

N



Software Release Note

ZW0201/ZW0301 Developer's Kit v4.53.00

Document No.:	SRN11886
Version:	2
Description:	Software Release note of ZW0201/ZW0301 Developer's Kit version 4.5x based on Keil PK51 v7.50.
Written By:	JFR;JSI
Date:	2011-12-06
Reviewed By:	CHL;JKA;JSI;PSH;NTJ;CST;BBR
Restrictions:	Partners Only

Approved by:

Date	CET	Initials	Name	Justification
2011-12-06	09:52:29	NTJ	Niels Thybo Johansen	

This document is the property of Sigma Designs Inc. The data contained herein, in whole or in part, may not be duplicated, used or disclosed outside the recipient for any purpose. This restriction does not limit the recipient's right to use information contained in the data if it is obtained from another source without restriction.



CONFIDENTIAL

REVISION RECORD

Doc. Rev	Date	By	Pages affected	Brief description of changes
1	20110916	JFR	All	Initial draft based on SRN11385 - ZW0201/ZW0301 Developer's Kit v4.52.00/01
1	20110916	JFR	6.2	TO's #2084, #2445, #3066, #3078, #3119, #3228; #3255 & #3258 added to list of fixed protocol defects.
1	20111027	JSI	Section 6.3	Updated known protocol defects with TO's #3317, #3319, #3322 & #3323
1	20111031	JSI JFR	Section 0, 8, 9 & 10	Updated apps and tools known defects
2	20111206	JFR	Section 6.3	Updated known protocol defects with TO's #3326

Table of Contents

1	INTRODUCTION	1
2	OVERVIEW	2
3	DETAILED DESCRIPTION	3
4	COMPATIBILITY AND UPGRADE.....	16
5	VARIOUS	17
5.1	Home ID	17
5.2	Manufacturer ID.....	17
5.3	3 rd party tools to Developer's Kit	17
6	Z-WAVE API LIBRARY	18
6.1	New features	18
6.2	Fixed defects in v4.53.00	19
6.3	Known defects.....	23
7	SAMPLE CODE FOR ZW0201/ZW0301 EMBEDDED APPLICATIONS	30
7.1	Binary Sensor Sample Code	30
7.1	Secure Binary Sensor Sample Code	30
7.2	Battery Operated Binary Sensor Sample Code	31
7.3	Secure Battery Operated Binary Sensor Sample Code	31
7.4	Development Controller Sample Code.....	31
7.5	Secure Development Controller Sample Code	32
7.6	Door Bell Sample Code.....	32
7.7	Door Lock Sample Code	32
7.8	Secure Door Lock Sample Code.....	33
7.9	LED Dimmer Sample Code	33
7.10	Secure LED Dimmer Sample Code	33
7.11	My Product Sample Code	34
7.12	Production Test DUT Sample Code	34
7.13	Production Test Generator Sample Code	34
7.14	Serial API Sample Code.....	34
7.15	Utility Functions	35
8	SAMPLE CODE FOR PC LIBRARIES ETC.....	36
8.1	Z-Wave DLL	36
8.2	Z-Wave HAL	36
8.3	Z-Wave Security HAL	36
8.4	Z-Wave Command Classes	36
8.5	WinFormsUI.....	37
8.6	ZensysFramework.....	37
8.7	ZensysFrameworkUI	37
8.8	ZensysFrameworkUIControls.....	38
9	SAMPLE CODE FOR PC APPLICATIONS	39
9.1	PC Controller Sample Code	39
9.2	Z-Wave Bridge Sample Code.....	39
9.3	PC Installer Tool Sample Code	39
10	TOOLS	40
10.1	Zniffer	40
10.2	Zniffer File Converter.....	40
10.3	XML Editor.....	40

10.4 PVT and RF Regulatory	41
10.5 Enhanced Reliability Test Tool.....	42
10.6 Z-Wave Programmer.....	43
10.7 SD3402 Crystal Calibration	43
10.8 uVision3 IDE.....	43
REFERENCES	44
INDEX.....	45

Table of Figures

Figure 1, Explorer frame capabilities	10
Figure 2, Network wide inclusion capabilities	11
Figure 3, FLiRS wakeup scenario	13
Figure 4, Wakeup beam capabilities	14

1 INTRODUCTION

The ZW0201/ZW0301 Z-Wave Developer's Kit specifically designed to help developers creating Z-Wave enabled products in a fast and cost effective manner. The software consists of Z-Wave libraries supporting controller and slave devices, as well as sample code for a broad range of home automation applications. The sample code serve as an example of how to realize home automation applications using the Z-Wave protocol. To realize the application in question it is often easier to modify the existing sample code than build one from scratch.

The ZW0201/ZW0301 Z-Wave Developer's Kit version 4.53.00 is a matured version of 4.50, 4.51 and 4.52 intended for ZW0201/ZW0301 based products entering volume production. SDK 4.53.00 is build using Keil PK51 v7.50 and the new version only includes defects correction.

2 OVERVIEW

The ZW0201/ZW0301 Developer's Kit version 4.5x contains the following major enhancements:

Self-healing Network

This version enhances network reliability and robustness by the introduction of dynamic route resolution. Dynamic route resolution allows recovery of broken routes with low latency.

Network Installation

This version introduces new degrees of freedom during installation. The protocol now supports that many nodes may be included at the same time. Furthermore, the protocol now supports that nodes out of direct reach can be included via other nodes in the network.

Production

This version enables simplified production by avoiding handling of home ID allocation.

Secure applications

This version facilitates easy development of secure applications by adding an API call for random network key generation.

Z-Wave Interoperability

Z-Wave Device and Command Class Specifications has been extended with new classes supporting a broader range of applications within the Z-Wave interoperability framework.

Reduced controller libraries

This version introduces a number of new libraries. A reduced portable, installer, static and bridge controller libraries has been introduced to enable larger applications in a single chip based product.

Reduced functionality compared to 5.0x

This version has a reduced set of functionalities compared to ZDK 5.0x. Some Z-Wave library variants and protocol features has been removed; such as Zensor nodes due to code space shortage. See the following section "Detailed Description".

Discontinued ZW0102

This version has discontinued the support of the ZW0102 Single Chip.

Application Size

This version uses the PK51 Keil Compiler to improve optimization of application/protocol memory footprint.

Time to Market

This version contains improved development tools, production software and documentation in order to accelerate time-to-market for the Z-Wave partners.

3 DETAILED DESCRIPTION

Keil PK51 development environment (4.52.01+)

All v4.5x SDK's is built and tested on Keil PK51 v7.50 except SDK 4.52.00, which is based on Keil PK51 v9.00. It is not possible to use earlier Keil PK51 versions than v9.00 in connection with SDK 4.52.00. Newer versions than the one used in the SDK should also apply according to Keil's support notes.

Random HomeID storing moved from power up to inclusion process (4.52.00+)

Writing home ID to external non-volatile memory immediately after power up of the module avoided. Random HomeID is still generated at startup to avoid having to turn off the radio later for generation of the random ID. However, it is not stored in non-volatile memory until the node needs to use the HomeID and at that point the non-volatile memory is checked too see if the HomeID is zero and if it isn't then the generated HomeID is discarded again. The sample applications are not affected by this change.

New slave libraries without automatic update of return routes (4.52.00+)

The following libraries are now available in a _nouupdate version without the functionality that updates the return routes automatically:

- slave_enhanced_232_nouupdate_ZW020x / slave_enhanced_232_nouupdate_ZW030x
- slave_enhanced_nouupdate_ZW020x / slave_enhanced_nouupdate_ZW030x
- slave_routing_nouupdate_ZW020x / slave_routing_nouupdate_ZW030x

The serial API applications below are based on the new libraries as follows:

- SerialAPI_Slave_Enhanced_232_Mr (slave_enhanced_232_nouupdate_ZW020x /30x)
- SerialAPI_Slave_Enhanced_Mr (slave_enhanced_nouupdate_ZW020x /30x)
- SerialAPI_Slave_Routing_Mr (slave_routing_nouupdate_ZW020x /30x)

Discontinued ApplicationRfNotify (4.52.00+)

To save code space the functionality to support an external RF amplified has been removed, obsolescing the ApplicationRfNotify() API call.

New extended enhanced slave (4.51+)

A new enhanced 232 slave supporting up to 232 associations backed up by return routes is introduced in this version. Note: It is only possible to assign up to five return routes at a time.

The new libraries are as follows:

```
ZW_slave_enhanced_232_ZW020xs.lib
ZW_slave_enhanced_232_ZW030xs.lib
ZW_slave_enhanced_232_noflirs_nomr_ZW020xs.lib
ZW_slave_enhanced_232_noflirs_nomr_ZW030xs.lib
```

Finally, additional serial API application supporting the new enhanced 232 slave is introduced in this version:

```
SerialAPI_Slave_Enhanced_232\serialapi_slave_enhanced_232_ZW020x_ANZ.hex
...
SerialAPI_Slave_Enhanced_232\serialapi_slave_enhanced_232_ZW030x_US.hex
SerialAPI_Slave_Enhanced_232_Mr\serialapi_slave_enhanced_232_mr_ZW020x_ANZ.hex
...
SerialAPI_Slave_Enhanced_232_Mr\serialapi_slave_enhanced_232_mr_ZW030x_US.hex
SerialAPI_Slave_Enhanced_232_Noflirs\serialapi_slave_enhanced_232_noflirs_ZW020x_ANZ.hex
...
SerialAPI_Slave_Enhanced_232_Noflirs\serialapi_slave_enhanced_232_noflirs_ZW030x_US.hex
```

Routing resolution enhancement (4.51+)

This version introduces a routing resolution enhancement, which qualitatively will reduce the number of explorer frames in a system.

When all cached and routing algorithm determined routes fails the routing resolution mechanism will try a direct attempt. In the event that the direct attempt also fails the route resolution algorithm will use an explorer frame as last resort in case TRANSMIT_OPTION_EXPLORE set.

Additionally the protocol has been improved with return route cleanup to get rid of latency due to nonworking routes. The new features will be available in all libraries.

The table below describes how the routing algorithms for slave libraries work in conjunction with return routes and response routes.

Library	Routing algorithm	Return route	Response route
Routing slave	<p>If TRANSMIT_OPTION_ACK is set and destination is available in response routes, try response route.</p> <p>If TRANSMIT_OPTION_ACK and TRANSMIT_OPTION_AUTO_ROUTE are set then try return routes.</p> <p>If TRANSMIT_OPTION_ACK and TRANSMIT_OPTION_AUTO_ROUTE are set then try direct.</p> <p>If TRANSMIT_OPTION_ACK and TRANSMIT_OPTION_EXPLORE are set, issue an explore frame as last resort.</p>	Five destinations having up to four routes each. Return routes can also contain direct attempts.	The FIFO contains up to two routes. New routes/direct are qualified for return route insertion by checking if the destination exist and route/direct do not exist. In that event the new route/direct entry will be placed either in a free route or the one having lowest priority.
Enhanced slave	<p>If TRANSMIT_OPTION_ACK is set and destination is available in response routes, try response route.</p> <p>If TRANSMIT_OPTION_ACK and TRANSMIT_OPTION_AUTO_ROUTE are set then try return routes.</p> <p>If TRANSMIT_OPTION_ACK and TRANSMIT_OPTION_AUTO_ROUTE are set then try direct.</p> <p>If TRANSMIT_OPTION_ACK and TRANSMIT_OPTION_EXPLORE are set, issue an explore frame as last resort.</p>	Five destinations having up to four routes each. Return routes can also contain direct attempts.	The FIFO contains up to two routes. New routes/direct are qualified for return route insertion by checking if the destination exist and route/direct do not exist. In that event the new route/direct entry will be placed either in a free route or the one having lowest priority.
Enhanced 232 slave	<p>If TRANSMIT_OPTION_ACK is set and destination is available in response routes, try response route.</p> <p>If TRANSMIT_OPTION_ACK and TRANSMIT_OPTION_AUTO_ROUTE are set then try return routes.</p> <p>If TRANSMIT_OPTION_ACK and TRANSMIT_OPTION_AUTO_ROUTE are set then try direct.</p> <p>If TRANSMIT_OPTION_ACK and TRANSMIT_OPTION_EXPLORE are set, issue an explore frame as last resort.</p>	232 destinations having up to four routes each. Return routes can also contain direct attempts.	The FIFO contains up to one route. New routes/direct are qualified for return route insertion by checking if the destination exist and route/direct do not exist. In that event the new route/direct entry will be placed either in a free route or the one having lowest priority.

The protocol will update response routes in a slave in the following situations:

- When receiving a successful explorer frame route.
- When receiving a successful routed/direct request from another node.
- When receiving a successful acknowledge for a transmitted explorer frame.
- When receiving a successful acknowledge for a transmitted routed/direct frame.

This version discontinues the TRANSMIT_OPTION_RETURN_ROUTE flag as the routing algorithm must always try return routes in case they exist. This version also removes the possibility to bypass Association Command Class by sending directly to all assigned return routes. This applies for both routing slave and enhanced slave.

Note: ZW_DeleteReturnRoute clears return routes for all nodes in the specified enhanced slave. ZW_AssignReturnRoute can assign return routes to all 232 nodes in the specified enhanced slave.

The table below describes how the routing algorithms for controller libraries work in conjunction with last working route.

Library	Routing algorithm	Last Working Route
Portable controller	<p>If last working route do not exist and TRANSMIT_OPTION_ACK set. Try direct with retries.</p> <p>If last working route exist and TRANSMIT_OPTION_ACK set. Try direct without retries. In case it fails, try the last working route. In case the last working route also fails, purge it.</p> <p>If TRANSMIT_OPTION_ACK and TRANSMIT_OPTION_AUTO_ROUTE are set then calculate up to two routing attempts per entry/repeater node. In case TRANSMIT_OPTION_EXPLORE set, a maximum number limits number of tries.</p> <p>If TRANSMIT_OPTION_ACK and TRANSMIT_OPTION_AUTO_ROUTE are set, then direct with retries.</p> <p>If TRANSMIT_OPTION_ACK and TRANSMIT_OPTION_EXPLORE are set then issue an explore frame as last resort.</p>	<p>232 destinations having up to one route/direct each.</p> <p>Last working route can also contain direct attempts.</p>
Installer controller	<p>If last working route do not exist and TRANSMIT_OPTION_ACK set. Try direct with retries.</p> <p>If last working route exist and TRANSMIT_OPTION_ACK set. Try direct without retries. In case it fails, try the last working route. In case the last working route also fails, purge it.</p> <p>If TRANSMIT_OPTION_ACK and TRANSMIT_OPTION_AUTO_ROUTE are set then calculate up to two routing attempts per entry/repeater node. In case TRANSMIT_OPTION_EXPLORE set, a maximum number limits number of tries.</p> <p>If TRANSMIT_OPTION_ACK and TRANSMIT_OPTION_AUTO_ROUTE are set, then direct with retries.</p> <p>If TRANSMIT_OPTION_ACK and TRANSMIT_OPTION_EXPLORE are set then issue an explore frame as last resort.</p>	<p>232 destinations having up to one route/direct each.</p> <p>Last working route can also contain direct attempts.</p>
Static controller	<p>If last working route do not exist and TRANSMIT_OPTION_ACK set. Try direct when neighbors.</p> <p>If last working route exist and TRANSMIT_OPTION_ACK set. Try the last working route. In case the last working route fails, purge it and try direct if neighbor.</p> <p>If TRANSMIT_OPTION_ACK and TRANSMIT_OPTION_AUTO_ROUTE are set then calculate up to two routing attempts per entry/repeater node. In case TRANSMIT_OPTION_EXPLORE set, a maximum number</p>	<p>232 destinations having up to one route/direct each.</p> <p>Last working route can also contain direct attempts.</p>

	<p>limits number of tries.</p> <p>If TRANSMIT_OPTION_ACK and TRANSMIT_OPTION_AUTO_ROUTE are set, then direct with retries.</p> <p>If TRANSMIT_OPTION_ACK and TRANSMIT_OPTION_EXPLORE are set then issue an explore frame as last resort.</p>	
Bridge controller	<p>If last working route do not exist and TRANSMIT_OPTION_ACK set. Try direct when neighbors.</p> <p>If last working route exist and TRANSMIT_OPTION_ACK set. Try the last working route. In case the last working route fails, purge it and try direct if neighbor.</p> <p>If TRANSMIT_OPTION_ACK and TRANSMIT_OPTION_AUTO_ROUTE are set then calculate up to two routing attempts per entry/repeater node. In case TRANSMIT_OPTION_EXPLORE set, a maximum number limits number of tries.</p> <p>If TRANSMIT_OPTION_ACK and TRANSMIT_OPTION_AUTO_ROUTE are set, then direct with retries.</p> <p>If TRANSMIT_OPTION_ACK and TRANSMIT_OPTION_EXPLORE are set then issue an explore frame as last resort.</p>	<p>232 destinations having up to one route/direct each.</p> <p>Last working route can also contain direct attempts.</p>

The protocol will update last working routes in a controller in the following situations:

- When receiving a successful explorer frame route.
- When receiving a successful routed/direct request from another node.
- When receiving a successful acknowledge for a transmitted explorer frame.
- When receiving a successful acknowledge for a transmitted routed/direct frame.

Routing/Enhanced Slaves without possibility to act as FLiRS node and manual routing (4.51+)

This version contains new libraries to increase application code space in slave nodes, which does not act as FLiRS nodes and do not need the manual routing option. However, the libraries still support wakeup beam functionality to assure communication to FLiRS nodes. The new libraries are as follows:

```
ZW_slave_enhanced_noflirs_nomr_ZW020xs.lib
ZW_slave_enhanced_noflirs_nomr_ZW030xs.lib
ZW_slave_routing_noflirs_nomr_ZW020xs.lib
ZW_slave_routing_noflirs_nomr_ZW030xs.lib
```

Finally this version contains, additional serial API application supporting the new libraries:

```
SerialAPI_Slave_Routing_Noflirs\serialapi_slave_routing_noflirs_ZW020x_ANZ.hex
...
SerialAPI_Slave_Routing_Noflirs\serialapi_slave_routing_noflirs_ZW030x_US.hex
SerialAPI_Slave_Enhanced_Noflirs\serialapi_slave_enhanced_noflirs_ZW020x_ANZ.hex
...
SerialAPI_Slave_Enhanced_Noflirs\serialapi_slave_enhanced_noflirs_ZW030x_US.hex
```

Watchdog handling support in serial API (4.51+)

Some PC based applications cannot guarantee kicking the watchdog before timeout causing the watchdog to reset the Z-Wave single chip unintentionally. Hence, this version provides the following serial API functions to avoid the issue:

- Start watchdog: Enable watchdog and start kick watchdog in ApplicationPoll
- Stop watchdog: Disable watchdog and stop kick watchdog in ApplicationPoll

Watchdog handling disabled when powered up and Sleep/FLiRS mode will temporary stop watchdog.

Z-Wave Programmer firmware source code (4.51+)

The source code of the ATmega ZDP02A/ZDP03A Z-Wave Programmer firmware is now available on the ZDK 4.51 including detailed communication protocol description.

Removed SUC controller's capability to request node info on protocol level (4.51+)

This version has moved application related activities away from the protocol such as requesting node info to obtain supported/controlled command classes for a given device. The capability to request of node info on application level is not affected.

Discontinue slave library (4.51+)

This version replaces the slave library with the routing slave library. This is done without any additional hardware cost.

PC Based Controller Sample Application (4.50 Patch1+)

This version provides the PC based Controller sample application in a non-secure and secure version.

Door Lock Embedded Sample Application (4.50 Patch1+)

This version provides the FLiRS based Door Lock embedded sample application in a non-secure and secure version.

User Guides (4.50 Patch1+)

In this ZDK are the following user guides released describing the latest features:

- INS10263 – Development Controller User Guide
- INS10250 – Z-Wave DLL Developer's Guide
- INS10241 – PC Installer Tool Application User Guide
- INS10240 – PC based Controller User Guide
- INS10245 – Z-Wave UPnP Bridge User Guide
- INS10680 – Z-Wave XML Editor User Guide

Supporting India (IN) frequency (4.50 Patch1+)

Embedded sample applications, tools etc. now support India frequency. EU Z-Wave modules support India frequency.

Self-healing Network

This release supports fast route recovery for ZDK 4.5 nodes, while staying fully compatible with Z-Wave releases prior ZDK 4.5.

Using explorer frames, a command, e.g. turn on a lamp, may be delivered to the intended target even before route resolution has been completed. The result is low response time. Working routes are cached so that future commands may follow the same route at the same speed as classic Z-Wave routing.

To enable dynamic route resolution a new transmit option TRANSMIT_OPTION_EXPLORE must be appended to the well known send API calls. This instructs the protocol to transmit the frame as an explore frame to the destination node if source routing fails. It is also possible to specify the maximum number of source routing attempts before the explorer frame kicks in using the API call ZW_SetRoutingMAX. Default value is five with respect to maximum number of source routing attempts. A ZDK 4.5 controller uses the routing algorithm from 5.02 to address nodes from ZDK's not supporting explorer frame. The routing algorithm from 5.02 ignores the transmit option TRANSMIT_OPTION_EXPLORE flag and maximum number of source routing attempts value. Note: An explorer frame cannot wake up FLIRS nodes.

The response route mechanism replaced by one last working route for each destination in the controllers routing table. Discontinued the API call ZW_LockRoute because each destination has a route making response route handling irrelevant.

Libraries supporting explorer frame capabilities are follows:

Z-Wave 200/300 Series Library	Able to issue an explorer frame	Able to forward an explorer frame
ZW_CONTROLLER_INSTALLER	YES	NO
ZW_CONTROLLER_PORTABLE	YES	NO
ZW_CONTROLLER_STATIC_NOSUC	YES	YES
ZW_CONTROLLER_STATIC_NOSUC_NOEP	YES	NO
ZW_CONTROLLER_STATIC_NOEP_MR	YES	NO
ZW_CONTROLLER_BRIDGE_NOSUC_NOEP	YES	NO
ZW_SLAVE_ROUTING	YES	YES
ZW_SLAVE_ENHANCED	YES	YES
ZW_SLAVE	YES	YES

Figure 1, Explorer frame capabilities

Network Installation

Developer's Kit 4.5 offers new degrees of freedom during installation.

1. Nodes may be included with a central static controller or a battery-powered portable controller.
2. Nodes may be included one at a time or many at the same time.
3. Nodes may be in their final location or in a convenient location prior to installation.

Developer's Kit 4.5 controllers and slaves are 100% compatible with previous releases, while offering significantly improved discovery mechanisms for Developer's Kit 4.5 nodes and future releases.

Note: If creating a network of Developer's Kit 4.5 nodes as well as nodes of earlier releases, all nodes must be in their final location when performing the inclusion. This is due to that earlier releases of slaves do not respond to explorer frames and earlier releases of controllers cannot perform dynamic route resolution.

ZDK 4.5x libraries supporting network wide inclusion capabilities are follows:

Z-Wave 200/300 Series Library	Able to act as NWI center	Able to be included via the NWI mechanism
ZW_CONTROLLER_INSTALLER	YES	YES
ZW_CONTROLLER_PORTABLE	YES	YES
ZW_CONTROLLER_STATIC_NOSUC	YES	YES
ZW_CONTROLLER_STATIC_NOSUC_NOREP	YES	YES
ZW_CONTROLLER_STATIC_NOREP_MR	YES	YES
ZW_CONTROLLER_BRIDGE_NOSUC_NOREP	YES	YES
ZW_SLAVE_ROUTING	NO	YES
ZW_SLAVE_ENHANCED	NO	YES
ZW_SLAVE	NO	YES

Figure 2, Network wide inclusion capabilities

Production

Enable simplified production by avoiding handling of Home ID allocation and programming due to automatically random generated Home ID on the Z-Wave Module. The generated random Home ID is selected within the range 0xC0000000-0xFFFFFFFF corresponding to one billion available Home IDs.

The automatically random generated Home ID is initiated in an external EEPROM (controllers and enhanced slave) or FLASH (slave or routing slave) when Home ID equal to 0x00000000. The Z-Wave Protocol will at startup check if Home ID is equal to 0x00000000 and if this is the case write a new random Home ID.

Duplicated nodes

Occasionally a user may reset the primary controller before all devices are excluded from the network. If the same primary controller is used to build a new network a home/node ID could be allocated twice. The Developer's Kit 4.5 new random home ID mechanism effectively eliminates the duplicated nodes issue because the home ID will differ for each network installation.

Secure applications

The API call `ZW_GetRandomWord` is added to all libraries enabling generation of security keys. The serial API also supports this API call. Refer to [4] for further details.

Notice that binaries and source code for AES encryption/decryption engine are not part of this Developer's Kit due to export restrictions to certain countries.

Detect SUC/SIS presence on application level

Routing and enhanced slaves now also supports the API call `ZW_GetSUCNodeID` to allow the application to retrieve the SUC/SIS node ID. Notice that another controller must in advance configure the SUC/SIS node ID in the routing or enhanced slave using the API call `ZW_AssignSUCNodeID`.

Assignment of SUC/SIS role

Assignment of SUC/SIS role is now default enabled in static controllers to accommodate advanced network capabilities without relying on prior enabling locally. An application program must call API call `ZW_EnableSUC(FALSE)` to reject assignment of the SUC/SIS role in the product.

Enhanced multicast support in bridge controller

This release supports enhanced multicast support to avoid the "popcorn effect" on the physical nodes mapped as virtual nodes. The `ApplicationSlaveCommandHandler` is now called when receiving a multicast and the `rxStatus` returns `RECEIVE_STATUS_TYPE_MULTI`. Furthermore, a bitmask indicates the virtual slaves addressed by the multicast. A multicast addressing the bridge itself will also trigger the `ApplicationCommandHandler`. Finally a virtual slave can act as source in a multicast transmitted from `ApplicationSlaveCommandHandler`.

Default power when doing neighbor search

In order to create more conservative and robust 40kbps links between nodes, the default value during neighbor search is now normal power minus 6dB instead of minus 2dB in previous release. The node itself can change the power level by calling the API call `ZW_RFPowerlevelRediscoverySet`.

Supporting Malaysia (MY) frequency

Embedded sample applications, tools etc. now support Malaysia frequency.

Embedded sample applications

All sample applications use now as default normal power during inclusion. A serial API slave and routing slave added.

Reduced functionality compared to 5.0x

ZDK v4.50 release still support FLiRS nodes but minor reductions has been introduced with respect to controllers due to code space shortage.

Always listening v4.50 controllers and slaves route the wakeup beam information when operating as repeaters. In addition can v4.50 slaves send a wakeup beam to a FLiRS.

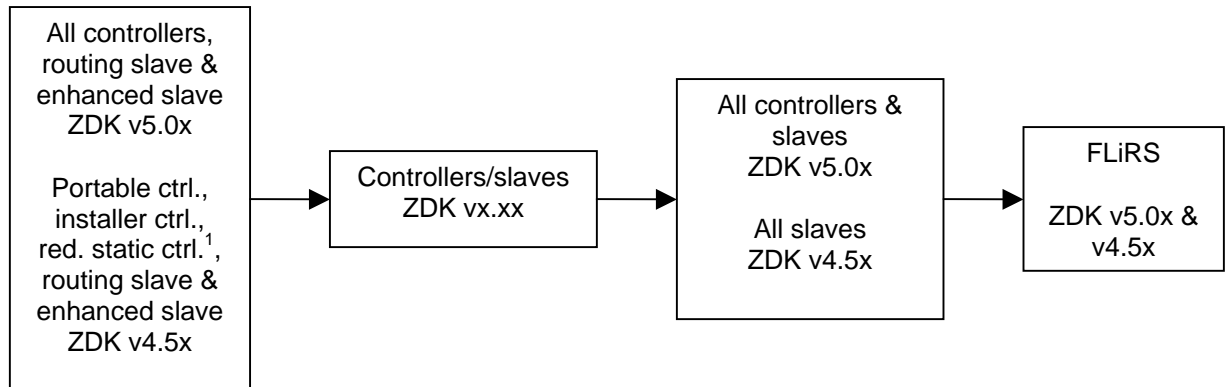


Figure 3, FLiRS wakeup scenario

1) Static controller reduced with respect to SUC/SIS role and repeater functionality.

ZDK 4.5x libraries supporting FLiRS capabilities are as follows:

Z-Wave 200/300 Series Library	Able to be a FLiRS node	Able to transmit beam when acting as repeater	Able to create route containing beam
ZW_CONTROLLER_INSTALLER	NO	NO	YES
ZW_CONTROLLER_PORTABLE	NO	NO	YES
ZW_CONTROLLER_STATIC_NOSUC	NO	NO	NO
ZW_CONTROLLER_STATIC_NOSUC_NOREP	NO	NO	YES
ZW_CONTROLLER_STATIC_NOREP_NOMR	NO	NO	NO
ZW_CONTROLLER_BRIDGE_NOSUC_NOREP	NO	NO	NO
ZW_SLAVE_ROUTING	YES	YES	YES¹
ZW_SLAVE_ENHANCED	YES	YES	YES¹
ZW_SLAVE	NO	YES	NO

Figure 4, Wakeup beam capabilities

1) Only when return routes has been assigned by a controller capable of creating routes containing beams.

This release has removed Zensor Net support completely and does therefore not supporting the smoke sensor embedded sample application.

This release has removed "I'm lost" support provided by the API call `ZW_RediscoveryNeeded` in all libraries except routing slaves and enhanced slaves due to code space shortage.

This release does not notify nodes in the network when a SUC/SIS node is introduced in the network.

This release has discontinued partial network update in all controllers supporting SUC/SIS functionality due to code space shortage. The API call `ZW_RequestNetWorkUpdate` cannot buffer up to 64 network changes anymore. The API call will now result in a full replication in case a change is registered, this replication is protocol information only and no application data is copied. Nodes which are removed and notified to SUC/SIS cannot be removed by the network update in the receiving controller when different from v4.5x and v5.0x.

Discontinuation of ZW0102 Single Chip

All libraries and sample applications for the ZW0102 target is discontinued. The provided tools still support 100 Series.

Reduced controller libraries

This release introduce three new reduced static controller libraries introduced as follows:

1. `zw_controller_static_nosuc_ZW0x0xs.lib`
(reject to act as SUC/SIS)
2. `zw_controller_static_nosuc_norep_ZW0x0xs.lib`
(reject to act as SUC/SIS and will not be used as repeater in the mesh network)
3. `zw_controller_static_norep_nomr_ZW0x0xs.lib`
(do not support manual routing and will not be used as repeater in the mesh network)

The two first reduced static controller libraries announces themselves as a static controller but reject both to accept the role as SUC/SIS in the Z-Wave network. The only difference between the two reduced static controller libraries is that `zw_controller_static_nosuc_norep_ZW0x0xs.lib` is not inserted into the routing table and will therefore not act as repeater in the Z-Wave network. This library is typically used in stationary devices where the power is switched off occasionally (e.g. an USB dongle). The last do not support repeater functionality and manual routing.

The bridge controller comes in one version without repeater and SUC/SIS functionality.

1. `zw_controller_bridge_nosuc_norep_ZW0x0xs.lib`
(reject to act as SUC/SIS and will not be used as repeater in the mesh network)

Due to code space shortage this release discontinue the static and bridge controller libraries:

1. `zw_controller_static_ZW0x0xs.lib`
2. `zw_controller_bridge_ZW0x0xs.lib`

Finally, serial API embedded applications are available for all controller variants.

Z-Wave application size

In order to minimize the overall memory footprint the code optimization has been moved to the last step in the build process to assure that all reductions across the library and application are exploited. The new Keil PK51 Extended Linker is also introduced to achieve the best possible code optimization. It is therefore difficult to estimate the available application memory for the relevant libraries and single chips before the last optimization pass is done on the entire code comprising of both application and library.

The ZW0201/ZW0301 flash based non-volatile memory used by slave and routing slave libraries supports typical 70k write cycles. In case the available write cycles are not sufficient then the enhanced slave can be used because it uses an external EEPROM as non-volatile memory. The external EEPROM typically supports 1,000,000 write cycles.

4 COMPATIBILITY AND UPGRADE

Porting Developer's Kit v4.0x, v4.1x and 4.2x based applications requires the following changes:

- Use the Keil PK51 compiler. It is not possible to use Keil CA51 compiler.
- Copy the Developer's Kit v4.xx based application to the directory C:\DevKit_4_51\Product\Your_Sample_Application, except the makefiles.
- Copy the makefiles from another sample application using the same library on Developer's Kit v4.5x to the directory C:\DevKit_4_5x\Product\Your_Sample_Application.
- Modify the makefiles to handle Your_Sample_Application.
- Add the API call ApplicationRfNotify in case the ZW0301 target is used.
- Add the parameter bWakeupReason in the API call ApplicationInitHW in case the ZW0201 and ZW0301 target is used.

Detailed documentation is available to ensure easy porting of ZW0102 and ZW0201 based applications to the new cost effective ZW0301 Single Chip. For details, refer to [8] and [3].

The Z-Wave Programmer can download hex files directly to the Z-Wave module making the Equinox PPC files obsolete. For details, refer to [16].

5 VARIOUS

5.1 Home ID

Each Z-Wave network uses a unique Home ID as address to differ between the networks when the individual nodes are addressed. The Developer's Kit CD contains a number of external EEPROM files with individual Home ID's embedded. It's essential to use different EEPROM file for the different controllers. If the same EEPROM file is loaded in different controllers then it is possible to create adjacent networks with the same Home ID, which will have the effect that they can confuse each other and control nodes unintentional. The allocated Home IDs (0x000000yy) must only be used for development purposes. Please contact Sigma Designs via zensys_support@sigmadesigns.com for allocation of a Home ID range as soon as you need them for production of your prototype controller units. Alternatively use the new random Home ID generation mechanism in ZDK 4.5x. For further details about Home ID's etc. refer to [4].

5.2 Manufacturer ID

The manufacturer ID is used by the Manufacturer Specific command class to announce manufacturer and product identification of a Z-Wave enabled device. This information enables also GUI developers to show your company logo and type of product on the user interface. Please contact zensys_support@sigmadesigns.com to acquire a manufacturer ID. An official list of manufacturer IDs can be found in the Z-Wave Device Class Specification [11]. Your company will first be added to the official list when a confirmation is sent to zensys_support@sigmadesigns.com.

The Manufacturer Proprietary Command Class uses also the manufacturer ID as parameter to obtain a more "secure" way to communicate proprietary commands between nodes. Please note, that the use of both the Proprietary and Manufacturer Proprietary Command Classes is restricted, and written approval from Sigma Designs is necessary before use. Please contact zensys_support@sigmadesigns.com for more information.

5.3 3rd party tools to Developer's Kit

The PK51 Professional Developer's Kit for the 8051 microcontroller is required before it is possible to develop Z-Wave applications. Notice that it is not possible to use Keil CA51 Developer's Kit anymore. Regarding 3rd party SW tools refer to [4].

It is also possible to buy the Keil Developer's Kit via Sigma Designs.

It is not possible to build the 200/300 Series secure sample applications on a secure ZDK due to the missing files 8051_AES_common.a51 and 8051_AES_core.a51 in Devkit\Products\util_func directory. These files must be acquired from the rightful IP owner SIC <http://jce.iaik.tugraz.at/>

6 Z-WAVE API LIBRARY

The Developer's Kit contains version 3.34 of the Z-Wave API library for ZW0201/ZW0301. For a detailed description of the API library refer to [4], which is included on the Developer's Kit. The API consists of eight different libraries; a Portable Controller library, a Static Controller library, an Installer Controller library, a Bridge Controller library, a Slave library, a Routing Slave library, an Enhanced Slave library, and a Slave Sensor library.

Updating the header file `ZW_classcmd.h` in the include directory will cause a warning when building the embedded sample applications. To avoid the warning change `LFLAGS=DW (13,16)` to `LFLAGS=DW (13,16,25)` in the `Makefile.common_ZW0x0x_appl` located in the common directory.

6.1 New features

Refer to Chapter 2.

6.2 Fixed defects in v4.53.00

The following defects are fixed compared to Developer's Kit v4.52.00/01:

Headline/TO:	A slave node doesn't acknowledge frames send to home ID / node ID equal to zero when in production test mode (TO #2084)
Library:	All libraries
ASIC:	200 and 300 series
Detailed Description:	When a slave node enters production test mode by returning false in ApplicationInitHW() it should acknowledge frames send to home ID = 0x00000000 and node ID = 0x00.
Consequence:	Communication cannot be tested using home ID / node ID equal to zero.
Resolution:	Slave now operates as expected when in production test mode.
Headline/TO:	A Z-Wave node cannot use Explorer Frames to send data to virtual nodes on a Bridge Controller (TO #2445)
Library:	All libraries
ASIC:	200 and 300 series
Detailed Description:	Protocol does not allow explorer frames to virtual nodes.
Resolution:	Explorer frames can now be used with virtual nodes.

Headline/TO:	Controller does not remove neighbor status for a given node when instructed by ZW_RequestNodeNeighborUpdate() (TO #3066)
Library:	Controllers without repeater functionality (norep in library file name)
ASIC:	200 and 300 series
Detailed Description:	<p>Routing ending at controllers without repeater functionality can fail in sparse network topologies. This happens because the target non-repeater controller will appear to be a neighbor of every repeater. Most of these apparent links will not work, and all routing attempts are used on the invalid links and an actual working route may never be tried.</p> <p>Return routes and SUC return routes can also fail if they are destined for a controller without repeater functionality.</p> <p>Routes beginning at controllers without repeater functionality can have increased latency because a direct range transmission is tried before routing.</p>
Consequence:	<p>Routing to controllers without repeater functionality may fail completely.</p> <p>Routing from controllers without repeater functionality may experience increased latency.</p>
Resolution:	After receiving a rangeinfo the neighbor status is now updated correctly.
Headline/TO:	Bridge controller cannot receive frames after inclusion of a virtual node by the bridge controller itself (TO #3078)
Library:	Bridge Controller
ASIC:	200 and 300 series
Detailed Description:	Bridge controller do not enter receive mode after inclusion of a virtual node making it impossible to send frames to bridge controller. The bridge controller in question includes the virtual node. Another inclusion controller can include virtual nodes on behalf of the bridge controller successfully.
Consequence:	RF communication fails.
Resolution:	Bridge now enters RF Receive after inclusion of a virtual node.

Headline/TO:	Timeout waiting for routed acknowledge is too short when sending to FLiRS nodes (TO #3119)
Library:	All
ASIC:	200 and 300 series
Detailed Description:	When a node is sending a routed frame to a FLiRS node, then the timeout used for waiting for a routed ack is not long enough if there is other traffic in the system that causes the repeater to make a backoff or a retransmission. If there is no delay in the frame flow and no retransmissions then the timeout is long enough.
Consequence:	The node sending the frame will timeout waiting for the routed ack and start a new routing attempt to the FLiRS node (if another route exists) while frames from the first routing attempt is still being transmitted. This can in rare cases lead to a situation where the source of the frame is blocked by network traffic and is unable to transmit until the traffic fades out.
Resolution:	Routed acknowledge timeout extended when destination is a FLiRS.
Headline/TO:	Too long Explore Search Result frame can bring routing devices in deep sleep (TO #3228)
Library:	All Explore Repeater libraries
ASIC:	200 and 300 series
Detailed Description:	Explore Search Result frames with CRC value 0x02 brings repeating node in deep sleep. Overflow on buffer "Explore Search Result" result in overwriting SleepMode parameter.
Consequence:	Repeater goes in deep sleep and can only be awoken by powercycle or Ext Int.
Resolution:	Buffer overflow resolved.
Headline/TO:	ACK timeout is too long when routing to FLiRS (TO #3255)
Library:	All libraries
ASIC:	200 and 300 series
Detailed Description:	When a node send a routed frame to a FLiRS node then it will set its timeout to 1300ms if the destination is a FLiRS node. But the correct way to set the timeout would be to only set the timeout to 1300 ms if the next node is the FLiRS node, not just based on the destination.
Consequence:	Routing to FLiRS nodes takes much longer than expected if more than one route has been tried.
Resolution:	Only set ACK timeout to 1300ms if next node is the FLiRS node.

Headline/TO:	Portable Controller tries only direct and routing is somehow lost (TO #3258)
Library:	Controller libraries
ASIC:	200 and 300 series
Detailed Description:	Controller should when including keep the initial route all through the inclusion even if it fails sometimes and another route is used successfully.
Consequence:	NWI inclusion sometimes fails.
Resolution:	Keep initial route all through the NWI inclusion process.

6.3 Known defects

Headline/TO:	Using ZW_SetRFReceiveMode to save power during 9.6kbps silent acknowledge communication (TO #1660)
Library:	All libraries
ASIC:	100, 200 and 300 series
Detailed Description:	In battery operated devices the application calls ZW_SetRFReceiveMode after reception of data to reduce battery consumption. This enables processing of data without wasting power on the radio. After introduction of silent acknowledge using 9.6kbps communication is it possible to shut down RF to fast by calling ZW_SetRFReceiveMode before the routing is completed. Silent acknowledge is not received by source node since the radio is turned off. The source will retransmit the routed acknowledge and thereby turn on the radio.
Consequence:	Increase latency and battery consumption.
Workaround:	Delay calling ZW_SetRFReceiveMode after calls to ZW_SendData().
Headline/TO:	Slaves do not cache direct communication (TO #1770).
Library:	Slave, Routing Slave and Enhanced Slave libraries
ASIC:	200 and 300 series
Detailed Description:	Slaves do not cache direct communication assuming it communicates with an always-listening node. When a FLiRS node is requesting information from a slave the transmitted response is without a wakeup beam. Therefore, if the FLiRS node have entered frequently listening mode to save battery it will not hear response from slave.
Consequence:	FLiRS node may not hear response from slaves. One example could be a missing report command requested by the FLiRS node.
Workaround:	The FLiRS node requesting information should stay awake until response has been received or a timeout has expired.

Headline/TO:	In special cases can the routing algorithm use long time to determine that no alternative routes exist (TO #1831).
Library:	All controller libraries
ASIC:	200 and 300 series
Detailed Description:	A source and destination node can see each other but the destination node does not have other nodes as neighbours in the network. The source has many neighbours in the network. In case the destination node is turned off the direct try fails. Afterwards the controller tries to find alternatively routes instead of returning immediately after the direct try.
Consequence:	Increased latency before the API call ZW_SendData returns.
Workaround:	None.
Headline/TO:	First entry point in routing can be used again if it is both most used and preferred repeater (TO #1832)
Library:	Static and bridge controller
ASIC:	200 and 300 series
Consequence:	Increase latency before reaching destination because a failing route may be used twice.
Workaround:	None.
Headline/TO:	Wakeup beam to another node can prevent a node from going into sleep mode (TO #1851)
Library:	Routing and enhanced slave configured as FLiRS
ASIC:	200 and 300 series
Detailed Description:	When a FLiRS node tries to enter sleep mode it will check if it is currently receiving a wakeup beam and if it is then it will refuse to power down. The wakeup beam check does not check the destination node ID in the beam so a beam destined for another FLiRS node can prevent the node from entering sleep mode.
Consequence:	Increased battery consumption in networks sending wakeup beams very often.
Workaround:	None

Headline/TO:	When a controller enters remove node from network it transmits a TransferPresentation frame causing other nodes in autoinclusion to answer with a node information frame (TO #1916)
Library:	All libraries
ASIC:	200 and 300 series
Detailed Description:	Transfer presentation must indicate whether an inclusion or an exclusion is processed, thereby making it possible for the node in learn mode to decide if it should react on the TransferPresentation frame.
Consequence:	Communication overhead.
Workaround:	None.
Headline/TO:	Increased 'Find nodes in range' latency (TO #1927)
Library:	All controller libraries
ASIC:	200 and 300 series
Detailed Description:	In case, the ack/routed ack is lost but destination node receives the 'find nodes in range' frame. Will the destination start neighbor discovery and at the same time will the source node try to retransmit the 'find nodes in range' frame. The retransmissions will likely not get through because the destination node is busy executing its neighbor discovery.
Consequence:	Neighbor discovery is performed but with a minor frame overhead.
Workaround:	None.
Headline/TO:	When an inclusion controller removes a node from network and notifies the SIS, the next time the removing controller asks for network updates, it gets the full network topology. (TO #1976)
Library:	All controller libraries
ASIC:	200 and 300 series
Detailed Description:	It only need full network topology when another controller made a network update prior to the removing controller request for network update.
Consequence:	Increased latency during exclusion of nodes
Workaround:	None

Headline/TO:	I/O pins pulse high after waking from sleep mode (TO #2198)
Library:	All except Static and Bridge Controller library
ASIC:	200 and 300 series
Detailed Description:	Setting an IO pin to an output and go in WUT mode then a pulse will occur on the pin when the chip wakeup gain.
Consequence:	Can trigger external hardware unintentionally.
Workaround:	In ApplicationInitHW the following code should be included before any IO initialization: P0 = P0; P1 = P1;
Headline/TO:	Virtual node indicators are not removed on new replication. (TO #2268)
Library:	Bridge controller
ASIC:	200 and 300 series
Detailed Description:	A Bridge with virtual nodes (2, 3 and 4) is put in Replication Receive without first calling ZW_SetDefault and is included into a network where nodeID 2, 3 and 4 are already exists the Bridge will then believe that node 2, 3 and 4 are virtual nodes and therefore treat them like such.
Workaround:	Always call ZW_SetDefault on Bridge prior to including a Bridge controller into a new network.
Headline/TO:	ZW_RemoveFailedNodeID do not inform application about removal of SIS/SUC in case failed node is a SIS/SUC. (TO #2454)
Library:	All controller libraries
ASIC:	200 and 300 series
Detailed Description:	When controller uses ZW_RemoveFailedNodeID to remove SIS/SUC then the function should inform the application that no SIS/SUC is available anymore. This should be done, via normal SUCID update through the APPLICATION_CONTROLLER_UPDATE functionality.
Workaround:	Ignore the callback function regarding SUC/SIS as the application already knows that it is in the process of removing a SUC/SIS.

Headline/TO:	ZW_SetLearnMode (in controllers) callback parameter bSource do not contain the new nodeID (TO #2478)
Library:	Controllers
ASIC:	200/300 Series
Detailed Description:	ZW_SetLearnMode (in controllers) callback parameter bSource do not contain the new nodeID as described in the application programmers guide but contains the nodeID on the Controller including the controller in question.
Consequence:	Controllers can not derive their own node ID from ZW_SetLearnMode() callback but should use MemoryGetID() instead.
Workaround:	None.
Headline/TO:	Routing Slave, Enhanced Slave and Enhanced 232 Slave nodes updates Assign Return Routes even if the new Response Route already is present. (TO #2814).
Library:	Routing, enhanced and enhanced 232 slaves
ASIC:	400 series
Detailed Description:	When testing if a newly received Response Route already exists in the Return Route list then it looks at a wrong place in NVM. This results in Response Routes, which already exists in the Return Route list being saved as a new Return Route.
Consequence:	Mesh network less robust
Workaround:	None.
Headline/TO:	Watchdog Reset doesn't clear MostUsed table in static controllers (TO #3093)
Library:	Static and Bridge Controllers
ASIC:	400 series
Detailed Description:	A watchdog reset does not reset the list of reset nodes on a static controller. The most used list is placed in non-zero initialized SRAM and because the SRAM stays intact after a watchdog reset the list is not cleared.
Consequence:	Watchdog reset (and ZW_SoftReset in serialAPI) doesn't clear information about most used nodes so there is some information in SRAM that survives a watchdog reset and that can potentially be dangerous because it could be that information that triggered the need for a reset.
Workaround:	None.

Headline/TO:	Source nodes and repeaters does not interpret a routed ack as silent ack in case node never heard silent ack in the first place (TO #3096)
Library:	All
ASIC:	200 and 300 series
Detailed Description:	A missing silent ack will initiate a retransmission even though a routed ack was returned because the frame arrived successfully at intended destination.
Consequence:	Additional latency in case silent ack is not heard
Workaround:	None.
Headline/TO:	CmdZWWaveGetRandom at Serial API Static controller returns 00 after running SIS stress test (TO #3317)
Library:	Controller libraries
ASIC:	200 and 300 series
Detailed Description:	After a Static controller has been used in a stress test it was reset using ZW_SetDefault and a Secure PC Controller was connected which requested some random number through the ZW_GetRandomNumber functionality, but it got a response that did not contain any random numbers.
Consequence:	No random bytes can be received from library until restart/reset/powercycle of node.
Workaround:	Execute a module reset/powercycle prior to starting a new network. Alternatively when random number API call returns no random numbers.
Headline/TO:	While changing SUC from nodeID 1 to nodeID4, nodeID 1 sends "SUC Node ID" frame to itself. (TO #3319)
Library:	Static Controller
ASIC:	200 and 300 series
Detailed Description:	A Static Controller which is SUC requests another Static Controller to be SUC and if it accepts, the old SUC transmits a "SUC Node ID" frame to itself. The SUC NodeID frame is transmitted to the old SUC to allow the network to be aware of the new SUC, but if the old SUC is the node which requested the SUC change then no such frame should of course be transmitted.
Consequence:	Communication overhead.
Workaround:	None.

Headline/TO:	If several Inclusion controllers are in NWI mode, including a new node can then result in several Nodes being added which actually do not exist (TO #3322)
Library:	Controller Libraries
ASIC:	200 and 300 series
Detailed Description:	When the include emits its NodeInformation frame it receives several "ASSIGN ID" frames directed at it (using the HomeID of the include), but instead of only "ACK"ing ONE of the "ASSIGN ID" frames it "ACK"s several prior to actually being included and from there on just ignoring communication from all other controller but ONE. The controllers receiving an "ACK" as answer to their "ASSIGN ID" transmission regards the node as being included even if the node do not answer to the following "NOP" frames.
Consequence:	None existing nodes gets included into network.
Workaround:	Do not start more than ONE controller in NWI mode.
Headline/TO:	SendDataAbort is not fully implemented in routing/enhanced/enhanced_232 slave libraries (TO #3323)
Library:	Slave Routing, Slave Enhanced and Slave Enhanced 232
ASIC:	200 and 300 series
Detailed Description:	When an Application uses ZW_SendData functionality in either of its incarnations the frame is marked as being initiated by an application and therefore can be aborted by calling ZW_SendDataAbort. This functionality is for Routing/Enhanced/Enhanced_232 slave based Applications only functional when the Application uses ZW_SendDataMulti.
Consequence:	Communication overhead.
Workaround:	None.
Headline/TO:	Controllers can use Last Working Route with non-existing repeaters (TO #3326)
Library:	All controllers
ASIC:	200 and 300 series
Consequence:	Controllers can use a Last Working Route (LWR) that contains a repeater that has been removed from the network. This will cause the LWR to fail and be deleted. So one extra route will be tried in this situation.
Workaround:	None.

7 SAMPLE CODE FOR ZW0201/ZW0301 EMBEDDED APPLICATIONS

The Developer's Kit contains sample code as well as compiled code for the sample applications. The sample code can be used as it is or changed according to the needs of the application programmer.

Be aware of that the callback function is not used throughout in the sample code in all ZW_SendData API calls. Using the callback function will prevent the transmit queue from overflowing. (TO #551).

7.1 Binary Sensor Sample Code

The Binary Sensor v3.49 sample code contains an example of a routing slave application. The binary sensor contains a button that when pressed will toggle ON/OFF up to 5 different slave nodes based on the Multilevel Switch command class for a certain time and dim level before switching them off. The details can be found in [4], which is included on the Developer's Kit CD. The source code is now split into a number of separate files to allow code reuse by other sample applications.

Known defects

None.

7.1 Secure Binary Sensor Sample Code

The Secure Binary Sensor v3.50 sample code contains an example of a routing slave application. The binary sensor contains a button that when pressed will toggle ON/OFF up to 5 different slave nodes based on the Multilevel Switch command class for a certain time and dim level before switching them off. The supported/controlled command classes are embedded in the Security command class. The details can be found in [4], which is included on the Developer's Kit CD.

NOTICE: Binaries and AES128 encryption/decryption engine are not distributed on the Developer's Kit CD due to export restrictions. Contact support via zensys_support@sigmadesigns.com for further information.

Known defects

- The 10 seconds Nonce Request timeout is too short for security enable applications. It should be 20 seconds. It can be fixed as follows; The security timeout is checked in functions StartSecuritySendTimeOut(BYTE timeOut) and StartSecurityTimeOut(BYTE timeOut) in file ZW_Security_AES_module.c. Change from 10 seconds to 20 seconds in above functions. (TO #3273)

7.2 Battery Operated Binary Sensor Sample Code

The Battery Operated Binary Sensor v3.49 sample code contains an example of a battery operated routing slave application with power down capabilities. The battery operate binary sensor contains a button that when pressed will toggle ON/OFF on for example an LED dimmer. The details can be found in [4], which is included on the Developer's Kit CD.

Known defects

- Battery Operated Binary Sensor that stops getting Wakeup No More Information commands from controller also stops sending commands to associated nodes when activating push button (TO #3099)

7.3 Secure Battery Operated Binary Sensor Sample Code

The Secure Battery Operated Binary Sensor v3.49 sample code contains an example of a battery operated routing slave application with power down capabilities. The battery operate binary sensor contains a button that when pressed will toggle ON/OFF on for example an LED dimmer. The supported/controlled command classes are embedded in the Security command class. The details can be found in [4], which is included on the Developer's Kit CD.

NOTICE: Binaries and AES128 encryption/decryption engine are not included on the Developer's Kit CD due to export restrictions. Contact support via zensys_support@sigmadesigns.com for further information.

Known defects

- Battery Operated Binary Sensor that stops getting Wakeup No More Information commands from controller also stops sending commands to associated nodes when activating push button (TO #3099)
- The 10 seconds Nonce Request timeout is too short for security enable applications. It should be 20 seconds. It can be fixed as follows; The security timeout is checked in functions StartSecuritySendTimeOut(BYTE timeOut) and StartSecurityTimeOut(BYTE timeOut) in file ZW_Security_AES_module.c. Change from 10 seconds to 20 seconds in above functions. (TO #3273)

7.4 Development Controller Sample Code

The Development Controller v3.47 sample code contains an example of a portable controller, which can be used to include/exclude nodes and control the slave sample applications included in the Developer's Kit. The details can be found in [4] and [7], which are included on the Developer's Kit CD. Be aware that the Development Controller will not be powered down when inactive, so remember to disconnect in case the battery pack is used as power supply. Notice also the different jumper settings on the Development Controller Unit depending on the single chip used.

Known defects

- S1 button does not work on a ZDP03A based Development Controller (TO #2901)

7.5 Secure Development Controller Sample Code

The Secure Development Controller v1.27 sample code contains an example of a portable controller, which can be used to include/exclude nodes and control the slave sample applications included in the Developer's Kit. The sample application uses an AVR ATmega128 as host on a ZDP02A Development module in combination with a Z-Wave module on the ZDP02A Development module hosting a serial API based portable controller sample application. The supported/controlled command classes are embedded in the Security command class. For further details refer to [4] and [17], which are included on the Developer's Kit CD.

NOTICE: Binaries and AES128 encryption/decryption engine are not included on the Developer's Kit CD due to export restrictions. Contact support via zensys_support@sigmadesigns.com for further information.

Known defects

- The 10 seconds Nonce Request timeout is too short for security enable applications. It should be 20 seconds. It can be fixed as follows; The security timeout is checked in functions StartSecuritySendTimeOut(BYTE timeOut) and StartSecurityTimeOut(BYTE timeOut) in file ZW_Security_AES_module.c. Change from 10 seconds to 20 seconds in above functions. (TO #3273)

7.6 Door Bell Sample Code

The Door Bell v1.65 sample code contains an example of how a door bell application can be implemented using a Development Controller application as push button and a frequently listening routing slave as chime. Detailed information regarding the Door Bell sample code can be found in [4], which is included on the Developer's Kit CD.

Known defects

- Should ignore Binary Switch Set command 0x64 including other values outside legal range. (TO #3349)

7.7 Door Lock Sample Code

The Door Lock v1.42 sample code contains an example of how a door lock application can be implemented using a Development Controller application as push button and a frequently listening routing slave as door lock. Detailed information regarding the Door Lock sample code can be found in [4], which is included on the Developer's Kit CD.

NOTICE: Binaries and AES128 encryption/decryption engine are not included on the Developer's Kit CD due to export restrictions. Contact support via zensys_support@sigmadesigns.com for further information.

Known defects

- Door Lock does not store Powerlevel Set command data in non-volatile memory (TO #2467)
- Door Lock cannot wakeup when sending a wakeup beam (TO #2928)
- Door Lock does not support command class Door Lock Command Class. Report is not returned when issuing a Door Lock Operation Get and Door Lock Configuration Get (TO #3316)

7.8 Secure Door Lock Sample Code

The Secure Door Lock v1.43 sample code contains an example of how a door lock application can be implemented using a Secure Development Controller application as push button and a frequently listening routing slave as door lock. The supported/controlled command classes are embedded in the Security command class. Detailed information regarding the Door Lock sample code can be found in [4], which is included on the Developer's Kit CD.

NOTICE: Binaries and AES128 encryption/decryption engine are not distributed on the Developer's Kit CD due to export restrictions. Contact support via zensys_support@sigmadesigns.com for further information.

Known defects

- Door Lock cannot wakeup when sending a wakeup beam (TO #2928)
- The 10 seconds Nonce Request timeout is too short for security enable applications. It should be 20 seconds. It can be fixed as follows; The security timeout is checked in functions StartSecuritySendTimeOut(BYTE timeOut) and StartSecurityTimeOut(BYTE timeOut) in file ZW_Security_AES_module.c. Change from 10 seconds to 20 seconds in above functions. (TO #3273)
- Door Lock does not support command class Door Lock Command Class. Report is not returned when issuing a Door Lock Operation Get and Door Lock Configuration Get (TO #3316)

7.9 LED Dimmer Sample Code

The LED Dimmer v3.49 sample code contains an example of a routing slave application. Detailed information regarding the LED dimmer sample code can be found in [4], which is included on the Developer's Kit CD.

Known defects

None

7.10 Secure LED Dimmer Sample Code

The Secure LED Dimmer v3.49 sample code contains an example of a slave application. The supported/controlled command classes are embedded in the Security command class. For detailed information regarding the LED dimmer sample code refer to [4], which is included on the Developer's Kit CD.

NOTICE: Binaries and AES128 encryption/decryption engine are not included on the Developer's Kit CD due to export restrictions. Contact support via zensys_support@sigmadesigns.com for further information.

Known defects

- The 10 seconds Nonce Request timeout is too short for security enable applications. It should be 20 seconds. It can be fixed as follows; The security timeout is checked in functions StartSecuritySendTimeOut(BYTE timeOut) and StartSecurityTimeOut(BYTE timeOut) in file ZW_Security_AES_module.c. Change from 10 seconds to 20 seconds in above functions. (TO #3273)

7.11 My Product Sample Code

As an alternative to modifying the ZW0201/ZW0301 sample code applications the My Product v1.53 can be used to build a slave application. The My Product contains the minimum framework to begin developing a slave application.

Known defects

None

7.12 Production Test DUT Sample Code

The Production Test DUT v1.54 sample 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 sample code can be found in [4], which is included on the Developer's Kit CD.

Known problems

None

7.13 Production Test Generator Sample Code

The Production Test Generator v1.68 sample 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 sample code can be found in [4], which is included on the Developer's Kit CD.

Known problems

None

7.14 Serial API Sample Code

The Serial API v3.61 sample code contains an example of how a serial UART interface to the Z-Wave protocol can be implemented. The Serial API support both PC based controller and slave applications. Notice that some libraries require reductions in the serial API calls to be able to reside on the Z-Wave module. For detailed information about the Serial API sample code and how to interface to the Serial API refer to [4], which are included on the Developer's Kit CD.

Known problems

- Missing serial API callback when requesting Node Information Frame - ZW_RequestNodeInfo. (TO #2358)
- uVision project use wrong bridge library name. (TO #2509)

7.15 Utility Functions

Helpful functions used by several embedded sample applications v1.67. New files added to support controller learn mode and security.

Known defects

- Security sample code does not wait for `nonce_get` finish. When a node sends a routed `nonce_get` frame then node could potentially receive a `nonce_report` while waiting on the `nonce_get` callback. If this happens then the security sample code will put the encrypted message in the tx queue. This can lead to that the encrypted message repeaters can be corrupted (TO #3329)

8 SAMPLE CODE FOR PC LIBRARIES ETC

The PC applications are based on a number of libraries described in the following sections. The libraries are implemented in C# using Microsoft Visual Studio 2005. The .NET Framework version 2.0 redistributable package (found on Microsoft Windows update site) must be installed to run applications using the .NET Framework.

8.1 Z-Wave DLL

The Z-Wave DLL v4.43 is a framework that simplifies the development of Z-Wave enabled Windows Forms or Console applications for Microsoft .NET Framework platform (Windows XP and Windows Vista applications). It is a high-level interface to the serial API interface on the ZW0102/ZW0201/ZW0301 based modules. For detailed information refer to [13], which is included on the Developer's Kit CD.

Known defects

None

8.2 Z-Wave HAL

The Z-Wave High-level Application Layer v4.40 is a High Level Application Layer in terms of Z-Wave Dll architecture. It contains common functions that are used in Z-Wave enabled PC applications: ZWavePCController, ZWaveProgrammer, ZWaveUPnPBridge etc. For detailed information refer to [13], which is included on the Developer's Kit CD.

Known defects

None

8.3 Z-Wave Security HAL

The Z-Wave Security High-level Application Layer v4.39 is a High Level Application Layer in terms of Z-Wave Dll architecture. For detailed information refer to [13], which is included on the Developer's Kit CD.

Known defects

None

8.4 Z-Wave Command Classes

The Z-Wave Command Classes v1.14 implement a XML parser, which enables parsing of Z-Wave frames by the Ziffer and creation of Z-Wave commands by the PC Controller.

Known defects

None

8.5 WinFormsUI

The WinFormsUI v1.00 implements the windows docking library used by the PC applications.

Known defects

None

8.6 ZensysFramework

The WinFormsUI v1.05 implements the additional functions, formatters, helpers. used by the PC applications:

Known defects

None

8.7 ZensysFrameworkUI

The WinFormsUI v1.13 implements Z-Wave UI elements that can be reused by the PC applications:

- Associations View Control;
- Bridged UPnP Device View Control;
- Controller View Control;
- Node View Control;
- UPnP Binary Light Device View Control;
- UPnP Device Scanner View Control;
- UPnP Media Renderer View Control.

Known defects

None

8.8 ZensysFrameworkUIControls

The WinFormsUI v1.07 implements Z-Wave additional UI elements that can be reused by the PC applications:

- ListDataView;
- TreeDataView;
- BitBox;
- ThreadSafeLabel.

Known defects

None

9 SAMPLE CODE FOR PC APPLICATIONS

9.1 PC Controller Sample Code

The PC based Controller v4.51 sample code contains an example of how to include, exclude and control the devices included in the Developer's Kit. System information can be replicated to and from the other controllers. The PC based Controller application is implemented in C# using Microsoft Visual Studio 2005. The .NET Framework version 2.0 re-distributable package (found on Microsoft Windows update site) must be installed to run applications using the .NET Framework. For detailed information refer to [9], which is included on the Developer's Kit CD.

Known defects

- PC Controller crash under heavy load after some time when polling a lot of devices and receiving unsolicited frames (TO #3158)
- Switch All On/Off Command works incorrect in Secure PC Controller (TO #3335)

9.2 Z-Wave Bridge Sample Code

The PC based Z-Wave Bridge v3.27 sample code contains an example of a Z-Wave/UPnP bridge. As an example it is possible to bridge between the UPnP device BinaryLight and a Z-Wave Power Switch. An UPnP Renderer can also be controlled as a Z-Wave Power Switch to play audio/video. The Z-Wave Bridge application is implemented in C# using Microsoft Visual Studio 2005. The .NET Framework version 2.0 re-distributable package (found on Microsoft Windows update site) must be installed to run applications using the .NET Framework. The Z-Wave Bridge application communicates with the Serial API on the Z-Wave module via the COM port. For detailed information refer to [14], which is included on the Developer's Kit CD.

Known defects

None

9.3 PC Installer Tool Sample Code

The PC based Installer Tool v2.27 sample code contains an example of how to implement the special installation features in the Z-Wave protocol. The features support the installer during installation and maintenance of the Z-Wave network. The PC based Installer Tool application is implemented in C# using Microsoft Visual Studio 2005. The .NET Framework version 2.0 re-distributable package (found on Microsoft Windows update site) must be installed to run applications using the .NET Framework. For detailed information refer to [10], which is included on the Developer's Kit CD.

Known defects

- Announce itself as listening device but should be non-listening (TO #3314)

10 TOOLS

The Developer's Kit contains various PC based tools for helping SW developers writing and debugging code.

10.1 Zniffer

The Z-Wave Zniffer v4.20 tool enables the Z-Wave developers to analyze unsecure/secure and 9.6/40/100kbps RF communication between Z-Wave nodes. The tool requires a Zniffer 400 Series firmware v2.28 or 300 Series firmware v2.10 downloaded to the Z-Wave module to enable logging of the RF communication. Be aware that the range of the Zniffer module may prevent logging of the entire network from one location. Zniffer supports Windows XP/2003/Vista(32/64bit)/7(32/64bit). For detailed information refer to [5] included on the Developer's Kit CD.

Known defects

- Zniffer can crash after long time having filtering enabled (TO #3188)
- Zniffer can crash when selecting millions of lines (TO #3191)

10.2 Zniffer File Converter

The Z-Wave Zniffer FileConverter v1.05 introduced to open old *znf files. Zniffer log file format (*.zlf) changed to support Windows 7 etc.

Known defects

None

10.3 XML Editor

The XML Editor v1.10 tool is used to define approved Z-Wave devices and command classes [12] used by the application layer of the Z-Wave protocol in the XML document that can be used by the Z-Wave Zniffer [5] for interpretation of the above mentioned devices and command classes.

Beside a XML file containing all the information is it also possible to generate a C# class file and C header file as foundation for application development. The Z-Wave XML Editor enables also the customer to define devices and command classes under development or proprietary command class structures.

For detailed information refer to [11] included on the Developer's Kit CD.

Known defects

- XML Editor does not remember "show hexadecimal" property on value (TO #2232)
- While navigating the listview with command classes and commands XML Editor crashes (TO #2233)
- Does not save all changes correctly (TO #2234)
- Does not rename parameters / generate cs code correctly (TO #2235)

10.4 PVT and RF Regulatory

The PVT & RF regulatory hex files v1.10 enables the developer to conduct verification testing and regulatory measurements on the Z-Wave enabled products. More details can be found in [15] included on the Developer's Kit CD.

New features

India (IN) frequency supported.

Fixed defects since last release

None

Known defects

None

10.5 Enhanced Reliability Test Tool

The Enhanced Reliability Test Tool v1.06 enables Z-Wave developers to test the RF reliability of Z-Wave products by sending Basic Set command frames and analyzing the acknowledge frames. The program can be configured to disable the Z-Wave protocol frame retransmission mechanism. The Enhanced Reliability Test Tool requires a special firmware version 2.52 of the Serial API based static controller where retransmission of frames can be disabled/enabled. More details can be found in [15] included on the Developer's Kit CD.

New features

The following features are added to the new ERTT:

- Possible to include the DUT directly to the ERTT and via a portable controller.
- Notification on GUI when a transmission error occurs.
- A field showing how many frames sent and how many frames received.
- A log file reporting which nodes have been failing during test and how many frames have been sent / received.
- Possible to reset ERTT.
- Possible to transmit frames with or without retransmission.
- Possible to transmit frames with or without low power.
- Possible to send frames as Switch On or Switch Off or toggling between Switch On and Switch Off.
- Possible to Transmit frames controlled by the ERTT module.
- Possible to use Send Data Meta functionality.
- Show only listening nodes in listview display.
- Ported to .NET 2.0

Fixed defects since last release

- The ERTT Controller now refuses the request to become a SUC. (TO #655)
- A primary Development Controller is typically used to add the nodes to test. Afterwards a controller replication is done from the Development Controller to the ERTT controller. In case the Development Controller is based on Dev. Kit 3.40+ then it will appear in the list of nodes on the ERTT controller because it have node ID = 0x01. This creates a problem because the ERTT controller will send frames to the Development Controller when the test is started. To avoid this problem use a Development Controller based on Dev. Kit 3.31 because then the node ID = 0xEF and is therefore not included on the list. Now it is possible to select which nodes to send to in the list. (TO #808)
- Removed deadlock situation after longer continuous operation. (TO #1096)
- Continue sending to all selected nodes. (TO #1098)

Known defects

- ERTT continues to use retransmissions after enabled once even though it is switched off again (ERTT starts without retransmissions as default). (TO #1566)

10.6 Z-Wave Programmer

The Z-Wave Programmer v2.42 software is necessary for programming the flash on the Z-Wave 100/200/300/400 Series Single Chips during SW and HW development and for small production series. The Z-Wave Programmer software uses the ZDP03A Development Platform configured with ATmega128 firmware v1.16 (both source code and hex files included). The Z-Wave Programmer supports also initialization of the external EEPROM including home ID on the Z-Wave modules. The Z-Wave Programmer can also configure transmission power, lock bits and RF settings on the Z-Wave modules.

Programmer supports Windows XP/2003/Vista(32/64bit)/7(32/64bit). For detailed information refer to [16] included on the Developer's Kit CD.

Known defects

None

10.7 SD3402 Crystal Calibration

The SD3402 crystal calibration firmware v1.19 used by the calibration box, refer to **Error! Reference source not found.** ZM4101 and ZM4102 are already calibrated during production. Hex file is located in Programmer folder.

Known defects

None

10.8 uVision3 IDE

The Keil uVision3 IDE used for developing Z-Wave applications as an alternative to the makefile system. How to configure uVision3 can be found in [20] included on the Developer's Kit CD.

WARNING: Configure uVision3 project as the makefile to obtain the same binary files. All options, order of compilation etc. must be the same.

New features

None

Fixed problems since last release

None

Known problems

None

REFERENCES

- [1] Sigma Designs, DSH10717, Datasheet, Datasheet for ZW0301 Z-Wave Single Chip.
- [2] Sigma Designs, INS10244, Instruction, Z-Wave Node Type Overview and Network Installation Guide.
- [3] Sigma Designs, APL10748, Application Note, Porting Z-Wave Appl. SW from ZW0201 to ZW0301.
- [4] Sigma Designs, INS11095, Instruction, Z-Wave ZW0201/ZW0301 Appl. Prg. Guide v4.53.00.
- [5] Sigma Designs, INS10249, Instruction, Z-Wave Zniiffer User Guide.
- [6] Sigma Designs, INS10309, Instruction, Using the Epsilon5 for programming the Z-Wave Single Chips.
- [7] Sigma Designs, INS10236, Instruction, Development Controller User Guide.
- [8] Sigma Designs, APL10248, Application Note, Porting Z-Wave Appl. SW from ZW0102 to ZW0201.
- [9] Sigma Designs, INS10240, Instruction, PC Based Controller User Guide.
- [10] Sigma Designs, INS10241, Instruction, PC Installer Tool Application User Guide.
- [11] Sigma Designs, INS10680, Instruction, Z-Wave XML Editor User Guide.
- [12] Sigma Designs, SDS10242, Software Design Specification, Z-Wave Device Class Specification.
- [13] Sigma Designs, INS10250, Instruction, Z-Wave DLL User Guide.
- [14] Sigma Designs, INS10245, Instruction, Z-Wave Bridge User Guide.
- [15] Sigma Designs, INS10336, Instruction, Z-Wave Reliability Test Guideline.
- [16] Sigma Designs, INS10679, Instruction, Z-Wave Programmer User Guide.
- [17] Sigma Designs, INS10681, Instruction, Secure Development Controller (ATmega) User Guide.
- [18] Sigma Designs, SDS10865, Software Design Specification, Z-Wave Security Application Layer.
- [19] Sigma Designs, APL10742, Application Note, ZM3102N with External PA and Switch.
- [20] Sigma Designs, INS10246, Instruction, Keil uVision3 IDE User Guide.
- [21] Sigma Designs, SDS11060, Software Design Specification, Z-Wave Command Class Specification.
- [22] Sigma Designs, INS11072, Instruction, Z-Wave Programmer Communication Protocol.

INDEX

A

Application memory	15
ApplicationCommandHandler	12
ApplicationSlaveCommandHandler	12
AVR ATmega128	32

C

Calibration	43
Crystal calibration	43

D

Duplicated nodes	12
------------------------	----

E

External EEPROM	43
-----------------------	----

F

FileConverter	40
FLiRS	13

I

India frequency	9
-----------------------	---

K

Keil PK51 v7.50	3
Keil PK51 v9.00	3

L

Lock bits	43
-----------------	----

M

Malaysia frequency	12
Memory footprint	86
Multicast support	12

N

Non-volatile memory write cycles	15
--	----

P

PVT & RF regulatory hex files	41
-------------------------------------	----

R

Random Home ID	11
Route recovery	10

S

Secure applications	12
---------------------------	----

T

TRANSMIT_OPTION_EXPLORE	10
-------------------------------	----

X

XML Editor	40
------------------	----

Z

ZDP02A Development module	32
ZW_AssignSUCNodeID	12
ZW_EnableSUC	12
ZW_GetRandomWord	12
ZW_GetSUCNodeID.....	12
ZW_LockRoute	10
ZW_RediscoveryNeeded.....	14
ZW_RequestNetWorkUpdate	14
ZW_SetRoutingMAX.....	10
Z-Wave Programmer	43
Z-Wave Ziffer	40
Z-Wave Ziffer FileConverter	40

APPENDIX A FIXED DEFECTS IN 4.52.00/01

The following defects are fixed compared to Developer's Kit v4.51:

Headline/TO:	Replication receive does not timeout in case the "Replication End" command is not received. This could happen if e.g. the including controller losses power during replication or brought outside of direct range or in situations with poor communication between the controllers in question. (TO #458)
Library:	All controllers
ASIC:	200 and 300 series
Consequence:	Replication process will hang requiring a reset of the receiving controller to recover
Headline/TO:	Static controller fails to route response to a portable controller requesting a routed network topology update – ZW_RequestNetWorkUpdate. This scenario happens when the response route allocated for the network topology update is overwritten. This happen when two different nodes access the static controller during the update. Then will the second node overwrite the response routes used for the network topology update which force the static controller to try direct to the portable controller due to lack of a response route. Direct is used because the static controller cannot calculate a route to a portable controller. (TO #1275)
Library:	Static and bridge controllers.
ASIC:	200 and 300 series
Consequence:	Overwriting the response route allocated for the network topology update will the remaining updates not be received by the portable controller. They will still have status as pending on the static controller so they will be transferred next time the portable controller request an update.

Headline/TO:	<p>A 200 series based device will have reduced sensitivity when placed in close proximity (less than 1ft/30cm) to a 100 series based device.</p> <p>This TO is a continuation of TO #965 that was discovered and communicated to Partners in Aug/Sept 2005. A fix was introduced in 4.02 and involved changing the RF configuration of the 200 series to use the opposite sideband. This reduced the coexistence issue from almost complete jamming to a noticeable reduction in range, which was however still above the range required for Z-Wave Certification (60ft/20m).</p> <p>The reduced range means that routing will take place in more situations, which for some latency critical situations is undesirable. (TO #1288)</p>
Library:	All 200 and 300 Series libraries
ASIC:	200 and 300 Series
Consequence:	<p>Since the co-location issue is only likely to happen in systems with multiple nodes, this also means that there will be a good repeater backbone available to reach the node via routing instead of direct.</p> <p>End-users are therefore unlikely to notice the reduced range, but may notice an increased latency when controlling a large number of nodes, due to the increased routing taking place in the network.</p> <p>Our measurements have shown the highest reduction in range of a 200 series based device when placed in-between two 100 series based products in a triple gang configuration. In dual gang configuration and with increased distance the reduction in sensitivity decreases.</p> <p>The range reduction drops to around 20% or less when the 200 series is moved 1ft/30cm away from the 100 series based device. This makes the range of the 200 series comparable to that of a 100 series based product. When moved further away (>3ft/1m) only very limited reduction in sensitivity can be measured, and the 200 series then has better performance than a 100 series based device.</p> <p>The level of LO leakage from the 100 series device also affect the results, with the US having a 10dB higher limit under FCC than what the RTT&E allows in Europe. Reducing the LO leakage from the 100 series based devices will therefore reduce the potential range reduction in the field of the 200 series based devices placed next to them.</p>
Workaround:	None.
Headline/TO:	Controller tries occasionally to route twice through repeater even though it was unsuccessful in the first trail. (TO #1520)
Library:	Static controller, bridge controller and both reduced static controller libraries.
ASIC:	200 and 300 series
Consequence:	Increased latency due to repeated routing attempts.
Workaround:	None.

Headline/TO:	Controllers sending a multicast immediately after a broadcast will always use 9.6kbps despite all the destination nodes involved support 40kbps. (TO #1529)
Library:	All controller libraries.
ASIC:	200 and 300 Series
Consequence:	Increased latency due to using 9.6kbps when 40kbps is possible. Accompanying single casts to the multicast will all be send using 40kbps.
Workaround:	None.
Headline/TO:	Delete slave node in learn mode does not provide correct callback (TO# 2778)
Library:	All
ASIC:	200/300 Series
Detailed Description:	When a deleted slave node is in learn mode the application doesn't get a callback if a controller tries to delete the node again.
Consequence:	An already deleted node will always time out of learn mode even if the controller try is to delete it.
Headline/TO:	Controllers cannot include a virtual node in a controller bridge from devkits prior to 4.51. (TO #2794).
Library:	Controller portable, static and installer
ASIC:	200/300 series
Detailed Description:	The inclusion procedure would fail when trying to include a virtual node in a bridge controller from devkits prior to 4.51 to a devkit 6.0 controller. This was caused by the old bridge controllers use of homeid 0x00000000 during inclusion.
Consequence:	Virtual nodes from old bridge controllers could not be included in networks controlled by SDK 4.51 controllers.

Headline/TO:	RemoveNode Stop does not work (TO #2803).
Library:	All controllers
ASIC:	200/300 series
Detailed Description:	RemoveNode Stop does not stop Remove Node state after putting a controller in RemoveNode.
Consequence:	RemoveNode continues despite instructed to stop.
Headline/TO:	Missing dimming duration in Switch Multilevel Start Level Change V2 Command Class (TO#2845)
Library:	ZW_classcmd.h
ASIC:	200/300 series
Detailed Description:	Dimming duration field was missing in definition of the Switch Multilevel Start Level Change V2 definition
Consequence:	Command structure defined in ZW_classcmd.h could not be used in the application.
Resolution:	ZW_classcmd.h fixed
Headline/TO:	Node do not ACK retransmissions of routed ACK/ERR frames (TO# 2903)
Library:	All
ASIC:	200/300 Series
Detailed Description:	Controller and Slave nodes ignore any retransmissions of a Routed ACK/ERR frame, which it (the destination node) already has acknowledged in response to a prior incarnation of the same Routed ACK/ERR frame.
Consequence:	If the last repeater do not hear the ACK from the source node then the last repeater will retransmit the Routed ACK/ERR frame twice resulting in increased latency and an increased chance for collisions and following unsuccessful frame transactions.

Headline/TO:	Nodes receiving an Explore frame do not always use the explored route as Response Route/Last Working Route. (TO# 2905)
Library:	All
ASIC:	200/300 Series
Detailed Description:	Nodes receiving an Explore frame do not always use the explored route as Response Route/Last Working Route. Instead a direct range route is stored as Response/Last Working route in the destination node.
Consequence:	In a polling situation, where a node makes a periodically "Get" to a node and this answers with a "Report" could in the worst case end in a situation where every "Get" and "Report" transmission is resolved by an Explorer frame causing increased latency.
Headline/TO:	Frames are send to node 0 when retransmitted (TO# 2909)
Library:	Routing slave/Enhanced Slave
ASIC:	200/300 Series
Detailed Description:	A routing and enhanced slaves can send a frame to a random node id (typically node id 0) if the node the frame is send to doesn't acknowledge the first 3 direct attempts and there is no return route configured to that destination node.
Consequence:	Added latency when communication with the destination node fails.
Headline/TO:	Application timer is not activated as supposed to when calling ZW_PWMSSetup() (TO #2925)
Library:	All
ASIC:	200/300 Series
Detailed Description:	Application timer is not activated as supposed to when calling ZW_PWMSSetup().This happens because a wrong register is read to check if we should activate the timer.
Consequence:	Controllers can not derive their own node ID from ZW_SetLearnMode() callback but should use MemoryGetID() instead.

Headline/TO:	Controller node which are in the process of including/excluding a node, calls ApplicationCommandHandler with frames received with foreign HomeId but directed at the Controllers NodeId (TO #2926)
Library:	All controllers
ASIC:	200/300 Series
Detailed Description:	If a Controller node (in the process of including/excluding a node) receives a frame transmitted on foreign homeid but with the Controllers nodeid, it handles the frame as it was the destination and calls ApplicationCommandHandler if frame is an Application frame.
Consequence:	Try to process application command from another network.
Headline/TO:	Secondary controller sends a frame with wrong communication speed (TO# 2937)
Library:	All controllers
ASIC:	200/300 Series
Detailed Description:	We add a slave node to the Primary controller and then replicate to a secondary controller. When the secondary controller sends a Basic Set to the slave node, will the transmitted frame use 9.6kbps instead of 40 kbps.
Consequence:	Communication latency.
Headline/TO:	4.2x based inclusion controller cannot include nodes when using a 4.51 based SIS (TO# 2938)
Library:	Static and bridge controller
ASIC:	200/300 Series
Detailed Description:	The 4.2x based inclusion controller receives a transfer end frame with an unknown status from the SIS when it does the controller update before trying to include a new node to the network.
Consequence:	4.2x based inclusion controller cannot include nodes.

Headline/TO:	In case direct communication fails then many legal routes will postpone trying a direct again (TO# 2947)
Library:	Static and bridge controller
ASIC:	200/300 Series
Detailed Description:	If a static controller fails a direct transmission to a neighboring node it will try to find a route to the wanted destination. It will continue use this and not try direct before all routes (according to routing scheme) have failed.
Consequence:	Communication latency.
Resolution:	If last working route exist try it. In case it fails, purge last working route and try direct if neighbor.
Headline/TO:	Enhanced 232 Slave do not keep the Cached list of node IDs, which have been given via Return Routes from a Controller (TO #2949)
Library:	Enhanced 232 slave
ASIC:	300 Series
Consequence:	ZW_RequestNetworkUpdate can fetch wrong node IDs from list.
Headline/TO:	When slaves get a new response route from a node that already is present in the assigned return routes then it saves the response route in the return route structure but destroys unfortunately speed information (TO #2953)
Library:	Routing slave, Enhanced slave and Enhanced 232 slave
ASIC:	300 Series
Detailed Description:	Return routes saved as 40kbps routes is now noted as 9.6kbps routes in the return route structure.
Consequence:	Increase latency due to degradation of speed.

APPENDIX B FIXED DEFECTS IN 4.51

The following defects are fixed compared to Developer's Kit v4.50 (Beta1) Patch1:

Headline/TO:	Static Controller use routes to reply direct communication (TO #1655)
Library:	Static, Bridge and Reduced Static Controller
ASIC:	100, 200 and 300 series
Detailed Description:	A node has been included in the network and do not have direct range to the Static Controller (SC). When the node is moved within direct and initiate direct communication to the SC, the SC use routes for the reply.
Consequence:	Increased latency and in case the node is moved to a position where the Static Controller has no known routes to reach it the reply will fail
Resolution:	Static Controller initiate rediscovery process of the node
Headline/TO:	ZW_rf_tf.h file distributed despite it is obsolete (TO #1675)
Library:	All libraries
ASIC:	200 and 300 series
Detailed Description:	A reference to file requires its presence.
Consequence:	None but do not erase file.
Resolution:	Removed from reference file and ZW_rf_tf.h file not distributed anymore.
Headline/TO:	Controllers may change node ID to 0xEF during inclusion in case another node (e.g. SUC) requests node information frame (NIF) from it while doing a neighbor search (TO #1840).
Library:	All controller libraries
ASIC:	200 and 300 series
Consequence:	Inclusion fails.
Resolution:	Ignore other nodes when conducting neighbor search.

Headline/TO:	Assign return routes sets wrong speed in pure 40kbps networks (TO #1841).
Library:	All controller libraries
ASIC:	200 and 300 series
Detailed Description:	The first assigned return route use always 40kbps direct or via repeaters. The remaining return routes use 9.6kbps.
Consequence:	Increased latency
Resolution:	Sets assigned return route speed correct.
Headline/TO:	ZW_RequestNodeNeighborUpdate() reports success on empty range info (TO #1953)
Library:	All controller libraries
ASIC:	200 and 300 series
Detailed Description:	ZW_RequestNodeNeighborUpdate() returns ok if a range info frame with no neighbors is received from the rediscovered node. The call should fail and the range info should be discarded not stored in the routing table.
Consequence:	Node performing node neighbour update result in having no neighbours in requesting controller.
Resolution:	Returns failure in case range info is empty.
Headline/TO:	Controller repeatedly requests network wide inclusion (NWI) inclusion via routing (TO #1964)
Library:	All controller libraries
ASIC:	200 and 300 series
Detailed Description:	The controller node requesting NWI inclusion was assigned a node ID and received the FindNodesInRange message. For unknown reasons, the node did not start sending out NOPpower messages to the indicated neighbors. After some time the node would give up and start requesting inclusion again. The NWI Master controller also aborted the inclusion session and started serving new inclusion requests.
Consequence:	Controller fails to get included via the NWI mechanism
Resolution:	Use same 40kbps speed throughout to avoid asymmetric links.

Headline/TO:	Sends explorer frame to a FLiRS node in case all assigned return routes fails. (TO #1999)
Library:	Routing and Enhanced Slave libraries
ASIC:	200 and 300 series
Detailed Description:	The routing and enhanced slave does not check whether it is a FLiRS node before issuing an explore frame.
Consequence:	Increased traffic when addressing a non-working FLiRS node.
Resolution:	Suppress explore frames if destination is a known FLiRS node.
Headline/TO:	Issue setting PWM output high. (TO #2003)
Library:	All libraries
ASIC:	200 and 300 series
Detailed Description:	The PWM hardware will lock if the function ZW_PWM_PRESCALE() is called with the a zero value in the 2nd parameter, e.g. ZW_PWM_PRESCALE(x,0).
Consequence:	Not possible to set PWM output constant high using the ZW_PWM_PRESCALE() function.
Resolution:	PWM output setting fix integrated into protocol API.
Headline/TO:	Return value from ZW_RemoveNodeFromNetwork missing (TO #2047)
Library:	All controller libraries
ASIC:	200 and 300 series
Detailed Description:	When calling ZW_RemoveNodeFromNetwork using REMOVE_NODE_STOP returns no callback to application.
Consequence:	Increased latency during exclusion of nodes
Resolution:	Use NULL pointer in parameter completedFunc when calling ZW_RemoveNodeFromNetwork using REMOVE_NODE_STOP to terminate function correctly.

Headline/TO:	Explorer frames should use reduced power. (TO #2048)
Library:	All libraries
ASIC:	200 and 300 series
Consequence:	Increase robustness of links by avoiding asymmetric links.
Resolution:	Explorer frame now uses the same reduced power as find neighbor call. Changing ZW_RFPowerlevelRediscoverySet settings will reflect the following explore frame transmissions.
Headline/TO:	Too many attempts before a successfully inclusion is accomplished (TO #2066)
Library:	All controller libraries
ASIC:	200 and 300 series
Detailed Description:	Many inclusion attempts fail in rare cases due to the AssignID and all other NWI Master messages never reaching the node seeking inclusion.
Consequence:	Increased latency during exclusion of nodes
Resolution:	Explorer frame now uses the same reduced power as find neighbor call.
Headline/TO:	Speed info saved together with response route / last working route not always used in all possible conditions instead 9.6kbps is used (TO #2077)
Library:	All libraries
ASIC:	200 and 300 series
Consequence:	Increased latency because 9.6kbps is sometimes used instead of 40kbps.
Resolution:	Correct speed is set.

Headline/TO:	A slave node being included can occasional abort the findneighbor function causing an incomplete neighbor result (TO #2078)
Library:	All slave libraries
ASIC:	200 and 300 series
Detailed Description:	A node in process of being included will for a period try help in the NWI process by repeating Auto Inclusion Explore frame requests received from not already included nodes. If one of these frames is received while the node is in its own findneighbor sequence then the repeating of the Auto Inclusion Explore frame request can stop the findneighbor sequence and thereby create a possible flawed rangeinfo for the node in question.
Consequence:	Incomplete neighbour result for included slave node.
Resolution:	Ignore other nodes when conducting neighbor search.
Headline/TO:	ZW_RequestNodeNeighborUpdate() from static controller can't always reach the destination in case network topology changes (TO #2081).
Library:	Static and Bridge Controller libraries
ASIC:	200 and 300 series
Detailed Description:	ZW_RequestNodeNeighborUpdate() will on a static controller use the existing routing table to do the rediscovery resulting in situations where ZW_RequestNodeNeighborUpdate() will never be able to reach all nodes in the network. This happens when a node is moved from out of direct range to within direct range and out of range of previous neighbors. This scenario will typically only happen in small networks.
Consequence:	Static controller requesting node neighbor updates will not get a fully updated network topology.
Resolution:	New route resolution mechanism introduces direct try.
Headline/TO:	ZW_ReplaceFailedNodeID can only use low power RF transmission. (TO #2177)
Library:	All controller libraries
ASIC:	200 and 300 series
Consequence:	Shorter inclusion range
Resolution:	It is now possible to select low or normal power.

Headline/TO: Explore frame have a 7% failure rate in reaching wanted destination. (TO #2188)

Library: All libraries

ASIC: 200 and 300 series

Consequence: Increase robustness of links by avoiding asymmetric links.

Resolution: Explorer frame now uses the same reduced power as find neighbor call.

Headline/TO: A FLIRS node is used as repeater for sending explorer frame (TO #2194)

Library: All controller libraries

ASIC: 200 and 300 series

Detailed Description: A FLiRS node will repeat explorer frames in case it is awake.

Consequence: Explorer frame necessary again for next command in case a FLiRS node is returned as repeater as a result of the explorer mechanism.

Resolution: Now FLiRS nodes do not repeat explorer frames.

Headline/TO: FLiRS (DoorBell) nodes use too much power. (TO #2212)

Library: All libraries

ASIC: 200 and 300 series

Consequence: Shorter battery lifetime

Resolution: Optimized initialization sequence with respect to random seed generation.

Headline/TO: Non-listening node used as repeater for explorer frame. (TO #2214)

Library: All libraries except static and bridge controller libraries

ASIC: 200 and 300 series

Consequence: Non-listening node may enter sleep mode when destination node try to return route resolution to source node.

Resolution: Now non-listening nodes do not repeat explorer frames when awake.

Headline/TO:	Explorer frame kicks-in to fast. (TO #2219)
Library:	All libraries
ASIC:	200 and 300 series
Consequence:	Lower probability that last route attempt is successful when followed by an explore frame.
Resolution:	Direct attempt inserted before trying an explore frame.
Headline/TO:	Routing/Enhanced Slaves trying response route as an alternative to return routes. (TO #2221)
Library:	Routing and enhanced libraries
ASIC:	200 and 300 series
Consequence:	Will not use a potential response route to the wanted destination, which is the latest successful route.
Resolution:	Try response route if exists prior to using return routes for the wanted destination.
Headline/TO:	A node using explore frames stores the search result route incorrectly in the response route structure. (TO #2249)
Library:	All slave libraries
ASIC:	200 and 300 series
Consequence:	Response route may not work afterwards.
Resolution:	Saves now search result route correctly in response route structure.

APPENDIX C FIXED DEFECTS IN 4.50 (BETA1) PATCH1

The following defects are fixed compared to Developer's Kit v4.50 (Beta1):

Headline/TO:	Keil compiler >7.50 reports warning C235 when compiling serialapi_enhanced... target (TO #2133)
Library:	All libraries
ASIC:	200 and 300 series
Detailed Description:	<p>Keil compiler >7.50(8.xx) have been changed somehow in the preprocessor part.</p> <p>In ZW_transport_API.h the ZW_LockRoute has a definition for Controllers and one for Slaves as shown below:</p> <pre>#ifndef ZW_SLAVE Void ZW_LockRoute(BYTE nodeID); #endif #ifdef ZW_CONTROLLER Void ZW_LockRoute(BOOL bLockRoute); #endif</pre> <p>At the second expression of ZW_LockRoute the Keil v8.xx compilers/preprocessor reports a Warning C235, telling the parameter has changed – when compiling for a Slave target. If target is Controller no warning is generated. Keil v7.50 gives no warning in either case.</p>
Consequence:	Target is not build.
Resolution:	<p>Change code above to:</p> <pre>#ifndef ZW_Slave Void ZW_LockRoute(BYTE nodeID); #else Void ZW_LockRoute(BOOL bLockRoute); #endif</pre>
Headline/TO:	Frame transmitter stops working occasionally in a serial API stress test (TO #2141).
Library:	All libraries
ASIC:	200 and 300 series
Consequence:	Necessary to reset serial API to resume communication.
Resolution:	Fixed hazardous transmit queue use.

Headline/TO:	A Controller, which has transmitted a frame to a FLiRS node via a repeater, may not detect the ACK sent by the repeater (TO #2166).
Library:	All Controller libraries
ASIC:	200 and 300 series
Detailed Description:	Route ACK from first trail will typically reach source before retransmit timeout.
Consequence:	Source may not detect that frame actually reached destination and start to retransmit.
Resolution:	ACK frame is now accepted even though silent ACK was expected.
Headline/TO:	Controllers sends wakeup beam to repeater when routing (TO #2185).
Library:	All controllers
ASIC:	200 and 300 series
Detailed Description:	A controller that first sends a singlecast to a FLiRS node and then continues to route the frame will erroneously send a wakeup beam to the repeater.
Consequence:	Result in increased latency when transmitting. Experience a one-second delay on application level when condition occurs.
Resolution:	Removed wakeup beam to repeater.
Headline/TO:	Routing slave sends wakeup beam to repeater when routing (TO #2189).
Library:	Routing and Enhanced Slaves
ASIC:	200 and 300 series
Detailed Description:	When a routing slave is using a return route to send a routed frame to a FLiRS node it erroneously sends a wakeup beam to the repeater.
Consequence:	Result in increased latency when transmitting. Experience a one-second delay on application level when condition occurs.
Resolution:	Removed wakeup beam to repeater.

APPENDIX D FIXED DEFECTS IN 4.50 (BETA1)

The following defects are fixed compared to Developer's Kit v5.02:

Headline/TO: ZW_RequestNewRouteDestinations result in SUC_Update.updateState not reset before callback (TO #1398)

Library: Routing slave and enhanced slave library

ASIC: 100 and 200 series

Consequence: All SUC related functionality called in the callback function from ZW_RequestNewRouteDestinations will fail.

Resolution: The state is now changed to idle before the callback to the application is made.

Headline/TO: Response route is removed when transmitting direct to a destination node and the attempt fails (TO #1421)

Library: All

ASIC: 100 and 200 series

Consequence: If a frame is send without routing to a destination where the protocol has a cached response route the response route will be deleted if the direct transmission fails.

Resolution: Response route is not deleted if direct transmission fails.

Headline/TO: A node can during inclusion accept a new Node ID assignment from another controller (TO #1422)

Library: All

ASIC: 200 and 300 series

Consequence: If several inclusion controllers are trying to add a node to the network simultaneously, the node might be included several times wasting node IDs in the network.

Resolution: Accept only one assignment.

Headline/TO: The API call ZW_SetSUCNodeID is acknowledge using normal power regardless the transmission power specified in the transmit power option (TO #1453)

Library: Controller libraries

ASIC: 100 and 200 series

Consequence: None.

Resolution: Acknowledge used now transmission power specified in the transmit power option.

Headline/TO: ZW_SetSleepMode serial API call missing (TO#1468)

Library: All libraries except Static and Bridge Controller

ASIC: 100, 200 and 300 series

Detailed explanation: ZW_SetSleepMode serial API call is documented in [4] but unfortunately not implemented.

Consequence: It is not possible to put the device in Sleep mode from an external host processor via the Serial API.

Resolution: Support implemented in serialappl.c, serialappl.h and ZW_serialAPI.h files.

Headline/TO: APPLICATION_NODEINFO_OPTIONAL_FUNCTIONALITY assigned a wrong value (TO #1480)

Library: All libraries, because all can include ZW_controller_api.h

ASIC: 100 and 200 series

Consequence: APPLICATION_NODEINFO_OPTIONAL_FUNCTIONALITY addresses the wrong bit in the node information frame.

Resolution: #define APPLICATION_NODEINFO_OPTIONAL_FUNCTIONALITY 0x80 is now declared in header file ZW_controller_api.h

Headline/TO:	ADC_SetAZPL has now effect with respect to configuration of the sample period. (TO #1554)
Library:	All libraries
ASIC:	200 and 300 series
Detailed Explanation:	Sample period is written erroneously to the ASIC register
Consequence:	The total conversion time is approximately 60uS independent of the selected sample period.
Resolution:	The sample period is shifted six times to the right in the API call ADC_SetAZPL to obtain correct sample period.
Headline/TO:	Static controller sends routed data at 9.6kbps despite all nodes support 40kbps in the network (TO #1570)
Library:	Static and Bridge Controller libraries
ASIC:	200 and 300 series
Detailed explanation:	The static controller uses 40kbps direct. In case direct fails the routing attempts uses only 9.6kbps.
Consequence:	Result in increased latency in case routing is necessary.
Resolution:	The speed of routed frames is not calculated directly from repeater and destination speed information.
Headline/TO:	In stress test scenarios such as broadcasting a "I'm lost" frame to several helping nodes a SUC/SIS can sometimes unintentionally ask the lost node to look for neighbors using an defective nodemask (TO #1571)
Library:	Static and Bridge controller libraries
ASIC:	100, 200 and 300 series
Detailed explanation:	During the "I'm lost" process is the send data callback not used as handshake resulting in unintentional access to the transmit buffer before all data is transmitted. This may occur in communication stress test scenarios and is seen when broadcasting an "I'm lost" frame to several helping nodes.
Consequence:	A number of potential neighbours are missing in returned response.
Resolution:	Callback is now used as handshake before neighbour request is written to RF transmit buffer.

Headline/TO:	Erroneous timeout error handling when a 4 hop routing attempt fails. (TO #1574)
Library:	All
ASIC:	100, 200 and 300 series
Consequence:	Increased latency due to erroneous timeout error handling when a 4 hop routing attempt fails. The increase latency is only observed in 10% of the above mentioned cases.
Resolution:	Timeout error handling parameter extended to avoid overflow when a 4 hop timeout is calculated.
Headline/TO:	ZW_RequestNodeNeighborUpdate function does not work correctly, when called from application (TO #1578)
Library:	All controllers
ASIC:	100, 200 and 300 series
Detailed Description:	When calling ZW_RequestNodeNeighborUpdate after an attempt to include a slave node, which is already part of the network, "Get Nodes In Range" fails. "Find Nodes In Range" is sent to the correct target node, but "Get Nodes In Range" is sent, retried and not acknowledged to NodeID = 0.
Resolution:	A more comprehensive test for controller being busy doing another update is done in the beginning of the ZW_RequestNodeNeighborUpdate function.
Headline/TO:	Routing slave returns "done" to application before ZW_RediscoveryNeeded is fully completed (TO #1579)
Library:	Routing and Enhanced Routing Slave libraries
ASIC:	100, 200 and 300 series
Consequence:	When ZW_RediscoveryNeeded returns ZW_ROUTE_UPDATE_DONE the SUC/SIS return routes are not assigned yet. The SUC/SIS return routes will therefore not be assigned in case the application power off RF immediately after ZW_ROUTE_UPDATE_DONE is received. This is a possible scenario in battery operated applications.
Resolution:	ZW_RediscoveryNeeded now returns ZW_ROUTE_UPDATE_DONE when it is fully completed.

Headline/TO:	ZW_RequestNodeNeighborUpdate "completes" too early (TO #1595)
Library:	All
ASIC:	100, 200 and 300 series
Detailed explanation:	Under normal operation, the ZW_RequestNodeNeighborUpdate function waits for a "Command Complete" message to come back from the node it has requested the update from. However, occasionally the node doing neighbor rediscovery sends the "Command Complete" message, when the rediscovery is only about half complete. When the controller subsequently sends the "get nodes in range" message and gets a response, the "Nodes in Range" response is based on only half of the network. Furthermore the response to the function is still "success/done". The "Nodes in Range" response is all nodes by default initialized as neighbors. So neighbors not checked will be returned as neighbors in the "Nodes in Range" response.
Consequence:	Increased latency due to an erroneous routing table.
Resolution:	Changed handling of timeout counter when waiting for command complete to avoid false timeouts.
Headline/TO:	ZW_RequestNodeNeighborUpdate "terminates" when receiving a "command complete" aimed for another node (TO #1601)
Library:	All
ASIC:	100, 200 and 300 series
Detailed Description:	If a ZW_RequestNodeNeighborUpdate command is "completed" early, it will still send the "command complete" message back to the requesting node when it is finish. In case the requesting node has gone ahead and started to request the node neighbors from another device. The "command complete" message from the previous request is not checked for node ID resulting in termination of the current request.
Consequence:	Network topology becomes inaccurate.
Resolution:	Returns now when ZW_RequestNodeNeighborUpdate is fully completed.
Headline/TO:	ADC running in continuous mode cannot be stopped when using ZW0201 (TO #1607)
Library:	All
ASIC:	200 series
Detailed Description:	ZW0201 ADC running in continuous mode cannot be stopped by calling ADC_Stop
Resolution:	Call ADC_Init using single conversion mode. Alternatively can interrupt be disabled.

Headline/TO:	SUC is not being updated with information about deleted nodes by Primary Controller (TO #1620)
Library:	All controller libraries
ASIC:	100, 200 and 300 series
Detailed Description:	When communication fails from the primary controller to the SUC about a deleted node, due to e.g. heavy Z-Wave traffic which prevents the primary controller from transmitting the frame, the SUC will not be updated about the deleted node by the primary controller at a later stage which is the intention.
Consequence:	SUC will not get information about the deleted node
Resolution:	We now make sure that all transmit errors during SUC update result in the update being marked as pending so it will be send later.
Headline/TO:	Slave nodes do not update direct reach speed information (TO #1626)
Library:	Slave, Routing Slave and Enhanced Slave
ASIC:	200 and 300 series
Detailed Description:	<p>A 9.6/40kbps slave node saves the speed information of the last two nodes that it has received a non-routed frame from i.e. singlecast with no routing. This TO can occur in the below scenarios:</p> <p><u>Scenario 1</u></p> <ul style="list-style-type: none">a. A 9.6/40kbps slave node (B) has saved 40kbps as the speed information from another 9.6/40kbps node (A)b. Node (A) have been replaced with a 9.6kbps only nodec. Subsequently node (A) transmit a frame to node (B) at 9.6kbpsd. Now node (B) will not update the speed information to 9.6kbps, causing it to respond at 40kbps to a 9.6kbps only node <p><u>Scenario 2</u></p> <ul style="list-style-type: none">a. A controller includes a 9.6kbps only slave node (A1)b. The controller is set back to default (ZW_SetDefault) without having excluded node (A1)c. The controller includes two 9.6/40kbps slave nodes (A2 & B)d. Node (A2) initiates direct communication to slave node (B) at 40kbpse. Node (B) has saved 40kbps as the speed information to node (A2)f. Node (A1) re-enters the network since it was not previously excluded; hence having the same home id and node id as node (A2)g. Node (A1) initiates direct communication to node (B) at 9.6kbpsh. Now node (B) will not update the speed information to 9.6kbps, causing it to respond at 40kbps to a 9.6kbps only node
Consequence:	A Slave node will erroneously transmit a frame@40kbps to a 9.6kbps only node
Resolution:	Slave does now update direct speed information correct.

Headline/TO:	A failed response route attempt may reduce number of routing attempts. (TO #1627)
Library:	All
ASIC:	200 and 300 series
Detailed Description:	A failed response route attempt fails to reset the routing algorithm creating new routing attempts.
Consequence:	Less routing attempts.
Resolution:	Reset the routing algorithm to start from beginning.
Headline/TO:	ZW_GetRandomWord returns zero for a static controller (TO #1665)
Library:	Static Controller
ASIC:	200 and 300 series
Detailed Description:	ZW_GetRandomWord fails because it requires the radio to be set in power down mode before deriving a random number.
Consequence:	It is not possible to retrieve random word.
Resolution:	Static controller power now radio down correctly before calling ZW_GetRandomWord.
Headline/TO:	Large routed frames based on silent acknowledge can occasionally fail in a large busy network throughout all the routing attempts to reach the destination node ID (TO #1671)
Library:	All libraries
ASIC:	200 and 300 series
Detailed explanation:	The routed acknowledge timeout in the source node is too aggressive when the silent acknowledged functionality is used in a busy network throughout all the routing attempts to reach the destination node ID. In a large network this may result in premature new routing attempts in case repeaters are busy, which can collide with the original routing attempt.
Consequence:	Increased latency
Resolution:	Adjusted timeout for silent acknowledge and changed routed error handling.

Headline/TO:	Controllers receives incorrect application frame parameters in ApplicationCommandHandler() (TO #1693)
Library:	Portable and installer controllers
ASIC:	200 and 300 series
Detailed Description:	The define ZW_PROMISCUOUS_MODE is used in the protocol header files ZW_basis_api.h and ZW_transport_api.h but is unfortunately missing in the application make files.
Consequence:	Incorrect application frame parameters returned to ApplicationCommandHandler().
Resolution:	The parameter BYTE destNode is now passed correctly to ApplicationCommandHandler().
Headline/TO:	ADC macro calls ZW_ADC_SELECT_AD3 and ZW_ADC_SELECT_AD4 are missing in ZW_adcdriv_api.h header file (TO #1695)
Library:	All libraries
ASIC:	200 and 300 series
Consequence:	Calling the mentioned macros in application will cause an unsuccessful compile.
Resolution:	Following defines added to ZW_adcdriv_api.h: <pre>#if defined(ZW020x) defined(ZW030x) #define ZW_ADC_SELECT_AD3 (ADC_SelectPin(ADC_PIN_3)) #define ZW_ADC_SELECT_AD4 (ADC_SelectPin(ADC_PIN_4)) #endif /*ZW020x,ZW030x*/</pre>
Headline/TO:	Including a virtual node result in an additional node range info frame from the bridge itself (TO #1698)
Library:	Bridge Controller libraries
ASIC:	200 and 300 series
Detailed Description:	The bridge sends two node range info frames when including a virtual slave node. It sends one from the virtual slave and one from the bridge node itself. This causes the callback function to report an inclusion error in the API call ZW_AddNodeToNetwork.
Consequence:	The primary/inclusion controller detect an inclusion failure even though the virtual slave was successful included.
Resolution:	Only one node range info frame is returned from the bridge when including a virtual slave.

Headline/TO:	9.6kbps node out of direct range of the transmitting 9.6/40kbps controller cannot be reached when certain conditions occur in a mixed 9.6 and 40kbps nodes network (TO #1712)
Library:	All controller libraries
ASIC:	200 and 300 series
Detailed Description:	<p>When a 9.6/40kbps controller tries to transmit to a 9.6kbps node not within direct range the controller will not try to route to the node when the following conditions applies:</p> <ol style="list-style-type: none">1. No cached response route to the 9.6kbps node available.2. The previous routing attempt to another 9.6/40kbps node used 40kbps speed successfully before transmission to the 9.6kbps node. <p>The error will not occur in case the previous routing attempt to a 9.6/40kbps node used 9.6kbps speed successfully.</p>
Consequence:	9.6kbps nodes out of direct range of the transmitting 9.6/40kbps controller not reachable when above conditions occur.
Resolution:	The controller node routes as last resort in above scenario.

Headline/TO:	A cached response route at 9.6kbps data rate will cause the destination node in question to use wrong data rate in the response, in case the node has successfully received a direct frame at 40kbps from the same source node prior to the transmission of the 9.6kbps response (TO #1719).
Library:	All libraries
ASIC:	200 and 300 series
Detailed Description:	<p>This TO can occur in the below described scenario:</p> <ol style="list-style-type: none">1. A direct "Get" command between two 40kbps nodes fails because the source node cannot hear the "Acknowledge" frame and the "Report" command from the destination node2. The source node then initiates routing at 9.6kbps data rate and the "Get" command is successfully received by the destination node3. Prior to the destination node initiating transmission of the "Report" command to the routed 9.6kbps "Get" command, it has received two direct "Report" command retransmissions at 40kbps from the same source node due to the description in above item 14. Again, the source node cannot hear the two direct "Acknowledge" and "Report" commands, resulting in the destination node to transmit a routed "Report" command at 40kbps through a 9.6kbps only route
Consequence:	Routed response frame fails to reach destination.
Resolution:	The node uses now correct speed for the cached response route in question.
Headline/TO:	Static controller does not send "Transfer End" frame after assigning new return routes to a lost routing slave (TO #1734).
Library:	Static and bridge controller libraries
ASIC:	100, 200 and 300 series
Detailed Description:	When a routing slave is lost and requests new return routes from the SUC, no "Transfer End" frame is sent by the SUC after the return routes are assigned.
Consequence:	ZW_RediscoveryNeeded() returns ZW_ROUTE_UPDATE_ABORT instead of ZW_ROUTE_UPDATE_DONE in the callback function after successful completion of the route assignment.
Resolution:	Controller sends now "Transfer End" frame after assigning new return routes to a lost routing slave.

Headline/TO:	Node supporting FLiRS is during inclusion requested to find FLiRS neighbors but not requested to report range info (TO #1748).
Library:	All controller libraries
ASIC:	200 and 300 series
Detailed Description:	The primary/inclusion controller timer can expire before the FLiRS neighbors search procedure has completed.
Consequence:	The node's capabilities with respect to transmitting wakeup beams to a FLiRS neighbour node are not reported back to the primary/inclusion controller.
Resolution:	Now requested to report range info.
Headline/TO:	FLiRS nodes can loose a new frame when powering down (TO #1753).
Library:	Routing and enhanced slave libraries
ASIC:	200 and 300 series
Detailed Description:	When a FLiRS node power down as a result of the application making a call to ZW_SetSleepMode() a beam frame can be lost because the API doesn't check if we are receiving a beam or frame before powering down.
Consequence:	Result in increased latency because the FLiRS node will first be aware of the new frame when retransmitted. Furthermore, will a secure FLiRS node that missed the wakeup beam causing retransmissions in the security handshake result in a timeout of the nonce timer before the frame reaches the destination.
Resolution:	The call ZW_SetSleepMode() check first for an incoming frame to itself before powering down.

Headline/TO:	ZW_SetLearnMode() returns ASSIGN_COMPLETE callback to early in FLiRS populated networks (TO #1757).
Library:	Routing and enhanced slave libraries
ASIC:	200 and 300 series
Detailed Description:	The problem happens in a network consisting of both always listening and FLiRS nodes. Up to three neighbour scans are necessary in case the network consist of always listening, 1000ms beam able FLiRS and 250ms beam able FLiRS. The slave returns erroneously ASSIGN_COMPLETE callback after each neighbor scan to the application.
Consequence:	The application can erroneously assume that the inclusion process is finished despite additional neighbour scans are carried out. The neighbour scans can be interrupted in case the FLiRS node calls ZW_SetSleepMode() causing an incomplete neighbor scan.
Resolution:	It returns only ASSIGN_COMPLETE callback after the last neighbor scan is finished.
Headline/TO:	ZW_RequestNodeNeighborUpdate() does not send wakeup beam to FLiRS nodes (TO #1771).
Library:	All controller libraries
ASIC:	200 and 300 series
Detailed Description:	Routine checking if a wakeup beam is necessary failed to return correct result.
Consequence:	Sleeping FLiRS node never waked up due to missing wakeup beam and therefore never heard request for neighbor update.
Resolution:	Routine detect now correct if wakeup beam is necessary.
Headline/TO:	ZW_RediscoveryNeeded() makes erroneously a ZW_SetLearnMode() callback (TO #1773).
Library:	Routing and enhanced slave libraries
ASIC:	200 and 300 series
Detailed Description:	When calling ZW_RediscoveryNeeded() on a slave node a callback is made to the callback function specified by the application in the ZW_SetLearnMode() with the status ASSIGN_RANGE_INFO_UPDATE
Consequence:	None.
Resolution:	Ignore ZW_SetLearnMode() callback.

Headline/TO:	ZW_RediscoveryNeeded have a too short abort timeout during neighbor search (TO #1775).
Library:	Routing and enhanced slave libraries
ASIC:	200 and 300 series
Detailed Description:	Abort (ZW_ROUTE_UPDATE_ABORT) kicks in when network comprises of 15+ nodes not within direct range of the node conducting neighbor search.
Consequence:	Neighbor search actually completes but ZW_RediscoveryNeeded returns erroneously callback ZW_ROUTE_UPDATE_ABORT before neighbor search is completed.
Resolution:	ZW_RediscoveryNeeded state machine changed to enter the correct state in the state machine resulting in a longer abort timeout.
Headline/TO:	SUC/SIS presence detection in an already included routing/enhanced slave node does not initiate request for route to SUC/SIS (TO #1776).
Library:	Routing and enhanced slave libraries
ASIC:	200 and 300 series
Consequence:	Slave does not acquire a SUC/SIS return route despite the SUC/SIS presence is notified to it.
Resolution:	Slave will now acquire a SUC/SIS return route.
Headline/TO:	ZW_RequestNodeNeighborUpdate() can fail on a FLiRS node when calling ZW_SleepMode (TO #1777).
Library:	Routing and enhanced slave libraries configured as FLiRS
ASIC:	200 and 300 series
Detailed Description:	When a controller does a rediscovery of a FLiRS node the rediscovery might fail in case the FLiRS node enter sleep mode during neighbor search.
Consequence:	Rediscovery is not completed.
Resolution:	A FLiRS node calling sleep mode will first enter sleep mode when a rediscovery has completed.

Headline/TO:	Pending updates to SUC/SIS fails if ZW_AddNodeToNetwork() is called (TO #1779).
Library:	All controller libraries
ASIC:	200 and 300 series
Detailed Description:	The problem is caused by the application calling ZW_AddNodeToNetwork(ADD_NODE_STOP,...) while the protocol sending the pending updates to the SUC/SIS.
Consequence:	SUC/SIS may not receive pending updates.
Resolution:	Make sure that only call ZW_AddNodeToNetwork(ADD_NODE_STOP,...) once after having called ZW_AddNodeToNetwork(ADD_NODE_ANY ADD_NODE_SLAVE ADD_NODE_CONTROLLER,...).
Headline/TO:	A routing slave node from a previous FLiRS populated network start to send wakeup beams to an always-listening node in a new network after assignment of return routes (TO #1782).
Library:	Routing and enhanced slave libraries
ASIC:	200 and 300 series
Detailed Description:	Wakeup beam information was not erased when excluding a FLiRS node from a network.
Consequence:	Will always send wakeup beams to this particular node ID which was a FLiRS node in a former network.
Resolution:	Beam information stored in flash is now deleted when new return routes are received.
Headline/TO:	Routing in 1000/250ms populated FLiRS networks can result in 1000ms wakeup beams to 250ms FLiRS (TO #1793).
Library:	All controller libraries
ASIC:	200 and 300 series
Detailed Description:	When a controller has transmitted a routed 1000ms wakeup beam, the following attempts to route to a 250ms FLiRS will result in a 1000ms wakeup beam. This happens because the former 1000ms wakeup beam bit is not cleared when building the new source routing frame.
Consequence:	Result in 1000ms wakeup beams to a 250ms FLiRS node causing increased latency and battery consumption.
Resolution:	The extended routing header in the transmit buffer where routing beam information is stored is now cleared between use.

Headline/TO:	SUC Presence notification does not work in all scenarios (TO #1798).
Library:	Routing slave, enhanced slave, static controller and bridge controller libraries
ASIC:	200 and 300 series
Detailed Description:	<p>When a controller detects presence of a SUC will it request the SUC node ID. This fails in the following cases:</p> <ul style="list-style-type: none">• A routing/enhanced slave will never respond on this request.• A static/bridge controller will only respond in learn mode, which is a very rare situation.
Consequence:	SUC node ID is not exchange in above scenarios.
Resolution:	Above libraries will now respond when requested of SUC node ID.
Headline/TO:	Default normal RF power for US still 0x1B (TO #1809).
Library:	All libraries
ASIC:	200 and 300 series
Consequence:	Default normal RF power lower than specified.
Workaround:	Change the normal RF power value to 0x2A
Headline/TO:	Controller in Promiscuous Mode responds to foreign protocol commands (TO #1828).
Library:	Portable and installer controller libraries
ASIC:	200 and 300 series
Detailed Description:	A node with enabled Promiscuous Mode will respond to logged protocol frames addressed to another node.
Consequence:	The Promiscuous Mode enabled node can potentially jeopardize exchange of protocol frames between other nodes in the network.
Resolution:	Will not respond to received protocol frames promiscuously received, i.e. not addressed to the node itself.

Headline/TO:	The API call ZW_ReplaceFailedNode triggers unintentionally ApplicationControllerUpdate (TO #1855)
Library:	All controller libraries
ASIC:	200 and 300 series
Detailed explanation:	See headline.
Resolution:	Call to ApplicationControllerUpdate removed.
Headline/TO:	Static and Bridge controller (without repeater functionality) with SUC/SIS enabled does not respond when a lost node is requesting help (TO #1871).
Library:	Bridge controller library
ASIC:	200 and 300 series
Consequence:	A lost node requesting help by calling ZW_RediscoveryNeeded will not get new return routes in case the SUC/SIS resides on a bridge controller.
Resolution:	Now removal of repeater functionality does not affect the nodes capability to respond to a lost node requesting for help.
Headline/TO:	SIS role disappears when a SIS controller power cycled immediately after assignment. (TO #1873).
Library:	Static and bridge controller libraries
ASIC:	200 and 300 series
Detailed explanation:	When the SIS controller has included at least one node the SIS role cannot disappear anymore.
Consequence:	The SIS role is lost in case the SIS controller is power cycled immediately after assignment of the SIS role.
Resolution:	Initialize non-volatile memory correct.

Headline/TO:	Enhanced Slave based FLiRS wakes up every one-second and calls application level (TO #1912).
Library:	Enhanced slave library. Notice that this TO is not present in routing slave library.
ASIC:	200 and 300 series
Consequence:	Increased battery consumption.
Resolution:	If a wakeup beam is not detected then the node returns to sleep mode immediately.
Headline/TO:	Lost silent acknowledge cause to many errors (TO #1952)
Library:	All libraries acting as repeaters
ASIC:	100, 200 and 300 series
Detailed Description:	If a repeater in a route does not hear a silent acknowledge and starts to retransmit the frame then there is a 15% chance of the routing attempt failing.
Consequence:	Frame transmission may fail.
Resolution:	Uses also routed acknowledge as silent acknowledge for the frame transmitted to destination.

APPENDIX E KNOWN DEFECTS INHERITED

Below a list of known defects, which is inherited from previous releases such as SDK 4.5x, 5.0x etc.

Headline/TO:	When excluding (resetting) a Node from a different HomeID, the API may return false success or false fails. (TO #571)
Library:	All controller libraries
ASIC:	100 and 200 Series
Detailed explanation:	<p>When a primary/inclusion controller resets a Node (AssignID 0) it sends a couple of NOP (No Operation) frames to the NodeID the reset node had before (e.g. NodeID 42), to ensure it does not reply (ACK) and is actually reset. When the controller is used to exclude a node from another network the NOP frame is transmitted with the wrong home ID i.e. the one of the resetting controller and not the one the excluded node had previously.</p> <p>In case the node didn't receive the "Assign ID" frame and is not reset, it will still not ACK the NOPs because they are not sent with its own HomeID.</p> <p>The primary controller will assume the Node is excluded properly because it never received any ACK's to the NOP's.</p> <p>In the same scenario, the network with the HomeID of the primary/inclusion controller may have a Node with a NodeID (e.g. 42) that corresponds to the NodeID of the node being reset.</p> <p>Because the primary/inclusion controller sends the NOPs with it's own HomeID, the existing Node 42 from its own network will ACK the NOPs if it is within direct range, and the controller will assume the exclusion went wrong, which is most likely not the case.</p>
Consequence:	The API may return success, despite the resetting actually failing (AssignID 0 was not heard), or may return failed despite actually being successful (AssignID 0 <u>was</u> heard).
Workaround:	None.

Headline/TO:	SIS based application receives UPDATE_STATE_ADD_DONE before the entire inclusion process is completed (TO #1289)
Library:	Static and Bridge controller libraries
Consequence:	Sending commands from the SIS to the newly included node immediately after UPDATE_STATE_ADD_DONE is received will fail in case the newly included node is out of direct range. UPDATE_STATE_ADD_DONE is called already when the node ID has been assigned to the new node, but the neighbor discovery process has not completed. Only when the neighbor update is complete and the information is send back to the SIS will the SIS be able to calculate a route to the new node.
Workaround:	After UPDATE_STATE_ADD_DONE is received then delay access to the newly included node to ensure neighbor information is available. The delay depends on the number of nodes in the Z-Wave network. Please contact zensys_support@sigmadesigns.com if you need assistance on calculating the expected delay as a function of the network size. In case ZW_SendData fails can it be retried a number of times using an additional delay.
Headline/TO:	ApplicationNodeInformation changes are not reflected on protocol level when ApplicationNodeInformation content is changed during startup or while application is running (TO#1311)
Library:	All libraries
ASIC:	200 and 300 series
Detailed explanation:	The ApplicationNodeInformation is read and stored internally by the Z-Wave protocol after ApplicationInitHW, but before ApplicationInitSW.
Consequence:	An application changing ApplicationNodeInformation during ApplicationInitSW, ApplicationPoll or ApplicationCommandHandler will not be reflected on protocol level.
Workaround:	Make sure the function ApplicationNodeInformation returns the correct values no later than during ApplicationInitHW.
Headline/TO:	When adding/removing nodes controller accepts frames from a foreign Home ID after assignment of Node ID (TO #1385)
Library:	All
ASIC:	200 and 300 series
Consequence:	During inclusion a protocol frame from another network could be accepted and start some protocol functionality that causes inclusion to fail.
Workaround:	None.

Headline/TO:	When ZW_RequestNodeNeighborUpdate is called an extra callback is issued after the function is completed. (TO #1546)
Library:	All controller libraries.
ASIC:	200 and 300 series
Consequence:	The extra callback is issued due to an error in the state machine. Status in the callback is invalid.
Workaround:	Ignore the extra callback.
Headline/TO:	Range information for the failing node is removed from the routing table when aborting the Replace Failed Node process (TO #1550)
Library:	All Controller libraries
ASIC:	200 and 300 series
Detailed explanation:	When initiating the Replace Failed Node process, the range information for the node in question is removed from the routing table if it does not reply with an acknowledge to the frame sent to it. If the user aborts the Replace Failed Node process at this point, it will no longer be possible to route to the node that was attempted to be replaced.
Consequence:	Not possible to route to the node that was attempted to be replaced.
Workaround:	Re-include the node that was attempted to be replaced or re-do the Replace Failed Node process.
Headline/TO:	When a controller is requested to do a network Update via a route command complete is not routed (TO #1558)
Library:	Static Controller, Bridge Controller
ASIC:	200 and 300 series
Detailed explanation:	During a routed rediscovery of a Static Controller, the Command Complete frame is missing as a reply to the Find Nodes in Range frame. However, the sending controller has a timeout to ensure the process is completed.
Consequence:	Added latency to routed rediscovery process of Static Controllers
Workaround:	None

Headline/TO:	Inclusion controller can disable SIS functionality on a static controller (TO #1561)
Library:	Static Controller and Bridge Controller libraries
ASIC:	200 and 300 series
Consequence:	Inclusion controllers cannot add/delete nodes to/from the network when the SIS functionality is removed.
Workaround:	In case a device is an inclusion controller it must not be allowed to disable SIS functionality on a static controller.
Headline/TO:	New Range registered is re-transmitted because of missing ACK (TO #1563)
Library:	Static and Bridge Controller
ASIC:	200 and 300 series
Detailed explanation:	When a 'New Node Registered' frame fails to the SUC then the following call of 'New Node Registered' result in 'New Range Registered' is transmitted twice because of missing ACK.
Consequence:	Increased latency during inclusion.
Workaround:	None
Headline/TO:	The receiving controller in a Controller Change process do not revert back to the Secondary Controller role if the replication procedure fails (TO #1568)
Library:	All Controller libraries
ASIC:	200 and 300 series
Detailed explanation:	During a Controller Change procedure when the Controller Change has failed because of a missing acknowledge to the frame instructing the secondary controller to become primary, the original Primary Controller will inform the Secondary Controller that the Controller Change failed and that it should not entering the Primary Controller role anyway. However the Secondary Controller ignores this frame which results in having two Primary Controllers in the network.
Consequence:	There will be 2 Primary Controllers in the network.
Workaround:	Re-do the Controller Change if it fails and the original Primary Controller will take the Secondary Controller role when replication is successful.

Headline/TO:	SUC Node ID persist in Primary Controller despite the SUC node has been removed from the network (TO #1591)
Library:	All controller libraries
ASIC:	200 and 300 series
Detailed explanation:	When a SUC node is removed by the primary controller, the SUC node ID is not cleared in the Primary Controller. The Primary Controller believes the SUC node still persists, which will cause confusion in the network.
Consequence:	When a new node is included or a node is excluded, the Primary Controller will continue sending New Node Registered to the SUC node ID when in fact the SUC node do not exist in the network. In addition the API call ZW_GetSUCNodeID will return the old SUC node ID. Essentially this will generate increased latency due to retransmissions, since the SUC node is not responding.
Workaround:	None except including a new SUC node in the network.
Headline/TO:	A node that is in the process of performing an RediscoveryNeeded will try to help another node (TO #1599)
Library:	All except a slave library
ASIC:	200 and 300 series
Detailed explanation:	If a node receives a LOST request it will try to pass it on to the SUC/SIS even if it is in the process of being rediscovered itself.
Consequence:	This cause the Rediscovery process to be slowed down.
Workaround:	None.
Headline/TO:	SUC does not delete response routes when receiving range info from a node (TO #1602)
Library:	Static, Bridge and Reduced Static Controller libraries
ASIC:	200 and 300 series
Detailed Description:	When a SUC receives a new range info frame from a node in the network as a result of a rediscovery of the node, the SUC does not delete its cached response route to that node
Consequence:	The SUC will continue to use an "old" route to a node even though there might be a faster route to the node now.
Workaround:	None

Headline/TO:	ZW_RequestNodeNeighborUpdate with own node ID has no effect (TO #1608)
Library:	All Controller libraries
ASIC:	200 and 300 series
Detailed Description:	When a controller based application calls ZW_RequestNodeNeighborUpdate with its own node ID as input parameter, the controller itself will not "PING" i.e. transmit NOP frames to the nodes it has been requested to find.
Consequence:	The controller will loose its links to other neighbour nodes.
Workaround:	Avoid calling ZW_RequestNodeNeighborUpdate with own node ID
Headline/TO:	Portable controller will not reply a Set SUC frame (TO #1610)
Library:	Portable and Installer Controller
ASIC:	200 and 300 series
Detailed Description:	Portable and Installer controllers do not contain the SUC/SIS functionality. In the scenario where an application holding the Primary controller role insist on an attempt to set the SUC role by transmitting Set SUC frame to the Portable or Installer controller, it will cause the Primary controller to hang since no reply is received
Consequence:	The Primary Controller will hang waiting for the reply
Workaround:	Do not set a Portable or Installer controller as always listening node
Headline/TO:	ZW_StoreHomeID do not update HomeID in Installer Controller library (TO #1625)
Library:	Installer controller
ASIC:	200 and 300 series
Detailed Description:	After changing the home ID by calling ZW_StoreHomeID in installer tool, the protocol will still be using the old home ID.
Consequence:	ZW_StoreHomeID do not update the variable storing the homeID
Workaround:	Power cycle the ASIC or use the watchdog to perform a reset immediately after calling ZW_StoreHomeID

APPENDIX F EMBEDDED SAMPLE APPLICATION SIZES

To minimize the overall memory footprint is the code optimization moved to the last step in the build process to assure minimum size across library and application. It is therefore not possible to list the library size alone. The optimization process is not linear function compared to the code entered because it depends on how different code pieces can be reused causing the size to jump up and down even when small changes are made. The table below shows code and data space usage for selected sample applications including MyProduct, which is an application without functionality.

Sample Application	Code [Bytes]	Data [Bytes]
Development Controller	32110	1609
Led Dimmer	29107	1415
Secure Led Dimmer	30383	1889
MyProduct – Routing Slave	24748	1261
Serial API – Controller Static without repeater, FLIRS and manual routing functionality	32488	2002
Serial API – Controller Static without SUC and repeater functionality	32489	1995
Serial API – Enhanced 232 Slave	22821	1777
Serial API – Slave Routing	28169	1819

Available code space is 32Kbytes and data space is 2Kbytes. A secure application requires typically 1-2 Kbytes extra code space.

Code and data space for Development Controller is located in the MAP file situated in directory:

C:\DevKit_4_52_01\Product\dev_ctrl\build\dev_ctrl_ZW030x_ANZ\Rels\dev_ctrl_ZW030x_ANZ.map

Look at the bottom of the file after code and xdata.