Software Release Note

Z-Wave 400 Series Developer's Kit v6.02.00

Document No.:	SRN12148
Version:	1
Description:	Guideline for developing 400 Series based applications using the application programming interface (API) based on Developer's Kit v6.0x
Written By:	JFR
Date:	2012-05-29
Reviewed By:	CHL;PSH;NTJ;JKA;BBR;JSI;TRO;SAR;DDA
Restrictions:	Partners Only

Approved by:

Date	CET	Initials	Name	Justification
2012-05-29	16:28:44	NTJ	Niels Thybo Johansen	

REVISION RECORD

Doc. Rev	Date	By	Pages affected	Brief description of changes
1	20120510	JFR	All	Initial draft based on SRN11967 – Z-Wave 400 Series Developer’s Kit v6.01.03
1	20120511	JFR	All Section 3 Section 6.2 Section 6.3	Added 6.02.00 and updated versions New features TO’s #3465, #3468, #3493, #3497, 3499, #3511, #3540, #3543 & #3587 added to list of fixed protocol defects. Updated know n defects with TO’s #3579, #3590, #3596 & #3627

Table of Contents

1 INTRODUCTION

The 400 Series Z-Wave Developer’s Kit is 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 400 Series Z-Wave Developer’s Kit version 6.02.00 is a matured version of 6.01.0x intended for 400 Series based products entering volume production.

Be aware of that Z-Wave Developer’s Kit version 6.02.00 requires Keil PK51 v9.0 or later and it is not possible to use earlier Keil PK51 versions than v9.0 in connection with version 6.02.00 because Keil changed object format.

This release corrects a number of software defects found in version 6.01.03. For details refer to the sections describing libraries, sample code etc. Furthermore, this release also introduces support for an external power amplifier to increase range, new PVT Tool, ZWaveProgrammer source code, and debugging with uVision4 IDE. For details, refer to the section 3.

The 400 Series Z-Wave Developer’s Kit version 6.0x is original based on version 4.50 but includes also functionality from 5.02. Refer to [22] for a detailed description of contents.

2 OVERVIEW

The 400 Series Developer's Kit version 6.0x combines the main functionality groups:

- The functionality known from ZDK 4.5x (non-secure and secure)
    - Explorer Frame route resolution
    - Network wide Inclusion
    - Remove reduced SUC functionality (Full replication only)
    - Random number generator
    - Application level security sample code
- The functionality known from ZDK 5.0x
    - Increased Route diversity calculation
    - Frequently Listening Routing Slave (FliRS)
        - Beam wakeup
- The new PHY functionalities in the 400 series ASIC
    - New additional 100kbps channel
    - Concurrently listening on 9.6/40kbps and 100kbps channels for sub-1GHz operation
    - Concurrently listening on three 100kbps channels for 950MHz[1] operation (JP)
    - Additional peripherals such as AES-128 engine, USB port, UART1 port, SPI1 interface, LED controller etc.
- Discontinued support of 200/300 Series ASIC's.

[1] Note: For support of the new 920MHz frequency band, use SDK 6.10.00.

3 DETAILED DESCRIPTION

Support of external power amplifier (02.00+)

The protocol now supports an external power amplifier to increase RF range of the 400 Series chips. By calling function ApplicationRFNotify every time the radio change state to enable control of the external power amplifier via output pins. Refer to [19] regarding an application note on how to achieve higher transmitter output power.

Micro PVT Tool included (02.00+)

The Micro PVT Tool contains 400 Series firmware to check the RF performance on a device regardless of the contents in the OTP. This makes the tool suitable to investigate the RF performance of un-programmed or already programmed devices and when performing PVT measurements. The only requirement for using the tool is that the programming interface to the chip must be available and the UART0 interface for communication. For further details, refer to [29].

Extension of Serial API Power Management (02.00+)

Extend control of serial port pin TXD0 allowing user to set it to high/low depending on power management application.

ZWaveProgrammer source code included (02.00+)

For easy integration into a production environment is the ZWaveProgrammer PC source code now available on the SDK. The ZWaveProgrammer [16] provides Windows GUI and interface to ATMega128 situated on the ZDP02A/ZDP03A Development Platform. For further details regarding communication protocol interface, refer to [30].

The Z-Wave Programmer firmware source code for the ATMega128 is already available.

Debugging with uVision4 IDE (02.00+)

The uVision4 IDE support debugging of embedded applications including utilization of breakpoints supported by the 400 Series chip. Refer to [25] regarding how to enable debugging in the uVision4 IDE. For details about the debugger facilities refer to the Keil uVision4 IDE documentation.

LED Dimmer sample code application enhancement (02.00+)

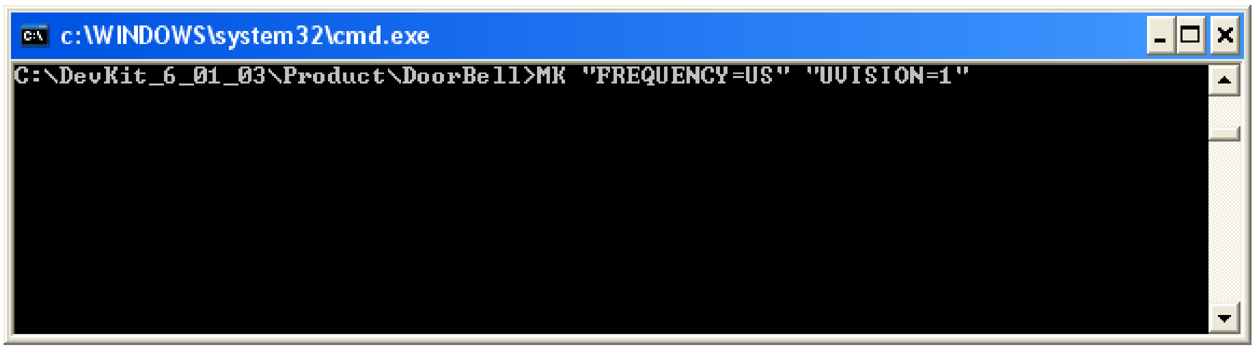
LED Dimmer supports now Meter Command Class preparing it to report current power consumption of connected load. The rate of Meter Report commands are determined by the Configuration command class. Refer to [4] for details.

Serial API Power Management (01.03+)

Serial API extended with calls supporting I/O handling for a broad range of applications such as power management of set-top boxes etc. The standby power of the device is reduced to powering the Z-Wave chip waiting to be activated by a Z-Wave enabled remote control. Depending on command received, the Z-Wave chip will power the appropriate functions and pass on the command to the host computer. Using a radio frequency signals based remote control does not require 'line of sight' to operate device as seen on traditional infrared remote controls.

uVision Project Generation (01.03+)

uVision project files are now generated by a tool in the DOS prompt called uVisionProjectGenerator. Generating uVision projects (normal mode, development mode and starter development mode) for a Doorbell using US frequency requires the following parameters in the DOS prompt:



```
C:\DevKit_6_01_03\Product\DoorBell>MK "FREQUENCY=US" "UVISION=1"
```

Notice that the uVision project files are now removed from all the embedded sample applications on the distributed SDK. For details, refer to [25].

USB adapter (01.02+)

The Z-Wave USB adapter based on a serial API static controller included on SDK [28]. UZB dongle exports the Serial API through a USB CDC/ACM compliant connection over a USB A plug. Both source and hex files included (serialapi_controller_static_UZB_USBVCP_EU.hex).

Streamlining patch system source code handling (01.01+)

Reduce time to market by simplifying alignment of source code between development and normal mode. The advantages are:

- Patchable source module can now be identical to the patch source module. Sync'ing these files are now much easier. In fact, the …_patch.c files are removed from SVN, and will be copied automatically during the build process.
- Global variables in patch modules can now be initialized just like in the normal program. - It is now possible to build an empty patchable application with library, and start with the complete application in development RAM as a patch.
- Patch source modules do not need to exist anymore. They will be copied from the patchable source module, when needed.
- When building an empty patchable application with library, the complete application patch contains standard wrappers for all standard ApplicationFunctions.

Refer to [25] for further details patch system and development environment.

Enhanced production test (01.01+)

LED Dimmer and Development Controller supports now in production test mode both receive of NOP frames in conjunction with Production Test Generator and carrier/modulated signal on the available channels in ApplicationTestPoll. Production Test DUT is not available anymore.

Support of serial API calls (01.01+)

Additional serial API calls are now supported extending accessible functionality when using a host processor in conjunction with a Z-Wave module hosting a serial API application. API calls enabled such as ZW_GetRoutingMAX, ZW_SetRoutingMAX, ZW_IsPrimaryCtrl, ZW_SendConst, ZW_SetListenBeforeTalkThreshold, ZW_SendTestFrame etc.

Serial API having USB VCP support (01.01+)

USB solution now supports virtual COM port (VCP) solution instead of direct USB connection.

Sample applications supporting ZM4102 (01.01+)

LED Dimmer and Serial API based static controller now also supports ZM4102 target. ZM4102 have less I/O's available and UART0 have moved to another pin position.

Endorse use of SIS functionality (01.01+)

Disable API call ZW_Enable_SUC to avoid applications disabling assignment of SIS role. The API call ZW_SetSUCNodeID will enable the SIS role if possible. It is not possible to only enable the SUC role anymore.

ZW_SendConst extended (01.01+)

ZW_SendConst extended with parameters allowing application layer to initiate constant RF transmission independently on each channel as well as whether signal is modulated or not modulated. Applies for both 2 and 3 channel systems.

Porting to 400 Series (01.01+)

Document and sample code showing the steps involved when porting from SDK 4.5x to 6.0x. Refer to [27] for further details.

Windows 7 based Zniffer and Programmer (01.01+)

Zniffer and Programmer now supports Windows XP/2003/Vista(32/64)/7(32/64). Zniffer log file format changed (*.zlf) to support Windows 7 and new FileConverter program introduced to open old *znf files.

PVT & RF Regulatory programs (01.01+)

PVT & RF Regulatory programs now use calibration value as input if available in OTP. All programs can be executed on one device for a given frequency because they are situated in the development mode SRAM part. One hex file containing start vector etc. must be present in OTP for wanted frequency.

Micro RF Link tool (01.01+)

Micro RF Link tool enable diagnostics of the RF link on a device regardless of the contents in the OTP. The tool execute in SRAM (Execution out of SRAM mode) having the following functionalities:

- for setting frequency
- for accumulative receive mode
- for rolling receive mode
- for transmit mode
- for transmit carrier mode
- for setting transmit power
- for displaying status

Refer to [26] for further details.

Calibrated Z-Wave modules (01.01+)

ZM4101 and ZM4012 modules are per default preprogrammed with a Calibration value in OTP. This means that the OTP by default is not empty. The calibration value is stored in OTP address 0x06. The programming and verification process when handling calibrated devices should thus be as described below:

- Blank testing: Waive for OTP content in addr. 0x06
- Programming: Ensure that data value 0x00 is written to OTP addr. 0x06 (this leaves the calibration value in the OTP unchanged)
- Verifying: Waive for difference between OTP content and hex-file content for addr. 0x06.

The SD3402 chip is not preprogrammed with a calibration value in OTP. Skipping the crystal calibration step will likely reduce the SD3402's sensitivity when operated near other Z-Wave devices. The problem diminishes as devices are spaced more than 25cm or 10 inch to other devices. Refer to [24] for further details.

Linux based sample app (01.01+)

Z/IP Router sample code containing an example of how to offer IP enabled services to a Z-Wave network. However, the current Z/IP command classes will be discontinued as part of a larger revision process.

Support for sub-GHz 9.6/40/100kbps using two channels including frequency agility

This version introduces a new 100kbps channel to decrease latency and increase throughput in home and entertainment control applications. A frequency agility scheme avoids interference from other systems by shifting channel when appropriate. The 400 Series ASIC listen concurrently on both channels. Communication robustness improved by adding a 16-bit CRC error check to the 100kbps frame.

Communications channels:

EU:

Ch0: 100kbps using 869.85MHz

Ch1: 9.6/40kbps using 868.40MHz

US:

Ch0: 100kbps using 916.00MHz

Ch1: 9.6/40kbps using 908.40MHz

ANZ:

Ch0: 100kbps using 919.80MHz

Ch1: 9.6/40kbps using 921.40MHz

HK:

Ch0: 100kbps using 919.80MHz

Ch1: 9.6/40kbps using 919.80MHz

IN:

Ch0: 100kbps using 865.20MHz

Ch1: 9.6/40kbps using 865.20MHz

MY:

Ch0: 100kbps using 868.10MHz

Ch1: 9.6/40kbps using 868.10MHz

NOTICE: 9.6/40kbps and 100kbps use same frequency for HK, IN and MY respectively.

Support for 951-955MHz 100kbps operation using three channels including frequency agility

This version introduces a new 100kbps communication speed using three channels to support operation in Japan. The 400 Series ASIC listen concurrently on all three channels. A frequency agility scheme avoids interference from other systems by shifting channel when appropriate.

JP:

Ch0: 100kbps channel using 950.951MHz (for 32.005MHz Crystal)

Ch1: 100kbps channel using 954.551MHz (for 32.005MHz Crystal)

Ch2: 100kbps channel using 955.351MHz (for 32.005MHz Crystal)

Communication robustness improved by adding a 16-bit CRC error check to the 100kbps frame.

New 400 Series peripherals

This version supports the following 400 Series built-in peripherals:

- AES-128 Engine supported by the API (only available in secure ZDK due to export restrictions).
- SPI0 interface available via the API. SPI1 is used by the protocol as the interface to the external non-volatile memory.
- Two UART's available via the API.
- The 8051 timer0 is available via the API.
- Z-Wave PWM API is revised with respect to naming and additional parameters added.
- IR learning and generation is supported via the API.
- Keyscan support enabling easy integration with keypad peripherals.
- LED controller supporting 4 channels 16 bit PWM.
- USB interface to increase throughput and reduce cost by avoiding an external USB to RS232 converter. USB interface complies with USB 2.0 Enhanced Host Controller (EHCI).

4 COMPATIBILITY AND UPGRADE

The following guideline is recommended when porting applications to the 400 Series Single Chip:

- Select an appropriate embedded sample application distributed on the Developer's Kit. Use typically a sample application using the wanted library.

- Rename sample application to the application in question.

- Move source code into application taking development mode (patch system) into account [22].

- Update API calls to 400 Series, especially API calls related to peripherals have changed significantly.

- Build application.

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

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 Z-Wave_Certification@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 random Home ID generation mechanism. 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 Z-Wave_Certification@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 Z-Wave_Certification@sigmadesigns.com.

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 the Keil CA51 Developer's Kit cannot be used anymore. Regarding 3rd party SW tools refer to [4].

The Keil Developer's Kit can be purchased via Sigma Designs.

6 Z-WAVE API LIBRARY

The Developer's Kit contains version 3.41 of the Z-Wave API library for the 400 Series ASIC. For a detailed description of the API library refer to [4], which is included on the Developer's Kit. The API consists of seven different libraries; a Portable Controller library, a Static Controller library (also one without repeater functionality), an Installer Controller library, a Bridge Controller library, a Routing Slave library, an Enhanced Slave library, and an Enhanced 232 Slave 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 6.02.00

Headline/TO:	ZW_SetSleepMode should enter sleep mode directly despite presence of a wakeup beam in case it is not a FLiRS node (TO #3465)
Library:	All slaves
ASIC:	200/300/400 series
Consequence:	A wakeup beam from same or adjacent networks can keep a battery operated device awake despite calling ZW_SetSleepMode causing additional battery consumption.

Headline/TO:	Calling ZW_SetRFReceiveMode can result in problems wrt. entering sleep mode when later calling ZW_SetSleepMode (TO #3468)
Library:	All slaves
ASIC:	200/300/400 series
Detailed Description:	Calling ZW_SetRFReceiveMode when receiving a wakeup beam causes problems when later calling ZW_SetSleepMode.
Consequence:	Result in small additional battery consumption before device enters sleep mode.

Headline/TO:	The ZW_RequestNodeNeighborUpdate functionality can in networks with FLiRS nodes make network management functionality stop working (TO #3493)
Library:	All controllers
ASIC:	400 series
Detailed Description:	Controllers in networks with FLiRS nodes, can when using the ZW_RequestNodeNeighborUpdate functionality end in a situation where network management functionality stops working. Same situation can happen during inclusion in a network already consisting of FLiRS nodes. Simultaneous RF traffic can trigger this defect.
Consequence:	Result in management functionality stop working such as ZW_AddNodeToNetwork, ZW_RemoveNodeFromNetwork, ZW_RemoveFailedNode, ZW_ReplaceFailedNode and ZW_RequestNodeNeighborUpdate.

Headline/TO:	ZW_SendData's callback returns TRANSMIT_COMPLETE_FAIL when using explorer (TO #3497)
Library:	All
ASIC:	200, 300 and 400 series
Detailed Description:	If ZW_SendData called with the explorer transmit option and the frame has been sent but the destination node was not reached then the callback function's status will be TRANSMIT_COMPLETE_FAIL instead of TRANSMIT_COMPLETE_NO_ACK.
Consequence:	Callback status is not the same on transmission fail, when using Explorer Transmit Option as when NOT using the Explorer Transmit Option

Headline/TO:	A Controller, which is in the middle of requesting a node to find its neighbors, will if it receives a FindNeighbor command start executing this instead. (TO #3499)
Library:	Static Controller and Bridge Controller libraries
ASIC:	200, 300 and 400 series
Detailed Description:	If a controller receives a "FindNodesInRange" frame, while it is requesting another node for the same ("FindNodesInRange"), it will forget about finishing the initial protocol command and therefore not finish it in a normal way.
Consequence:	ZW_RequestNodesNeighborUpdate can in some cases stall and not call the callback with a ZW_RequestNodesNeighborUpdate finished status

Headline/TO:	A Slave node can when doing Neighbor discovery sometimes echo a received command from a Controller back to the Controller (TO #3511)
Library:	All repeater capable libraries supporting FLiRS
ASIC:	200, 300 and 400 series
Detailed Description:	In a network containing both repeaters and FLiRS nodes a controller executing ZW_RequestNodeNeighborUpdate on a repeater node it can sometimes happen that the repeater node cases seemingly returns/echos back a, from the Controller received, "FindNodesInRange" command to the Controller.
Consequence:	The TO can make TO#3499 appear.

Headline/TO:	Override max. routing attempts (default is five) when using TRANSMIT_OPTION_EXPLORER flag (TO #3540)
Library:	All controllers
ASIC:	400 series
Detailed Description:	Source node will probably try more routes than specified maximum routing attempts (default is five) depending on the network topology before route resolution (explorer frame) kicks in.
Consequence:	In case destination node have moved to a new location resulting in an invalid network topology then source node will experience increased latency before route resolution (explore frame) kicks in.

Headline/TO:	Missing timeout when receiving a wakeup beam that never terminates due to incoming RF communication (TO #3543)
Library:	All slaves
ASIC:	200, 300 and 400 series
Detailed Description:	When a slave receives, a wakeup beam or any random bytes that resemble a beam then the protocol enter a state (receiving wakeup beam) to indicate that we are receiving a wakeup beam. This state prevents the node from sending any frame. Protocol clears state when the node receives a Z-Wave frame to itself or any random bytes that not resembles a wakeup beam.
Consequence:	Slave nodes cannot send any frames during this state

Headline/TO:	A primary controller can very seldom changed its role to secondary during a stress test comprising of parallel networks (TO #3587)
Library:	All controllers
ASIC:	200, 300 and 400 series
Detailed Description:	Two adjacent networks executing a stress test using all the available bandwidth (9.6/40kbps) and thereby creating explore frame storms and frame collisions. This setup can result in corrupted frames having correct checksum in case the appropriate bit errors occur. In case the three first bytes in the corrupted frame matches the transfer end protocol frame and destination node is a listening primary controller it can change role to secondary. It is very unlikely that this can happen for a non-listening primary controller because it is only exposed during operation.
Consequence:	Controller cannot include/exclude nodes in network.

6.3 Known defects

Refer to Appendix H for known defects, which is inherited from previous releases such as SDK 4.5x and 5.0x.

Headline/TO:	Using ZW_SetRFReceiveMode to save power during 9.6kbps silent acknowledge communication (TO #1660)
Library:	All libraries
ASIC:	400 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 library
ASIC:	400 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:	May not hear response from slaves.
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:	400 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:	400 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:	Find nodes in range frame retransmissions doesn’t get through (TO #1927)
Library:	All controller libraries
ASIC:	400 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:	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:	All
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 APPLIACTION_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:	Frequency agility on 3 channel (JP) systems can fail during inclusion (TO #2630).
Library:	Controller (3 channel)
ASIC:	400 series
Detailed Description:	Frequency agility changes channel on RSSI level to high and on missing acknowledge. However, inclusion starts with a broadcasted node information frame and that frame will only change channel on RSSI to high. So a situation where the including controller is jammed on one channel (Channel 0) and the node that should be included doesn't have a too high RSSI value on that channel (channel 0) then the node will not change channel and send the node information frame on another channel. Therefore, it will never be included because the controller will never receive the node information.
Consequence:	Inclusion fails
Workaround:	Repeat.

Headline/TO:	Direct, as last resort is not always send with the highest possible speed (TO #2729).
Library:	All controllers (2 channel)
ASIC:	400 series
Consequence:	Increased latency
Workaround:	None.

Headline/TO:	Controllers will, if a repeater being used in a Last Working Route is removed, try to use the Last Working Route with the removed repeater (and fail) until another successful route is found (TO #2769).
Library:	All controllers
ASIC:	400 series
Detailed Description:	The Last Working Routes are not updated when a repeater is removed Last Working Routes using the removed repeater will be tried until a new successful Route is found.
Consequence:	Destination not reached
Workaround:	Try again and after a successful transmission will the failing route be overwritten by a new Last Working Route

Headline/TO:	Misleading mode names for SPI (TO #2786).
Library:	All
ASIC:	400 series
Detailed Description:	SPI_SIG_MODE_1 to 4 should be SPI_SIG_MODE_0 to 3 instead
Consequence:	
Workaround:	Use SPI_SIG_MODE_1 to 4.

Headline/TO:	When a 100kb controller node transmits direct to a 100kb node with TRANSMIT_OPTION_ACK set but without TRANSMIT_OPTION_AUTO_ROUTE set, it only tries with 2 direct transmission on 100kb (TO #3026).
Library:	All controllers
ASIC:	400 series
Detailed Description:	When a 100kb controller tries to transmit to a 100kb node with both the TRANSMIT_OPTION_ACK and the TRANSMIT_OPTION_AUTO_ROUTE set, it first tries twice on 100kb then once with 40kb and then try routes etc. However, if only TRANSMIT_OPTION_ACK is set then the controller will only transmit twice on 100kb and then give up.
Consequence:	Will not reach a node out of direct range when above conditions are fulfilled.
Workaround:	Avoid using only TRANSMIT_OPTION_ACK.

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 node and repeaters does not handle routed ack as ack (TO #3096)
Library:	All
ASIC:	400 series
Detailed Description:	In case a source node or repeater does not hear ack during routing but frame reach destination. Then destination returns the routed ack and the node that missed the ack will not interpret the routed ack as an ack causing a retransmission.
Consequence:	Communication latency in case above happens.
Workaround:	None.

Headline/TO:	Controller including another controller routed (NWI) seems to purge the last working route used if including fails to follow the inclusion protocol properly (TO #3111).
Library:	All controllers
ASIC:	400 series
Detailed Description:	If the controller being included fails to transmit a Command Complete after acknowledging a TransferNodeInfo frame it seems that the including controller drops the last working route, which is the only route available and start transmitting directly.
Consequence:	Inclusion fails.
Workaround:	Require a new inclusion attempt.

Headline/TO:	Portable controller doesn't change HomeID when is is removed with a SIS (TO #3327).
Library:	Portable Controller
ASIC:	400 series
Consequence:	When a portable controller is removed from the network with a SIS it doesn't change it's own HomeID to another random ID. This can cause duplicated node IDs if the reset controller is used to build another network.
Workaround:	None

Headline/TO:	Static and Bridge Controller do not try to send direct after last working route was tried 3 times (TO #3579)
Library:	Static and Bridge controllers
ASIC:	200, 300 and 400 series
Consequence:	No extra (last resort) direct transmission attempt are done as part of the routing scheme
Workaround:	Add the Transmit Option Explore to the ZW_SendData call.

Headline/TO:	TRANSMIT_OPTION_LOW_POWER has no effect on transmit power when a frame is send (TO #3590)
Library:	Static and Bridge Controller
ASIC:	200, 300 and 400 series
Detailed Description:	When specifying the transmit option TRANSMIT_OPTION_LOW_POWER in the API the frame send out will be marked as low power but it will be send with normal power even though the Zniffer show it is a low power frame.
Consequence:	Low power transmission and inclusion will be done with normal transmit power
Workaround:	None

Headline/TO:	Unexpected callback when adding virtual node and inclusion fails (TO #3596)
Library:	Bridge Controller
ASIC:	200, 300 and 400 series
Detailed Description:	If a Bridge Controller (inclusion controller) tries to generate a new Virtual Node via the ZW_SetSlaveLearnMode(VIRTUAL_SLAVE_LEARN_MODE_ADD) and fails inclusion then callback status will return ASSIGN_COMPLETE, orgID=0 and newID=LastNodeIDAssigned. Either LastNodeIDAssigned will be 0 or the previous NodeID assigned.
Consequence:	A Virtual Node is not generated, but callback status returned is wrong/confusing/undocumented.
Workaround:	Receiving an ASSIGN_COMPLETE in VIRTUAL_SLAVE_LEARN_MODE_ADD mode indicate that the generation of a new Virtual Node has failed (communication with SIS failed). Repeat inclusion again.

Headline/TO:	A controller node in the process of including/excluding a node can acknowledge a route frame with a foreign home ID and same node (TO #3627)
Library:	All controllers
ASIC:	200, 300 and 400 series
Detailed Description:	If a controller in learning mode receives an ingoing routed frame with a foreign home ID and the destination ID is the same as it self, then the controller will ack the frame with the repeater as the destination node.
Consequence:	Frame in foreign network do not reach destination but frame is perceived as delivered successfully.
Workaround:	None

7 SAMPLE CODE FOR 400 SERIES 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.

7.1 Binary Sensor Sample Code

The Binary Sensor v3.57 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

- 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 overwriting the transmit queue during sending. (TO #551).
- Version Report CC returns wrong application version number. Workaround: Change `#include <config_app.h>` to `#include "config_app.h"` (TO #3650)

7.2 Secure Binary Sensor Sample Code

The Secure Binary Sensor v3.57 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@sigmadesigns.com for further information.

Known defects

- 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 overwriting the transmit queue during sending. (TO #551)
- Time waiting for inclusion must max. be 10 seconds to reduce window of malicious attacks. Now 20 seconds because fix of defect TO #3273 did not differ between inclusion and normal operation (TO #3445)
- Version Report CC returns wrong application version number. Workaround: Change `#include <config_app.h>` to `#include "config_app.h"` (TO #3650)

7.3 Battery Operated Binary Sensor Sample Code

The Battery Operated Binary Sensor v3.57 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

- 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 overwriting the transmit queue during sending. (TO #551).
- Version Report CC returns wrong application version number. Workaround: Change `#include <config_app.h>` to `#include "config_app.h"` (TO #3650)

7.4 Secure Battery Operated Binary Sensor Sample Code

The Secure Battery Operated Binary Sensor v3.57 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 distributed on the Developer's Kit CD due to export restrictions. Contact support via zensys@sigmadesigns.com for further information.

Known defects

- 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 overwriting the transmit queue during sending. (TO #551)
- Time waiting for inclusion must max. be 10 seconds to reduce window of malicious attacks. Now 20 seconds because fix of defect TO #3273 did not differ between inclusion and normal operation (TO #3445)
- Version Report CC returns wrong application version number. Workaround: Change `#include <config_app.h>` to `#include "config_app.h"` (TO #3650)

7.5 Development Controller Sample Code

The Development Controller v3.55 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

- 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 overwriting the transmit queue during sending (TO #551)

- Cannot initiated production test mode by short-circuit J16-pin3 and J17-pin1 on ZDP03A (TO #3470). Workaround: Add the following lines to the end of Makefile and recompile development controller application:
  CDEFINES+=,\
  APPL_PROD_TEST
- Version Report CC returns wrong application version number. Workaround: Change `#include <config_app.h>` to `#include "config_app.h"` (TO #3650)

7.6 Secure Development Controller Sample Code

The Secure Development Controller v1.29 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 ZDP03A Development Platform in combination with a Z-Wave module hosting a serial API based portable controller sample application. The supported/controlled command classes are embedded in the Security command class. The details can be found in [4] and [17], which are 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@sigmadesigns.com for further information.

Known defects

- 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 overwriting the transmit queue during sending. (TO #551).
- 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)

7.7 Door Bell Sample Code

The Door Bell v1.72 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

- 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 overwriting the transmit queue during sending. (TO #551).
- Version Report CC returns wrong application version number. Workaround: Change `#include <config_app.h>` to `#include "config_app.h"` (TO #3650)

7.8 Door Lock Sample Code

The Door Lock v1.49 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.

Known defects

- 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 overwriting the transmit queue during sending (TO #551)
- Version Report CC returns wrong application version number. Workaround: Change `#include <config_app.h>` to `#include "config_app.h"` (TO #3650)

7.9 Secure Door Lock Sample Code

The Secure Door Lock v1.49 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@sigmadesigns.com for further information.

Known defects

- 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 overwriting the transmit queue during sending (TO #551)
- Time waiting for inclusion must max. be 10 seconds to reduce window of malicious attacks. Now 20 seconds because fix of defect TO #3273 did not differ between inclusion and normal operation (TO #3445)
- Basic Command Class listed in NIF enabling unsecure operation. This command class must be security enabled (TO #3490)
- Version Report CC returns wrong application version number. Workaround: Change `#include <config_app.h>` to `#include "config_app.h"` (TO #3650)

7.10 LED Dimmer Sample Code

The LED Dimmer v3.56 sample code contains an example of a slave application. Detailed information regarding the LED dimmer sample code can be found in [4], which is included on the Developer's Kit CD.

New features

LED Dimmer supports now Meter Command Class preparing it to report current power consumption of connected load. The rate of Meter Report commands are determined by the Configuration command class. Refer to [22] for details.

Known defects

- 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 overwriting the transmit queue during sending. (TO #551)
- Version Report CC returns wrong application version number. Workaround: Change `#include <config_app.h>` to `#include "config_app.h"` (TO #3650)

7.11 Secure LED Dimmer Sample Code

The Secure LED Dimmer v3.56 sample code contains an example of a slave application. The supported/controlled command classes are embedded in the Security command class. Detailed information regarding the LED dimmer 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@sigmadesigns.com for further information.

Known defects

- 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 overwriting the transmit queue during sending. (TO #551).
- Time waiting for inclusion must max. be 10 seconds to reduce window of malicious attacks. Now 20 seconds because fix of defect TO #3273 did not differ between inclusion and normal operation (TO #3445)
- NIF contains support of five No Operation command classes (TO #3485)
- Version Report CC returns wrong application version number. Workaround: Change `#include <config_app.h>` to `#include "config_app.h"` (TO #3650)

7.12 My Product Sample Code

As an alternative to modifying the 400 Series sample code applications the My Product v1.61 can be used to build a slave application. The My Product contains the minimum framework to begin developing a slave application.

Known defects

None

7.13 Production Test Generator Sample Code

The Production Test Generator v1.75 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 defects

None

7.14 Serial API Sample Code

The Serial API v3.69 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. Detailed information about the Serial API sample code and how to interface to the Serial API can be found in [4], which are included on the Developer's Kit CD.

Known defects

- Missing serial API callback when requesting Node Information Frame. (TO #2358)
- Version Report CC returns wrong application version number. Workaround: Change `#include <config_app.h>` to `#include "config_app.h"` (TO #3650)

7.15 Utility Functions

Helpful functions used by several embedded sample applications v1.74. New files added to support controller learn mode and security.

Known defects

None

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 re-distributable 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 v4.45 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 v4.44 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 v4.44 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 v1.15 implement a XML parser, which enables parsing of Z-Wave frames by the Zniffer 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 and 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 Scaner 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 v4.57 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 Ctrl. should only accept disabling of retransmission when SerialAPI_Controller_Static_Single are used. Corrupted frames are generated when disabling retransmission using other serial API based libraries (TO #3370)
- PC Ctrl. includes other controllers using low power. It should be normal power. The option ADD_NODE_OPTION_HIGH_POWER can be added to the bMode parameter for normal power inclusion to extend inclusion range. (TO #3423)
- PC Ctrl. fails to authenticate incoming security enable frames after shifting primary role to another controller A followed by a replication from controller A to PC Ctrl. (TO #3458)
- PC Ctrl. stops sending Wake Up No More Information commands to a Binary Battery sensor after power reset (TO #3469)
- ERTT controller connected to Serial API Controller Single crashes when sending data with Tx Control enable (TO #3473)
- ERTT controller stops sending frames if the error rate is high and PC Controller crashes (TO #3474)
- Secure PC Ctrl. inclusion of nodes passes despite Delay Scheme Get set to 40 sec (TO #3477)
- Setting Invalid Network Key (Verify) on included Secure PC Ctrl. and try to include the controller to another Secure PC Controller secure inclusion doesn't fail (TO #3478)
- Missing routed ACK to PC Ctrl. but receiving command complete from command causes the PC Ctrl. to overwrite Tx buffer. Seen when PC Ctrl. request another node to find nodes in range (TO #3480)
- Build problem (TO #3677). As workaround comment line 179 in "DetectDevice" function in the CommonActions.cs:
  ```
  //ControllerManager.ZWaveManager.FrameLayer.IsRemoteSerialApi = isRemoteSerialApi;
  ```
  But in this case PC Controller will not work with remote devices.

9.2 Z-Wave Bridge Sample Code

The PC based Z-Wave Bridge v3.28 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 v2.28 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 v4.27 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 v2.30 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.

NOTICE: It is strongly recommended to always uninstall a previous release before installing the latest Zniffer release.

Known defects

- Zniffer can crash after long time having filtering enabled (TO #3188)
- Zniffer may not start after Windows updates on Windows XP computer. Reinstalling .NET Framework 3.5SP1 solves problem (TO #3433)
- Zniffer fails when decoding a secure frame without payload (TO #3447)

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 hexidecimal" 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 (v1.32 supporting 200/300 Series and v1.34 supporting 400 Series) enables the developer to conduct verification testing and regulatory measurements on the Z-Wave enabled products. For detailed information refer to [15] included on the Developer's Kit CD.

Known defects

None

10.5 Micro PVT tool

Micro PVT tool v1.00 enable check of RF performance on a device regardless of the contents in the OTP. This makes the tool suitable to investigate the RF performance of un-programmed or already programmed devices and when performing PVT measurements. The tool execute in SRAM (Execution out of SRAM mode). The only requirement for using the tool is that the programming interface to the chip must be available and the UART0 interface for communication.

Refer to [29] for further details.

Known defects

None

10.6 Micro RF Link tool

Micro RF Link tool v1.00 enable diagnostics of the RF link on a device regardless of the contents in the OTP. The tool execute in SRAM (Execution out of SRAM mode) having the following functionalities:

- for setting frequency
- for accumulative receive mode
- for rolling receive mode
- for transmit mode
- for transmit carrier mode
- for setting transmit power
- for displaying status

Refer to [26] for further details.

Known defects

None

10.7 Enhanced Reliability Test Tool

Functionality moved to PC Controller sample code.

Known defects

None

10.8 Z-Wave Programmer

The Z-Wave Programmer v2.45 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.

NOTICE: It is strongly recommended to always uninstall a previous release before installing the latest Z-Wave Programmer release

WARNING: ZDP02A/3A Programmer is not a production-graded unit, which may result in unintentional violation of chip programming specifications etc. when used in a production environment. It is strongly recommended to use an industrial production chip programmer.

Known defects

Known defects

- Disable Flash Read doesn't prevent the programmer to read the Flash content (TO #3403)
- Z-Wave Programmer crash if USB is removed from ZDP03A board and USB is selected under Settings (TO #3434)

10.9 SD3402 Crystal Calibration

The SD3402 crystal calibration firmware v1.19 used by the calibration box, refer to [24]. ZM4101 and ZM4102 are already calibrated during production. Hex file is located in Programmer folder.

Known defects

None

10.1 uVision4 IDE

The Keil uVision4 IDE used for developing Z-Wave applications as an alternative to the makefile system. How to configure uVision4 can be found in [20] included on the Developer's Kit CD.

WARNING: Configure uVision4 project as the makefile to obtain the same binary files. All options, order of compilation etc. must be the same.

Known defects

None

APPENDIX A FIXED DEFECTS IN 6.01.03

Headline/TO:	Controllers in 3 channel (JP) systems do not use routes to reach a node, which only should be reachable with routes (TO #3089)
Library:	All controllers (only 3 channel - JP)
ASIC:	400 Series
Detailed Description:	Controllers cannot route to nodes out of direct range.
Consequence:	Node out of direct range cannot be reached in certain scenarios: A SDK 6.01.03 controller can route via SDK 6.01.00, 6.01.01, 6.01.02 and 6.01.03 slaves using it as first hop etc. A SDK 6.01.02 controller can route via SDK 6.01.00, 6.01.01, 6.01.02 and 6.01.03 slaves using it as first hop etc. A SDK 6.01.01 controller can route via SDK 6.01.00, 6.01.01 and 6.01.03 slaves using it as first hop etc. A SDK 6.01.00 controller can route via SDK 6.01.00, 6.01.01 and 6.01.03 slaves using it as first hop etc.
Resolution:	Fixed speed bits in NIF to be backward compatible with a SDK 6.01.00/01 controller.

Headline/TO:	When a 100kbps node are requested to find its neighbors on 100kbps, the NOP_POWER frame is transmitted twice on 100kbps and then it falls back to 40kbps (TO #3110).
Library:	All
ASIC:	400 series
Consequence:	40kbps have slightly larger range than 100kbps, which may cause problems reaching an edge node detected by 40kbps.

Headline/TO:	Integrate USB VCP driver into library (TO #3120).
Library:	All
ASIC:	400 series
Consequence:	USB VCP driver file conbufio_udc.c situated in same directory as serial API sample application.

Headline/TO:	USB interface results in up to 50% CPU load depending on Windows OS (TO #3195).
Library:	All
ASIC:	400 series
Consequence:	Slow down Windows OS based computer considerably.

Headline/TO:	Explore search result could render nodes unable to transmit for 1 second (TO #3228)
Library:	All libraries
ASIC:	400 series
Detailed Description:	When a node receives a explore search result it could end up in a state where it is unable to transmit for 1 second. After 1 second the node would continue to operate normally.
Consequence:	A one second delay could sometimes occure when transmitting frames.

Headline/TO:	IR Tx buffer is limited to 256 bytes (TO #3241).
Library:	All
ASIC:	400 series
Consequence:	IR Tx buffer is limited to 256 bytes instead of 512 bytes.
Workaround:	None.

Headline/TO:	Keypad drive can unintentionally miss an interrupt in rare situations (TO #3246).
Library:	All
ASIC:	400 series
Detailed Description:	Interrupt input used for keypad cannot be held active until 8051 has handled other interrupts arriving simultaneously such as a RF interrupt.

Headline/TO:	Portable controller would use direct communication during Network Wide Inclusion (TO #3258)
Library:	Portable and Installer libraries
ASIC:	400 series
Detailed Description:	During network wide inclusion a portable controller could forget the route that was discovered via explore and start to use direct communication for the inclusion
Consequence:	The Network Wide Inclusion would fail.
Resolution:	The explored route is now locked during the inclusion so it can't be replaced by another route or direct.

Headline/TO:	API call ZW_PWM_waveform_set write to incorrect SFR registers (TO #3262).
Library:	All
ASIC:	400 series
Consequence:	Call has no effect.

Headline/TO:	Unresolved reference for API call ZW_RF_above_3v_supply_guaranteed (TO #3268).
Library:	All
ASIC:	400 series
Consequence:	Cannot use API call.

Headline/TO:	The triac API calls ZW_TRIAC_init and ZW_TRIAC_dimlevel_set set wrong registers. (TO#3309)
Library:	All
ASIC:	400 series
Detailed Description:	Do not address correct SFR page.
Consequence:	The triac API calls ZW_TRIAC_init and ZW_TRIAC_dimlevel_set do not work.

Headline/TO:	Calling ZW_SetDefault() during an SUC update may make ZW_SetRandomWord() return zeroes. (TO#3317)
Library:	Static Controllers
ASIC:	400 series
Detailed Description:	When calling ZW_SetDefault() on a SUC/SIS it may end up in a state where ZW_GetRandomWodr() no longer works. The problem occurs if the ZW_SetDefault() call is made while the SUC/SIS is busy sending updates to another controller.
Consequence:	ZW_GetRandomWord() returns zero

Headline/TO:	Static controller doesn't send wakeup beam to FLiRS node in direct range (TO #3318)
Library:	All libraries
ASIC:	400 series
Detailed Description:	When a secondary controller has been included it will not send wakeup beam to FLiRS nodes in direct range the first time it tries to transmit to them. The next time it transmits to the FLiRS node it will use the wakeup beam.
Consequence:	Latency will be longer or transmission will fail the first time a secondary controller transmits to a FLiRS node.

Headline/TO:	ZW_SendDataAbort() doesn't work for singlecast on routing slaves (TO #3323).
Library:	All Slave libraries
ASIC:	400 series
Consequence:	The call ZW_SendData() does not support the abort functionality of ZW_SendDataAbort() on Routing, Enhanced and Enhanced 232 slaves. ZW_SendDataMulti() can be aborted with ZW_SendDataAbort()

Headline/TO:	Routing Slave cannot send wakeup beam to FLiRS nodes on JP frequency (TO #3324).
Library:	Routing Slave
ASIC:	400 series
Consequence:	Routing Slaves can not be used to communicate with FLiRS nodes in the network

Headline/TO:	Write to external non-volatile memory (NVM) performed without identical check on data (TO #3342).
Library:	All controllers
ASIC:	400 series
Detailed Description	Controller updates external NVM (serial FLASH or EEPROM) when receiving acknowledge for a transmitted frame or when receiving a response as result of a request.
Consequence:	Traffic in the network can make controller exceed number of recommended write cycles of a NVM. For example each time a Controller polls a node it will initiate two write cycles on the controllers external NVM.

Headline/TO:	Wrong calculation of communication timeout in USB driver (TO #3400).
Library:	All
ASIC:	400 Series
Detailed Description	Wrong timeout in USBVCP SerialPutByte() function. The reference time and actual time in the comparison were reversed, resulting in a much too large timeout time.
Consequence:	Communication can in rare cases stop in max. 3 seconds.

APPENDIX B FIXED DEFECTS IN 6.01.02

Headline/TO:	Bridge controller cannot receive frames after inclusion of a virtual node by the bridge controller itself (TO #3078).
Library:	Bridge controller
ASIC:	All
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.

Headline/TO:	Nodes do not find all neighbors in 3 channel (JP) systems (TO #3082)
Library:	All controllers (only 3 channel - JP)
ASIC:	400 Series
Consequence:	Node will typically only find one neighbor limiting routing considerably.

Headline/TO:	Inclusion of nodes fails when the network get larger in 3 channel (JP) systems (TO #3088)
Library:	All controllers (only 3 channel - JP)
ASIC:	400 Series
Detailed Description:	Transmitter is unable to send due to nvm access, which delays transmission of ack frame.
Consequence:	Inclusion fails.

Headline/TO:	New Range Registered is wrong when SIS sends updates to inclusion controller in 3 channel (JP) systems (TO #3092)
Library:	Static and Bridge controllers (only 3 channel - JP)
ASIC:	400 Series
Detailed Description:	Inclusion controllers that tries to add nodes ot the network will receive wrong range information from the SIS about new nodes that was added to the network
Consequence:	Inclusion controllers will be unable to route to nodes that has been added to the network by other controllers.

Headline/TO:	A FLiRS node has difficulties wrt. wake-up in a in 3 channel (JP) system (TO #3098)
Library:	All (only 3 channel - JP)
ASIC:	400 series
Detailed Description:	A timing issue with respect to the fragmented wake-up beam caused problems in detecting the wake-up beam in 3 channel systems.
Consequence:	FLiRS node may not wake-up.

Headline/TO:	Unintentional software reset during execution (TO #3105)
Library:	All
ASIC:	400 Series
Detailed Description:	Random code execution via a low-level protocol call due to a function pointer set to null.
Consequence:	Software reset of module.

Headline/TO:	Customers are not able to compile any applications on the Z-Wave DevKit (TO #3118)
Library:	All
ASIC:	400 Series
Detailed Description:	An internal environment variable was used in SDK makefiles.
Consequence:	Could not compile applications.

Headline/TO:	TX timeouts to FLiRS nodes are too short (TO #3119)
Library:	All
ASIC:	400 Series
Detailed Description:	Retransmission timeout when sending to FLiRS nodes was incorrect due to wrong baud rate conversion of timeout duration.
Consequence:	Increased latency due to retransmissions.

Headline/TO:	Patched function are never executed (TO #3124)
Library:	All
ASIC:	400 Series
Detailed Description:	Patch table used wrong checksum causing program to skip all patched function.
Consequence:	Cannot execute patched functions.

Headline/TO:	Nodes stop to respond after running a stress test with two parallel networks (TO #3174 & 3177)
Library:	All
ASIC:	400 Series
Detailed Description:	Frame processing could switch to transmit during retrieval of data.
Consequence:	Transmitter stops working.

Headline/TO:	100kbps system can degrade to 40kbps under heavy RF Interference (TO #3185).
Library:	All
ASIC:	400 series
Detailed Description:	Two 100kbps nodes continue to use 40kbps explorer discovered route until that route fails again.
Consequence:	Result in increased latency due to lower speed.

Headline/TO:	Controllers can't send due to low threshold after running a stress test in 3 channel (JP) systems (TO #3196)
Library:	All controllers (only 3 channel - JP)
ASIC:	400 Series
Detailed Description:	Illegal new node registered frame can overwrite listen before talk threshold.
Consequence:	Stops sending due to too low threshold or transmitting when not allowed due to too high threshold.

APPENDIX C FIXED DEFECTS IN 6.01.01

Headline/TO:	SIS based application receives UPDATE_STATE_ADD_DONE before the entire inclusion process is completed (TO #1289)
ASIC:	All
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. Requesting node information frame to the newly included controller doing neighbor search will also result in its node ID changes to 0xEF.
Resolution:	The SUC/SIS is not notified (UPDATE_STATE_ADD_DONE) about the new node before completion of inclusion process.

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:	All
Consequence:	Increased latency due to using 9.6kbps when 40kbps is possible. Accompanying single casts to the multicast will all be send using 40kbps.

Headline/TO:	ZW_rf_tf.h file distributed despite it is obsolete (TO #1675)
Library:	All libraries
ASIC:	All
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:	In special cases can the routing algorithm use long time to determine that no alternative routes exist (TO #1831).
Library:	All controller libraries
ASIC:	All
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.

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:	All
Consequence:	Inclusion fails.
Resolution:	Ignore other nodes when conducting neighbor search.

Headline/TO:	ZW_RequestNodeNeighborUpdate() reports success on empty range info (TO #1953)
Library:	All controller libraries
ASIC:	All
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:	Secondary controllers cannot be re-replicated (TO #2054)
Library:	Controller libraries
ASIC:	All
Detailed Description:	The secondary controller will not reply to replication frames once it has been included.
Consequence:	Secondary controllers cannot be re-replicated.

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:	All
Consequence:	Increased latency because 9.6kbps is sometimes used instead of 40kbps.
Resolution:	Correct speed is set.

Headline/TO:	Controller with Promiscuous Mode enable cannot hear frames from the same network (TO #2086).
Library:	Portable and installer controller
ASIC:	All
Detailed Description:	
Consequence:	Promiscuous Mode do not work

Headline/TO:	Virtual node indicators are not removed on new replication. (TO #2268)
Library:	Bridge controller
ASIC:	200, 300 & 400 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:	Prod_Test_Gen 100kbps use 40kbps frame format (TO #2519).
Library:	Slave production test generator
ASIC:	400 series
Detailed Description:	
Consequence:	Zniffer cannot interpret production test 100kbps frames correct.

Headline/TO:	RF Power adjustments to fulfill changed JP regulations (TO #2762).
Library:	All (3 channel)
ASIC:	400 series
Detailed Description:	Requires that output power is configurable for each channel
Consequence:	

Headline/TO:	Network wide inclusion does not work out of direct range on JP (TO #2815).
Library:	All controllers (3 channel)
ASIC:	400 series
Detailed Description:	When trying to NWI include a node that is out of direct range of the including controller the controller doesn’t react on the repeated node information explore frame that is send to it, so the inclusion process never starts for nodes out of direct range.
Consequence:	Cannot include nodes out of direct range.

Headline/TO:	Incorrect 100K speed in Assign Return Route frames to FliRS destination (TO #2821).
Library:	All controllers (2 channel)
ASIC:	400 series
Detailed Description:	For a FliRS destination, the entire route speed must be 40K even if all repeaters, source and destination support 100K. When assigning return routes, the speed is erroneously set to 100K.
Consequence:	This bug has little impact because the beaming flag overrides the speed information, so the frame is transmitted at 40K anyway.

Headline/TO:	Missing dimming duration in Switch Multilevel Start Level Change. (TO #2845).
Library:	All
ASIC:	400 series
Detailed Description:	Wrong definition in ZW_classcmd.h compared to [21].

Headline/TO:	A 3-ch system (JP) cannot include virtual nodes to a bridge having the SIS role using another controller as inclusion controller. (TO #2865).
Library:	Bridge controller (3 channel)
ASIC:	400 series
Detailed Description:	Bridge sends NIF having a wrong node ID.
Consequence:	Cannot include virtual nodes to a bridge using another controller as inclusion controller.

Headline/TO:	Enhanced 232 slave do not use callback to signal to the application that a controller was excluding it (TO #2868).
Library:	Enhanced 232 slave
ASIC:	400 series
Detailed Description:	When an enhanced 232 Slave is in learnMode and a controller excludes it by ASSIGN ID homeID 0x00000000 and nodeID 0x00, it do not signal this to the application using the appropriate callback functionality.
Consequence:	When excluding an enhanced 232 slave is application never notified.

Headline/TO:	No macro/function for reading Timer0 interrupt flag (TO #2869).
Library:	All
ASIC:	400 series

Headline/TO:	Enhanced 232 Slave Beams to controller, which is not a FliRS node (TO #2872).
Library:	Enhanced 232 slave
ASIC:	400 series
Detailed Description:	Interpretation of response routes may cause a preceding wake-up beam to an always listening node, typically node ID 1.
Consequence:	Increased latency.

Headline/TO:	Missing ZW_SLAVE_ROUTING define in enhanced slave and enhanced 232 slave (TO #2874).
Library:	Enhanced slave and Enhanced 232 slave
ASIC:	400 series
Consequence:	ZW_SendDataAbort missing

Headline/TO:	A 6.0x slave node that have a 100kbps direct Response Route to a node ID that also matches a 200/300 series controller from another network introduce an additional NIF using 100kbps. (TO #2875).
Library:	All
ASIC:	400 series
Detailed Description:	The 6.0x slave node will transmit the Node Information Frame (NIF) at 100kbps based on the Response Route stored as answer to the Transfer Presentation from the 200/300 series controller. The 200/300 series controller cannot hear the NIF as 100kbps is not supported. A 6.0x controller node will always use 40kbps, which is supported by the 200/300 series controller.
Consequence:	Slave node is excluded because a slave always transmits a broadcast NIF when entering learn mode. Defect only have a latency impact due to an extra 100kbps NIF.

Headline/TO:	Duty cycle compliance in Japan (TO # 2882)
Library:	All libraries based on 3 channel
ASIC:	400 Series
Detailed Description:	In case that duration of carrier sense is over 128µs must the radio stop the radiation within 100ms from activation, it may retransmit after 100ms downtime, and furthermore the total duration of transmission shall be within 360sec per one hour. However, it may retransmit without 100ms downtime in case that duration between the beginning of first transmission and the end of retransmission is within 100ms.
Consequence:	Do not fulfil regulations in Japan

Headline/TO:	Node do not ACK retransmissions of routed ACK/ERR frames (TO # 2903)
Library:	All
ASIC:	400 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:	3-ch node does no transmit an Explorer Search Result frame as Response to an Explore Search Frame directed at it (TO #2904).
Library:	All
ASIC:	400 series
Detailed Description:	A node receiving an Explore Search frame, which has been repeated, is not answered with an Explore Search Result frame. However, the piggybacked payload is responded to so the frame is received. Repeater removes the ACK bit in the frame. This error applies only to 3-ch and not 2-ch systems.
Consequence:	3-ch repeaters remove ACK bit and thereby the destination do not respond with an Explore Search Result as intended.

Headline/TO:	Nodes receiving an Explore frame do not always use the explored route as Response Route/Last Working Route. The consequence is excessive use of explore frames in true ping-pong communication scenarios. (TO # 2905)
Library:	All
ASIC:	All
Detailed Description:	When a source node explores a destination node, both ends (the source node and the destination node) should have their respective response route/Last working route updated with the new explored route. This allows for an efficient GET / REPORT communication. (I.e only the first GET potentially triggers an Explore frame and all subsequent communication between the nodes uses single cast frames until an new change of the topology between the two occurs). In the event that the communication between the source node and the destination node is a true ping pong communication (one frames upstream always triggers exactly one frame downstream), this TO will cause destination nodes to loose its response route/Last working route when the source node is updated correctly through its explore frame and vice versa.
Consequence:	The consequence is higher latency in true ping-pong communication scenarios. In a true ping-pong communication scenario this TO will cause explore frames for all GET and REPORTS. At the time where the a true ping-pong communication is not present – the excessive use of explore frames will stop. (E.g. A SET followed by a GET /REPORT – 2 frames upstream -1 down stream).

Headline/TO:	ZW_TIMER0_ext_clk and ZW_TIMER0_ext_gate are not implemented (TO #2924)
Library:	All
ASIC:	400 Series
Detailed Description:	The functions ZW_TIMER0_ext_clk and ZW_TIMER0_ext_gate are defined in the ZW_appltimer.h but not implemented in the library

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:	400 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:	4.2x based inclusion controller cannot include nodes when using a 4.51 based SIS (TO # 2938)
Library:	Static and bridge controller
ASIC:	All
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:	A node that repeats a frame will store the route as a response/last working route and use the route with itself as repeater. (TO # 2942)
Library:	All except portable and installer controller (3 channel)
ASIC:	400 Series
Detailed Description:	A node that acts as repeater between node A and B will store the route as a response/last working route. When the node then later tries to send a frame to node A it will use the stored route and thereby try to use itself as a repeater.
Consequence:	Communication latency.

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:	200/300/400 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:	200/300/400 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:	200/300/400 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.

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: 200/300/400 Series

Detailed Description: 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.

Return routes and SUC return routes can also fail if they are destined for a controller without repeater functionality.

Routes beginning at controllers without repeater functionality can have increased latency because a direct range transmission is tried before routing.

Consequence: Routing to controllers without repeater functionality may fail completely.

Routing from controllers without repeater functionality may experience increased latency.

APPENDIX D FIXED DEFECTS IN 6.01.00

Headline/TO:	Explore frame have a 7% failure rate in reaching wanted destination. (TO #2188)
Library:	All libraries
ASIC:	200/300/400 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:	Enhanced/232 Slave based nodes does not send wakeup beams when using return routes (TO #2758).
Library:	Enhanced and enhanced 232 slaves
ASIC:	400 series
Detailed Description:	If an Enhanced/232 Slave based nodes is assigned return routes with beam information in the slave node will not send wakeup beam when using the return routes.
Consequence:	Enhanced/232 Slave node cannot communicate with FLiRS nodes.

Headline/TO:	Serial API static ctrl 400 series send Basic Set to Slave 400 series with 9,6 kbps if Slave 200 series was included first (TO #2783).
Library:	All controllers
ASIC:	400 series
Detailed Description:	
Consequence:	Increased latency

Headline/TO:	Repeaters send out invalid frames when repeated 100K frames are not ACK'ed (TO #2790).
Library:	All
ASIC:	400 series
Detailed Description:	A node repeating a routed 100K frame and not receiving an ACK or silent ACK would corrupt the frame before retransmitting it, leading to two corrupt transmissions from the repeater.
Consequence:	Routing attempt via that route would fail.

Headline/TO:	Controllers cannot include a virtual nodes in a controller bridge from devkits prior to 4.51. (TO #2794).
Library:	Controller portable, static and installer
ASIC:	400 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 devkit 6.0 controllers.

Headline/TO:	Enhanced 232 Slave do not repeat frames (TO #2795).
Library:	Enhanced 232 Slave
ASIC:	400 series
Detailed Description:	
Consequence:	Cannot act as repeater

Headline/TO:	Controllers do not route to 40kbs node through 100kbs repeaters (TO #2796).
Library:	All controllers
ASIC:	400 series
Detailed Description:	Controllers do not use 100kbps nodes as repeaters when trying to route to a 40kbps node in a network.
Consequence:	Controllers will have a limited or no routes to 40kbps nodes in a network with 100kbps nodes.

Headline/TO:	USB potential data corruption (TO #2802).
Library:	All
ASIC:	400 series
Detailed Description:	The 4K XRAM locations from 0x500 - 0x507 need to be reserved, since it is overwritten during enumeration.
Consequence:	Data corruption

Headline/TO:	RemoveNode Stop does not work (TO #2803).
Library:	All controllers
ASIC:	400 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:	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

Headline/TO:	Binary Sensors don't use all repeaters from Assign Return Route when routing to associated node (TO #2817).
Library:	Routing, enhanced and enhanced 232 slaves
ASIC:	400 series
Detailed Description:	Uninitialized non-volatile memory storing return routes.
Consequence:	Mesh network less robust

Headline/TO:	Beaming from Enhanced Slave to FLiRS node happens at wrong speed and beam sequence (TO #2820).
Library:	Enhanced Slave
ASIC:	400 series
Detailed Description:	Response routing missing beam info causing the first response route to fail.
Consequence:	Increased latency in case a response route exists to the wanted destination.

Headline/TO:	FLiRS nodes do not wake up on button press (TO #2825).
Library:	Routing, enhanced and enhanced 232 slaves
ASIC:	400 series
Detailed Description:	WUT handling in conjunction with interrupt from pressing button caused the problem
Consequence:	See headline

Headline/TO:	Power consumption is too high when 400 Series chip is in power down state (TO #2831).
Library:	All
ASIC:	400 series
Detailed Description:	Entered WUT/Stopped mode erroneously.
Consequence:	Inadequate battery lifetime.

Headline/TO:	When NoAck and AutoRoute is not set Slave (400series), will revert to 9.6 kbps after power cycling, and cannot recover back to 100kbit/sec (TO #2833).
Library:	Routing slave, enhanced slave and enhanced 232 slave
ASIC:	400 series
Detailed Description:	When slave is using senddata with TXOption NoAcknowledge and without AutoRoute, it does not use the assigned return route, and if it is power cycled the response route is lost, meaning it will use 9.6 kbps.
Consequence:	Increased latency.
Resolution:	Node never caches 9.6 kbps direct.

Headline/TO:	Not possible to adjust "Listen Before Talk" threshold level downwards to fulfil JP regulations (TO #2840).
Library:	All
ASIC:	400 series
Detailed Description:	API call ZW_SetListenBeforeTalkThreshold allows only adjustment upwards.
Consequence:	Difficult to fulfil JP regulations in case default "Listen Before Talk" threshold level must be changed downwards.
Resolution:	Can now also be adjusted downwards.

Headline/TO:	A controller does not cache a successful routed frame as last working route (TO #2848).
Library:	All controllers
ASIC:	400 series
Detailed Description:	After three unsuccessful direct attempts the following successful routed frame was tagged erroneously and therefore not stored as last working route.
Consequence:	Communication latency

Headline/TO:	Enhanced slave do not repeat or acknowledge routed frames in which it is listed as repeater (TO #2849).
Library:	Enhanced slave
ASIC:	400 series
Detailed Description:	Interpret frame header source routing info wrong.
Consequence:	Cannot support source routing mechanism.

Headline/TO:	A 2-ch system cannot include virtual nodes to a bridge having the SIS role using another controller as inclusion controller. (TO #2855).
Library:	Bridge controller
ASIC:	400 series
Detailed Description:	Bridge sends NIF having a wrong node ID.
Consequence:	Cannot include virtual nodes to a bridge using another controller as inclusion controller.

Headline/TO:	Controller skip explore frame attempt when last working route is populated (TO #2858).
Library:	All controllers
ASIC:	400 series
Detailed Description:	A controller having a last working route for destination will skip explorer frame as last resort to reach destination node.
Consequence:	Routing attempts without use of explorer frame

APPENDIX E FIXED DEFECTS IN 6.00.05 (BETA1)

Headline/TO:	Static Controller use routes to reply direct communication (TO #1655)
Library:	Static Controller
ASIC:	400 Series
Detailed Description:	In case a node has been included in the network and do not have direct range to the Static Controller (SC) the following scenario will take effect: 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

Headline/TO:	Assign Return Route sets wrong speed (TO #1841)
Library:	Controllers
ASIC:	400 Series
Detailed Description:	Controller sets route for FLiRS devices over repeating slave to 9.6kbit/sec but it is a 40kb/sec repeater device.
Consequence:	Increased latency

Headline/TO:	Replacing a SIS with "Replace Failed Node" doesn't work correctly (TO #2320).
Library:	Static controller
ASIC:	400 series
Detailed Description:	"Replace Failed Node" doesn't work correct at Static ctrl when it tries to replace failed SIS - Command Complete is not sent after Transfer Node Info
Consequence:	

Headline/TO:	ZW_SetLearnMode (in controllers) callback parameter bSource do not contain the new nodeID (TO #2478)
Library:	Controllers
ASIC:	400 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.

Headline/TO:	Assign SUC Return Route is missing when calling ZW_RediscoveryNeeded (TO #2528)
Library:	Static Controller
ASIC:	400 Series
Detailed Description:	Scenario: SIS network. Call ZW_RediscoveryNeeded from Slave node to Inclusion controller. The assign SUC Return Route frames are missing at the end of rediscovery.
Consequence:	SUC return routes are in some scenarios not assigned

Headline/TO:	When SUC ctrl makes associations and Assign Return Route with Binary sensor it doesn't use all existing repeaters from the network (TO #2626).
Library:	Static Controller
ASIC:	400 series
Detailed Description:	When SUC ctrl makes associations and Assign Return Route with Binary sensor it doesn't use all existing repeaters from the network
Consequence:	Non-optimal return route generation.

Headline/TO:	The Primary controller in a network is never informed about the SIS being disabled if the controller doing the disabling is not the Primary itself (TO #2659).
Library:	Controllers
ASIC:	400 series
Detailed Description:	When the Primary Controller does an Automatic Controller Update it gets a Z-WAVE_TRANSFER_UPDATE_DISABLED from the node which previously was SIS but do not use this information to remove the SIS information and return to primary controller functionality.
Consequence:	This means that no nodes can be added or removed in the network.

Headline/TO:	Serial API Slave Enhanced FLIRS doesn't send Wakeup Beam to FLIRS node (TO #2670)
Library:	Serial API Slave Enhanced FLIRS
ASIC:	400 Series
Detailed Description:	Serial API Slave Enhanced FLIRS doesn't send Wakeup Beam to FLIRS node
Consequence:	Communication cannot be established between the two nodes

Headline/TO:	Serial API Static controller doesn't send Wakeup Beam to Flirs when the ctrl has repeater function (TO #2671)
Library:	Serial API Static controller
ASIC:	400 Series
Detailed Description:	Serial API Static controller doesn't send Wakeup Beam to Flirs when the ctrl has repeater function
Consequence:	Communication cannot be established between the two nodes

Headline/TO:	Serial API Bridge Controller uses node ID 255 as Source Node ID when sending commands to included nodes, and therefore does not hear responses from nodes. (TO #2675)
Library:	Serial API Bridge Controller
ASIC:	400 Series
Detailed Description:	Scenario: When Bridge Controller sends Basic Set to a LED Dimmer, it uses Node ID 255 as its Source Node ID. Thus, the Dimmer uses 255 as Destination Node ID in ACK frames, which frames the Bridge Controller ignores, and resends the command 6 times + Explorer frame.
Consequence:	The bridge controller can not communicate with other nodes in the network.

Headline/TO:	Test for ZERO rangeinfo was done on every received rangeinfo not the combined rangeinfo (TO #2712).
Library:	Bridge Controller
ASIC:	400 series
Detailed Description:	When a node is requested to do a NodeNeighborUpdate it is asked to make several Ranginfo inquiries with different speeds (9.6/40 and 100). These rangeinfos should then be combined and the decision if a ZW_RequestNodeNeighborUpdate fails should be made on the combined rangeinfos.
Consequence:	Node Range Info only 100kbps nodes were presented.

Headline/TO:	SUC doesn't assign SUC Return Route to Slave Enhanced after removing repeaters from the network (TO #2715)
Library:	Static Controller
ASIC:	400 Series
Detailed Description:	Scenario: Add four Slaves to a SUC network and then remove 2 slaves from the network again with Primary Controller. When one of the remaining Slaves nodes sends Static Router request to the SUC, it doesn't assign SUC Return Route to that node.
Consequence:	SUC return Routes are in some scenarios not assigned

Headline/TO:	2-ch frequency agility (TO #2730).
Library:	All
ASIC:	400 series
Detailed Description:	The routing protocol will try both 40kbps and 9.6kbps communication on the same channel although 40kbps has already failed
Consequence:	Slower frequency agility due to the 9.6kbps communication attempt

Headline/TO:	All nodes accept the same Node ID during Network Wide Inclusion (3 channel systems) (TO #2743)
Library:	All
ASIC:	400 Series
Detailed Description:	When several nodes are in NWI Learn Mode simultaneously, they all respond with "Ack" to the first "Assign ID" from the Controller and accept the same Node ID.
Consequence:	Duplicated NodeIds

Headline/TO:	Slave nodes not in network enter NWI Learn Mode after they hear that a node inclusion has happened nearby (TO #2744)
Library:	All
ASIC:	400 series
Detailed Description:	After a node inclusion of any kind, all slave nodes in direct range that are not included in any network start sending Node Info in Explorer frame.
Consequence:	Un-intentional inclusion of nodes

Headline/TO:	Repeaters added after controller are not used in route generation after a controller replication(TO #2755)
Library:	Static Controller
ASIC:	400 Series
Detailed Description:	Repeaters added after controller are not used in route generation after a controller replication
Consequence:	Non optimal Routing

Headline/TO:	The update of Assigned Return Route feature has not been enabled for Enhanced/232 Slave nodes. (TO #2760)
Library:	Enhanced/232 Slave
ASIC:	400 Series
Detailed Description:	The update of Assigned Return Route feature has not been enabled for Enhanced/232 Slave nodes
Consequence:	The update of Assigned Return Route feature does not work

Headline/TO:	SIS Stress test (TO #2772)
Library:	Static Controller
ASIC:	400 Series
Detailed Description:	Scenario: SIS network slave nodes: are sending data ewery 30 sec, and making rediscovery every minute. Repeaters: are making linktest and rediscovery every 5 minute. SIS: send data every 5 sec static controllers: send data every 5 sec, network update every 1 min.
Consequence:	Updates are stopped

Headline/TO:	Explorer frames are not retransmitted by nodes in a 3 channel network (TO #2774)
Library:	All
ASIC:	400 Series
Detailed Description:	Explorer frames are not retransmitted by nodes in a 3 channel network
Consequence:	Explore functionality does not work in case a node wants to obtains new routes

Headline/TO:	Routed Response frame is missing an ack (TO # 2775)
Library:	All
ASIC:	400 Series
Detailed Description:	When a controller in network sends a routed basic get command to a node, the node receive the frame correctly but the routed basic response doesn't get back to the controller.
Consequence:	Communication failure when doing a Get, response communication

Headline/TO:	DoorBell Do not everytime Wakeup and ACK the frame transmitted to it. (TO #2780)
Library:	Slave
ASIC:	400 Series
Detailed Description:	Every other Transmit to the DoorBell is not ACKed including all retransmits.
Consequence:	Failing communication

APPENDIX F FIXED DEFECTS IN 6.00.04 (BETA1)

Headline/TO:	Frequency agility speed changes should not be cached by LastWorkingRoute (TO #2400).
Library:	All controllers
ASIC:	400 series
Detailed Description:	Frequency agility occasionally decides to use a route at a lower speed than supported. This is meant to be a temporary change. However, this decision is cached by the LastWorkingRoute, making it permanent.
Consequence:	40kbps will be used in situations where 100kbps could be used instead

Headline/TO:	Assign SUC Return Route is missing when calling ZW_RediscoveryNeeded (TO #2528).
Library:	Routing slaves
ASIC:	400 series
Detailed Description:	
Consequence:	
Workaround:	None.

Headline/TO:	100kbit/s modulation Gauss shaping (TO #2592).
Library:	All
ASIC:	400 series
Detailed Description:	
Consequence:	Maximum output power is limited to -4dBm

Headline/TO:	Automatic Controller Update on Inclusion Controller is not correct after rediscovery of the SlaveAPI in Lost mode (TO #2594).
Library:	All controllers
ASIC:	400 series
Detailed Description:	
Consequence:	

Headline/TO:	Crystal calibration of the JP based Z-Wave module as a part of the programming process is missing in patch2 (TO #2605).
Library:	All
ASIC:	400 series
Detailed Description:	
Consequence:	

Headline/TO:	When SUC ctrl makes associations and Assign Return Route with Binary sensor it doesn't use all existing repeaters from the network (TO #2626).
Library:	Static controller
ASIC:	400 series
Detailed Description:	
Consequence:	Routing and Enhanced slaves does not always get as many return routes as they shoud

Headline/TO:	Slaves use 9.6K instead of 100K routing via SUC Return (TO #2655).
Library:	Routing and Enhanced Slave
ASIC:	400 series
Detailed Description:	
Consequence:	Increased latency during SUC communication.

Headline/TO:	Timer0 is functioning (TO #2661).
Library:	All
ASIC:	400 series
Detailed Description:	Timer0 cannot be used by the application
Consequence:	

Headline/TO:	When a 100kbps node is included into a network containing 40kbps nodes it is asked for neighbor both by 100kbps and 40kbps but only the 100kbps rangeinfo is transmitted to the SUC/SIS (TO #2683).
Library:	All controller libraries
ASIC:	400 series
Detailed Description:	
Consequence:	Missing 40kbps rangeinfo.

Headline/TO:	SUC/SIS cannot Assign SUC Return Routes to 400 series FLiRS nodes (TO #2689).
Library:	Bridge and Static controller libraries
ASIC:	400 series
Detailed Description:	
Consequence:	400 series FLiRS nodes cannot communicate with SUC/SIS using return routes.

APPENDIX G FIXED DEFECTS IN 6.00 (BETA1) PATCH3

Headline/TO:	Same return route tried twice when target is not responding (TO #2277).
Library:	Routing slave
ASIC:	400 series
Detailed Description:	A routing slave using a return route will retry the route one extra time if the target does not answer.
Consequence:	Increased latency

Headline/TO:	Return routes assigned use in certain cases wrong speed on routing slave (TO #2423).
Library:	All routing slaves
ASIC:	400 series
Detailed Description:	Assigned Return Routes do not use the correct assigned speed when using the assigned Return Routes.
Consequence:	Increased Latency.

Headline/TO:	EEPROM_APPL_OFFSET not adjusted to enhanced 232 slave internal memory usage (TO #2508).
Library:	Enhanced 232 slave
ASIC:	400 series
Detailed Description:	The EEPROM_APPL_OFFSET is defined as 0x120 which is valid for normal Enhanced Slave but for a Enhanced 232 Slave it should be 0x1200.
Consequence:	Wrong addressing

Headline/TO:	Response route from Slave 400 Series to Static ctrl 400 Series via Slave 200/100 Series does not work correctly (TO #2536).
Library:	All slaves
ASIC:	400 series

Headline/TO:	Static controller does not try more then two entry points for each speed to reach slave node (TO #2546).
Library:	Static controller
ASIC:	400 series
Detailed Description:	One entry point can generate up to two route attempts depending on topology.

Headline/TO:	The primary controller in a network is not informed when the SIS is disabled if the controller that performed the disabling is not the primary itself. (TO #2659).
Library:	
ASIC:	400 series

Headline/TO:	When ApplicationCommandHandler receives a frame, which was received via a routed singlecast frame, it is noted as being from a multicast frame (TO #2660).
Library:	All
ASIC:	400 series

APPENDIX H 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:	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 is brought outside of direct range or in situations with poor communication between the controllers in question. (TO #458)
Library:	All controllers
ASIC:	400 series
Consequence:	Replication process will hang requiring a reset of the receiving controller to recover
Workaround:	Application on the receiving controller could start a new timer when LEARN_MODE_STARTED is received. This timer will be restarted every time a frame is received. In case the frame flow stops unintentionally the timer will expire and this must trigger a reset of the receiving controller. This could be done via an output pin or stop kicking the watchdog.

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:	400 Series
Detailed explanation:	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. 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. The primary controller will assume the Node is excluded properly because it never received any ACK's to the NOP's. 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. 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.
Consequence:	The API may return success, despite the resetting actually failing (AssignID 0 was not heard), or may return failed despite actually being successfull (AssignID 0 <u>was</u> heard).
Workaround:	None.

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:	400 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.
Workaround:	Call ZW_RequestNetWorkUpdate again in case the network topology is incomplete.

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:	400 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:	A node can during inclusion accept a new Node ID assignment from another controller (TO #1422)
Library:	All
ASIC:	All
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.
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:	400 Series
Consequence:	Increased latency due to repeated routing attempts.
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:	400 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:	400 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:	Missing Command Complete from Static Controllers in routed rediscovery (TO #1558)
Library:	Static Controller, Bridge Controller
ASIC:	400 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:	400 series
Consequence:	Inclusion controllers can not 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:	400 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:	400 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:	400 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:	400 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:	400 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:	400 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:	400 series
	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:	400 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 I 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 (2-channel systems)	Code [Bytes]	Data [Bytes]
Development Controller	42876	2447
Led Dimmer	35069	1885
Secure Led Dimmer	40889	2754
MyProduct – Controller Bridge	43868	2405
MyProduct – Controller Installer	38975	2288
MyProduct – Controller Portable	38849	2281
MyProduct – Controller Static	43018	2327
MyProduct – Controller Static without repeater functionality	41731	2325
MyProduct – Enhanced Slave	29476	1801
MyProduct – Enhanced 232 Slave	29622	2013
MyProduct – Routing Slave	28471	1714
Serial API – Controller Static	50610	3760
Serial API – Controller Static without repeater functionality	49407	3758
Serial API – Enhanced 232 Slave	35931	3426

Available code space is 64Kbytes and data space is 4Kbytes. A secure application requires typically 4-5 Kbytes. A 3-channel system (JP) can deviate +/- 200 Bytes compared to a 2-channel system (US, EU etc.).

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

```
C:\DevKit_6_02_00\Product\dev_ctrl\build\dev_ctrl_ZW040x_ANZ\Rels\dev_ctrl_ZW040x_ANZ.map
```

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

REFERENCES

[1] SD, DSH10717, Datasheet, Datasheet for ZW0301 Z-Wave Single Chip.
[2] SD, INS10244, Instruction, Z-Wave Node Type Overview and Network Installation Guide.
[3] SD, APL10748, Application Note, Porting Z-Wave Appl. SW from ZW0201 to ZW0301.
[4] SD, INS12034, Instruction, Z-Wave 400 Series Application Programming Guide v6.02.00.
[5] SD, INS10249, Instruction, Z-Wave Zniffer User Guide.
[6] SD, INS10309, Instruction, Using the Epsilon5 for programming the Z-Wave Single Chips.
[7] SD, INS10236, Instruction, Development Controller User Guide.
[8] SD, APL10248, Application Note, Porting Z-Wave Appl. SW from ZW0102 to ZW0201.
[9] SD, INS10240, Instruction, PC Based Controller User Guide.
[10] SD, INS10241, Instruction, PC Installer Tool Application User Guide.
[11] SD, INS10680, Instruction, Z-Wave XML Editor User Guide.
[12] SD, SDS10242, Software Design Specification, Z-Wave Device Class Specification.
[13] SD, INS10250, Instruction, Z-Wave DLL User Guide.
[14] SD, INS10245, Instruction, Z-Wave Bridge User Guide.
[15] SD, INS10336, Instruction, Z-Wave Reliability Test Guideline.
[16] SD, INS10679, Instruction, Z-Wave Programmer User Guide.
[17] SD, INS10681, Instruction, Secure Development Controller (ATmega) User Guide.
[18] SD, SDS10865, Software Design Specification, Z-Wave Security Application Layer.
[19] SD, APL10742, Application Note, ZM3102N with External PA and Switch.
[20] SD, INS10246, Instruction, Keil uVision3 IDE User Guide.
[21] SD, SDS11060, Software Design Specification, Z-Wave Command Class Specification.
[22] SD, INS12035, Instruction, Z-Wave 400 Series Developer's Kit Contents v6.02.00.
[23] SD, INS11043, Instruction, Z/IP Router User Guide (WL-500W).
[24] SD, INS11552, Instruction, 400 Series Crystal Calibration User Guide.
[25] SD, INS11709, Instruction, Working in 400 Series Environment User Guide v6.02.00.
[26] SD, INS11596, Instruction, Micro RF Link Tool.
[27] SD, APL10979, Application Note, Porting Z-Wave Appl. SW from ZW0301 to 400 Series.
[28] SD, INS11850, Instruction, UZB User Manual.
[29] SD, INS12131, Instruction, Micro PVT Tool.
[30] SD, INS11072, Instruction, Z-Wave Programmer Communication Protocol.

INDEX