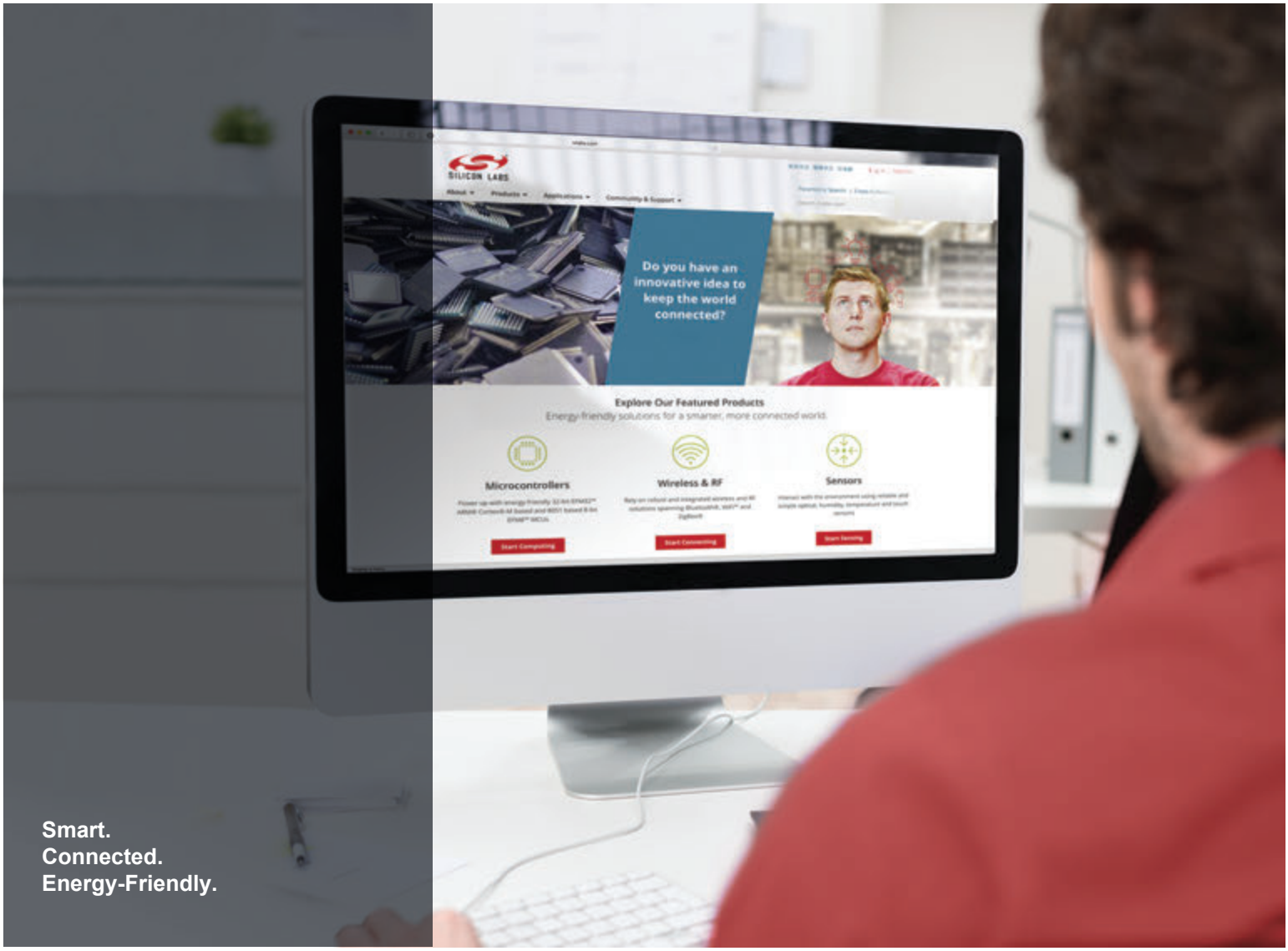
REFERENCES

[1] INS12712, Instruction, Z-Wave Network Installation and Maintenance Procedure User Guide.
[2] INS13478, Instruction, Z-Wave 500 Series Application Programming Guide v6.71.0x.
[3] SDS10242, Software Design Specification, Z-Wave Device Class Specification.
[4] SDS10865, Software Design Specification, Z-Wave Security Application Layer.
[5] SDS13781, Software Design Specification, Z-Wave Application Command Class Specification.
[6] SDS13782, Software Design Specification, Z-Wave Management Command Class Specification.
[7] SDS13783, Software Design Specification, Z-Wave Transport-Encapsulation Command Class Specification.
[8] SDS13784, Software Design Specification, Z-Wave Network-Protocol Command Class Specification.
[9] SDS13548, Software Design Specification, List of defined Z-Wave Command Classes.
[10] INS13477, Instruction, Z-Wave 500 Series SDK Contents v6.71.0x.
[11] INS12366, Instruction, Working in 500 Series Environment User Guide.
[12] SDS11846, Software Design Specification, Z-Wave Plus Role Types Specification.
[13] SDS11847, Software Design Specification, Z-Wave Plus Device Types Specification.
[14] SDS12467, Software Design Specification, 500 Series Z-Wave Chip NVR Flash Page Contents.
[15] INS12350, Instruction, Serial API Host Appl. Prg. Guide.
[16] INS13427, Instruction, Z-Wave Plus Application Framework v6.71.0x.
[17] INS12213, Instruction, 500 Series Integration Guide.
[18] SDS13349, Software Design Specification, Security considerations in Home Control installations.
[19] APL13434, Application Note, FAQ: On the use of S0, S2 and Supervision CC in product design and deployments.

INDEX

I

M



Products
www.silabs.com/products

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS®

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com