



Instruction

Getting Started with Z-Wave™ Developer's Kit 4.5x

Document No.:	INS11245
Version:	2
Description:	This document describes guidelines on how to get started with the Z-Wave™ Developer's Kit.
Written By:	CHL;JFR
Date:	2009-03-22
Reviewed By:	CHL;JFR;MAM;SGR;SML
Restrictions:	Public

Approved by:

Date	CET	Initials	Name	Justification
2009-03-22	22:15:37	CHL	Chong Li	on behalf of NTJ

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



REVISION RECORD				
Doc. Rev	Date	By	Pages affected	Brief description of changes
1	20081202	JFR	All	Initial draft
2	20090201	CHL	3.1.3	Updated software directory structure
		JFR	5.1	Added door bell application
		CHL	Multiple	Updated several pages for ZDK 4.50 Beta1

Table of Contents

1	ABBREVIATIONS.....	5
2	INTRODUCTION.....	6
2.1	Purpose	6
2.2	Audience and prerequisites.....	6
3	Z-WAVE™ DEVELOPER'S KIT CONTENTS	7
3.1	Software and documentation.....	8
3.1.1	Adobe	8
3.1.2	CERT_DOC	8
3.1.3	DevKit	8
3.1.4	HW_DOC.....	8
3.1.5	SW_DOC	8
3.2	Network Wide Inclusion.....	9
3.3	Dynamic Route Resolution.....	10
4	FIRST STEPS USING THE DEVELOPER'S KIT	11
4.1	Install the Developer's Kit Software	11
4.2	Create Z-Wave network using ZDK modules.....	11
4.2.1	Development Controller Unit	12
4.2.2	Controller/Slave Unit.....	13
4.2.3	Z-Wave Zniffer Unit	13
4.2.4	Creating a Z-Wave network.....	14
5	Z-WAVE NETWORK NODE TYPES AND LIBRARIES OVERVIEW	15
5.1	Controller Nodes	15
5.1.1	Portable Controller.....	15
5.1.2	Installer Controller	16
5.1.3	Static Controller.....	16
5.1.4	Bridge Controller.....	17
5.2	Slave Nodes	18
5.2.1	Slave.....	18
5.2.2	Routing Slave	18
5.2.3	Enhanced Slave	18
6	DEVELOPER'S KIT SAMPLE CODES	19
6.1	Embedded Sample Applications	19
6.1.1	Binary Sensor	19
6.1.2	Development Controller.....	19
6.1.3	LED Dimmer	20
6.1.4	Door Bell.....	20
6.1.5	Serial API.....	20
6.1.6	MyProduct.....	21
6.1.7	Production Test Software	21
6.2	Z-Wave Security enabled Sample Applications	21
6.3	PC Sample Applications.....	22
6.3.1	Z-Wave PC Controller	22
6.3.2	Z-Wave UPnP Bridge	23
6.3.3	Z-Wave Installer Tool	24
7	DEVELOPER'S KIT TOOLS.....	25
7.1	Z-Wave Zniffer.....	25
7.2	Z-Wave ERTT.....	26
7.3	Z-Wave Programmer.....	27

8	PROGRAMMING A Z-WAVE ASIC.....	28
8.1	KEIL™ Software Development Tool	28
9	REFERENCES.....	29

1 ABBREVIATIONS

Abbreviation	Explanation
ADR	Adaptive Source Routing
API	Application Programming Interface
DRR	Dynamic Route Resolution
DUT	Device Under Test
FLIRS	Frequently Listening Routing Slave
GUI	Graphical User Interface
NWI	Network Wide Inclusion
SIS	SUC Id Server
SUC	Static Update Controller
ZDK	Z-Wave™ Developer's Kit

2 INTRODUCTION

This document describes guidelines on how to get started with the Z-Wave Developer's Kit (ZDK).

2.1 Purpose

The purpose of this document is to provide the reader a step-by-step instruction on how to get started with the ZDK.

2.2 Audience and prerequisites

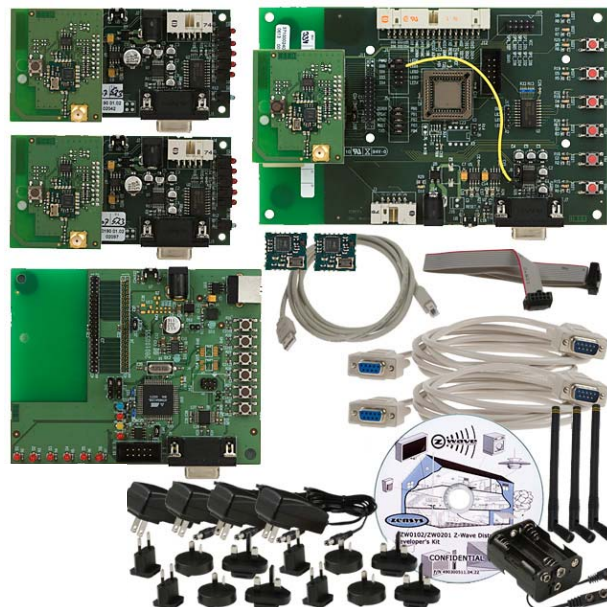
This document is targeted to developers designing Z-Wave enabled products and general interest in learning how to Z-Wave network is being created.

It is recommended to have read the "Z-Wave Node Type Overview and Network Installation Guide" document prior to this document.

3 Z-WAVE™ DEVELOPER'S KIT CONTENTS

Z-Wave Modules

- ZM3102 (2pcs)
- ZM3120C-EW (3pcs)
- Development Module
- Interface Module (2pcs)
- ZDP02A Z-Wave Programmer
- Antenna UHF Flexi (3pcs)
- 9V Power Adapter (4pcs)
- 9V Battery Pack
- Serial Cable RS232 (2pcs)
- USB A-B Cable
- ISP Cable





Z-Wave Technical Service Website

The Z-Wave Technical Service Website is a on-line resource designed to give all Z-Wave Developers access to the latest Z-Wave documentation.

Having purchased a Z-Wave Developers Kit and accepted the Developer's Kit Agreement, you are granted access to the Z-Wave Technical Service website which is an on-line resource designed to provide all Z-Wave Developers access to the latest Z-Wave protocol release and technical documentation.

Note that there is no CD-ROM included in the ZDK. Visit <http://support.zen-sys.com> to download your copy of the ZDK.

For additional kits and/or Z-Wave™ modules contact Zensys distribution partners:

WORLDWIDE DISTRIBUTOR

Digi-Key

701 Brooks Avenue South,
Thief River Falls, MN 56701
USA

Phone: +1-800-344-4539 or +1-218-681-6674

Fax: +1-218-681-3380

Email: rf.support@digkey.com

Web: www.digkey.com/Zensys



GERMAN DISTRIBUTOR

MEV Elektronik Service GmbH

Nordel 5A
D-49176 Hilter a.T.W.

Germany

Phone: +49 (0) 54 24 / 23 40-0

Fax: +49 (0) 54 24 / 23 40-40

Email: info@mev-elektronik.com

Web: <http://www.mev-elektronik.de>



3.1 Software and documentation

Once you have downloaded and unpacked the ZDK to your local hard drive, double click or open the start_here.html file with an internet browser. Start_here.html will provide an overview of the ZDK content.

You can also browse the ZDK content with the file explorer in which you will find the below subfolders of the “data” folder:

3.1.1 Adobe

This folder contains the Adobe® Acrobat Reader installation. All Z-Wave documentation is in Acrobat Reader file format.

3.1.2 CERT_DOC

The CERT_DOC folder contains relevant documentation on the Z-Wave Certification Program.

The certification process is designed to help OEM customers ensure that products have been correctly and robustly implemented and that the product will interoperate with other Z-Wave certified products from the same and other vendors, for the same and other applications. To help the developer ensuring correct implementation of the device and command classes the Compliance Test Tool is available for full and principle Z-Wave Alliance members. Visit <http://z-wavealliance.org> for details.

3.1.3 DevKit

The DevKit folder contains the Z-Wave development software consisting of a protocol part, sample applications and a number of tools used for developing and building of the sample and application code. By choosing to install the development software from the html overview, the contents in this folder will be copied to your local hard disc. The installation will create a new folder on the root of your C-drive and is named “DevKit_4_5x_Beta1”. The same task will also be performed when running the setup.exe file located in “data” folder.

The DevKit folder contains all files to develop a Z-Wave application, with the exception of the required Keil software to compile the source code. The Keil software must be purchased separately and availability is described in the chapter: [Necessary Third Party SW and Tools](#). For detailed description of the development software refer to [1].

Note! To avoid misbehaviour during compile time you must leave the directory structure as created from the installation. This requirement is due to compiler and linker dependencies.

3.1.4 HW_DOC

The HW_DOC folder contains the hardware release note. The folder content is similar to the ‘Hardware documentation’ menu on the html overview. On the Z-Wave Technical Service website, <http://support.zen-sys.com> you will be provided with relevant hardware documentation on Z-Wave modules e.g. datasheets, schematics etc.

3.1.5 SW_DOC

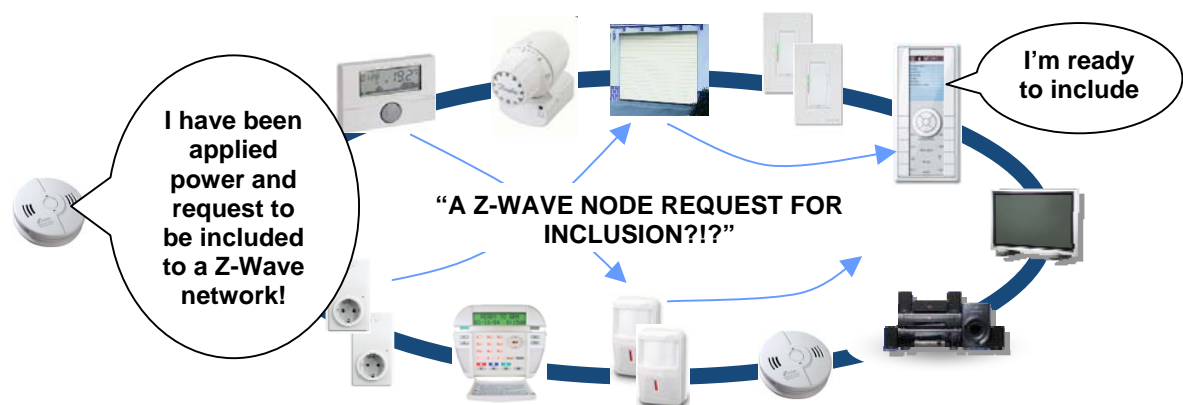
The SW_DOC folder contains relevant software documentation on the Z-Wave technology e.g. Z-Wave API, Application Notes, User guides etc. The folder content is similar to the ‘Software documentation’ menu on the html overview.

3.2 Network Wide Inclusion

With ZDK 4.5x, the Z-Wave technology introduces Plug and Play like network configuration enabling the Z-Wave developer to create very end-user friendly applications and end-products.

Network Wide Inclusion (NWI) enables both end-user friendly, Plug and Play like Z-Wave network installation as well as professional installation scenario where the inclusion process in terms of time will be reduced significantly. NWI is a feature supported by a new frame type named Explorer which enables the Z-Wave protocol to implement Adaptive Source Routing. Refer to [5] for details.

All embedded sample applications provided of the ZDK supports Network Wide Inclusion by verifying not already being included in a network followed by requesting to be included into the network when being applied power.



A new ZDK 4.5x based product can choose to implement auto inclusion when being applied power, and in case other ZDK 4.5x based products are already part of the network these will help the new device to transfer the message to the controller unit. The only user-trigger necessary is at the controller unit where the home owner must set the controller into a learning state. Once the auto inclusion request has been received by the controller the new unit will be assigned the home id of the network and an available node id.



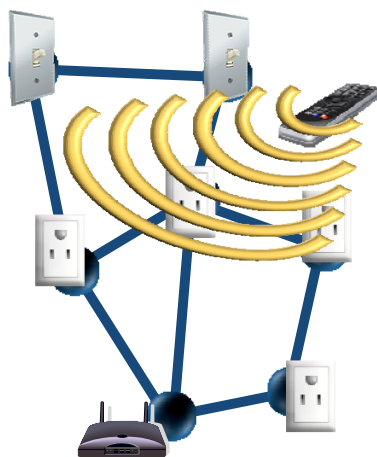
In a larger scenario there could be several new ZDK 4.5x based products that requires being included. This can be handled as including one product i.e. a group of products are being applied power and the controller unit will list the newly included products as these are being assigned the network home id etc.

3.3 Dynamic Route Resolution

With ZDK 4.5x, the Z-Wave technology introduces a true self-healing mesh network through Dynamic Route Resolution supported by the Explorer frame. The Explorer frame enables the Z-Wave protocol to implement Adaptive Source Routing which essentially means that the Explorer will learn its route to the destination through the network.

A simple use case where Dynamic Route Resolution is beneficial includes the Z-Wave enabled Remote Controller (RC) that is mobile and is never at a static position.

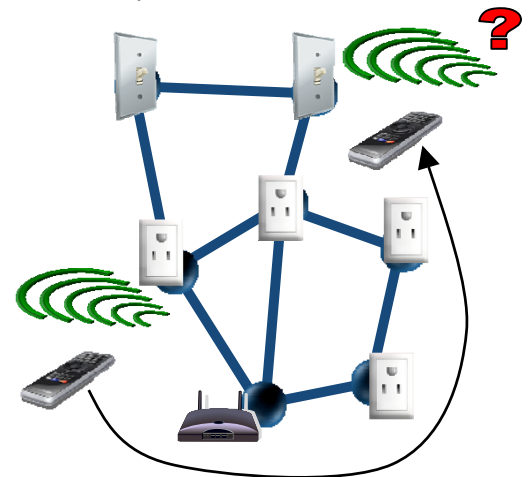
Through legacy Z-Wave protocol algorithm the RC will learn static routes that it will use when communicating with other products in the network. However the static routes may not be valid no longer when the RC is relocated.



When the Explorer frame reaches its destination the target node is provided with the required protocol data to acknowledge the package. Additionally it will carry out the application command immediately since this was included in the Search Request from the RC presenting low latency in terms of user feedback.

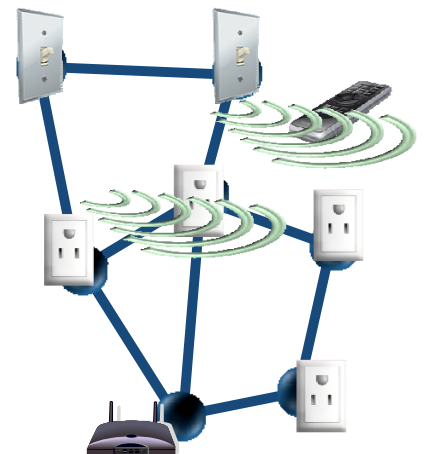
The target node replies with a Search Result command using the route learned during the Search Request command including a Search Stop bit to stop other nodes from forwarding the Search Request.

Dynamic Route Resolution is a ground breaking self-healing management feature that enables the Z-Wave developer to fully trust in the Z-Wave protocol to carry out reliable transportation of the package. This feature allows the Z-Wave developer to leave out complex network management¹ functions and focus on the application layer.



The ZDK 4.5x based RC is now capable of asking other ZDK 4.5x based products to help address the target node. The RC initiates an Explorer frame containing complex protocol information including a Search Request command and the application payload.


When neighbor nodes receive the Explorer frame from the RC they will insert the node id into the protocol part of the frame and forward the message. The source routing mechanism is now adaptive.



¹ Maintenance of static routes is recommended for controller applications. Refer to [1] for details.

4 FIRST STEPS USING THE DEVELOPER'S KIT

4.1 Install the Developer's Kit Software

First you need to install the Developer's Software into your local hard drive by navigating to "Developer's Kit SW" and press the  icon from the html overview of the ZDK content or by running the setup.exe file located in the root of the "data" folder.

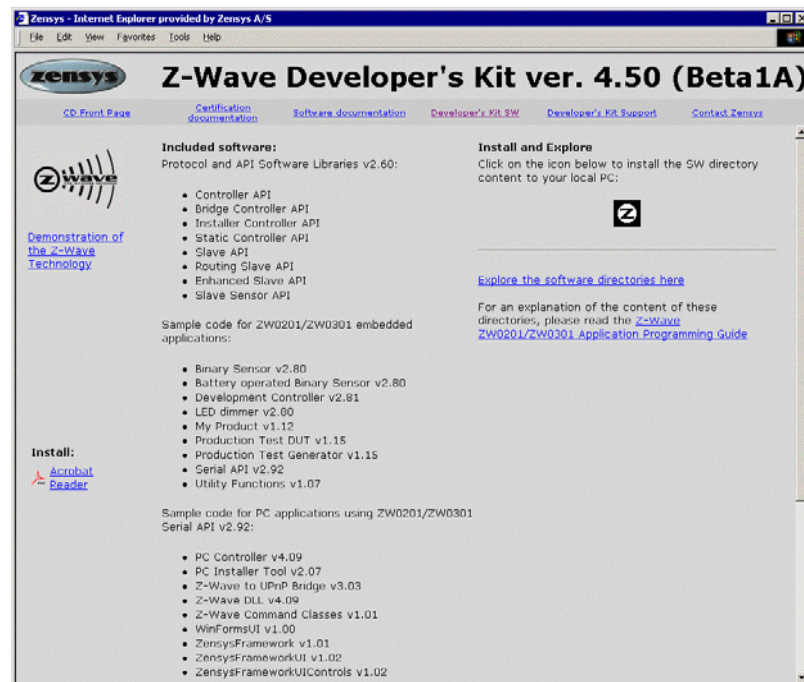


Figure 1, Screenshot of the 'Developer's Kit SW' menu

Please keep in mind that only the Developers Software is copied into your hard drive as described in chapter 3.2.2.

4.2 Create Z-Wave network using ZDK modules

This chapter will take you through the basic steps on how to set-up a Z-Wave network using the Development Controller Unit, Slave Units and the Z-Wave Zniffer Module. With the Z-Wave Zniffer and the Zniffer PC application you will get a picture of how the nodes interact.

The Z-Wave modules comes pre-programmed and with labels for fast and easy identification.

- Development Controller = dev_ctrl sample application
- LED Dimmer = LED_Dimmer sample application
- Z-Wave Zniffer = Zniffer firmware

This allows you to immediately start using and testing the Z-Wave network capabilities. In case the modules are not labelled, then you can distinguish between the modules by the LED(s) on the interface module(s).

When D1 is flashing, then it is a Zniffer module.

When D1, D2 and D3 can be switched ON/OFF and dimmed, then it is a LED Dimmer module.

4.2.1 Development Controller Unit

The Development Controller sample application provided with the ZDK is designed to simulate a Z-Wave enabled remote controller. It consists of a Z-Wave Module mounted on top of the Development Module.

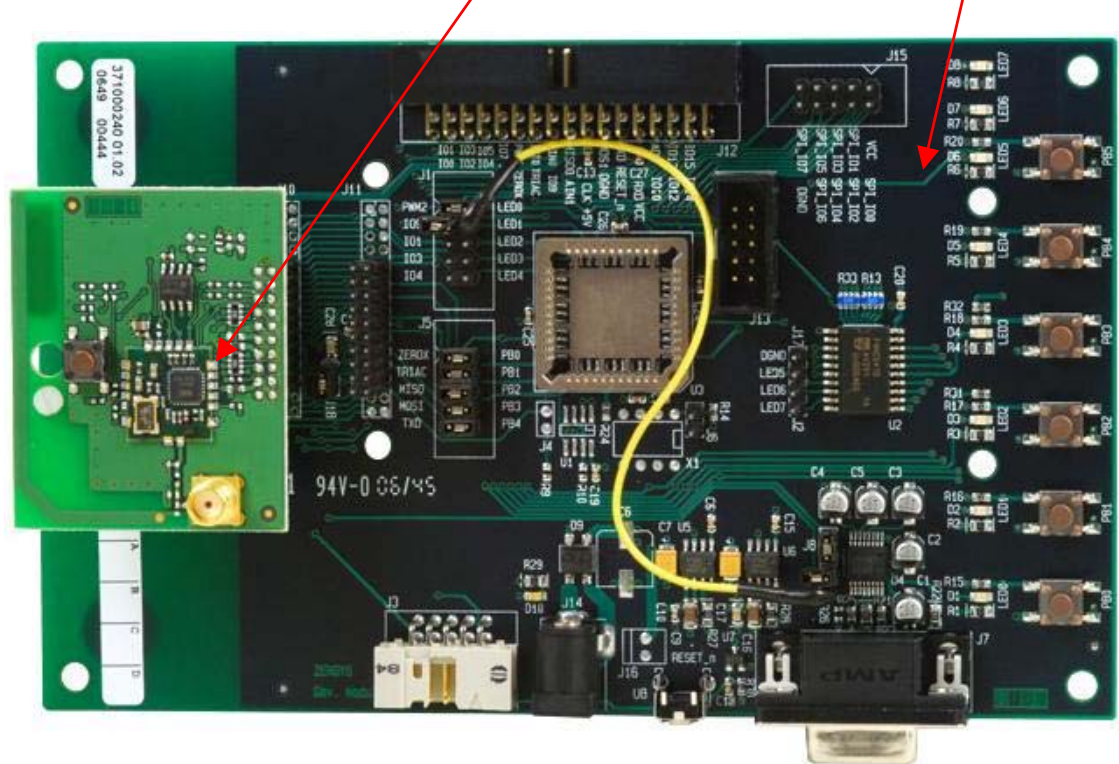


Figure 2, Development Controller Unit

To operate, you need to attach one of the 9 VDC power supplies to the power connector on the Development Module. If you need the Development Controller Unit to be portable, you can use the 9 VDC Battery Pack to power the Unit. Please note that the Development Controller sample application does not power down, and you should therefore re-connect to the AC/DC power adapters whenever you do not need the Development Controller Unit to be portable.

Once you have powered ON the Development Module, then press and hold PB0 and PB4 for two seconds to reset the controller. The correct feedback from the Development Module is LED1 will turn ON for about one second followed by a short flash from LED0. This indication means that the Development Controller Unit is now reset and ready for the next step.

Refer to [4] for more details.

4.2.4 Creating a Z-Wave network

This section will describe how to set-up a simple Z-Wave network using the modules from the ZDK. It is recommended that you prior to this have read the “Z-Wave Node Type Overview and Network Installation Guide”. Please refer to [2]. The following units are required for the network:

1. Prepare the Z-Wave Zniffer
 - a. Install the Z-Wave Zniffer program located in `..\DevKit_4_5x\Tools\Zniffer\PC`
 - b. Connect the Z-Wave Zniffer Unit to the COM port of the PC
 - c. Launch the Z-Wave Zniffer program installed from step 1
2. Verify Z-Wave Zniffer status
 - a. Apply power to the Development Controller Unit
 - b. Verify that there was broadcasted an Explorer Frame containing the Node Information and Auto-Inclusion request² on the Zniffer program
 - c. Interrupt the Auto-Inclusion process by pressing either of the pushbuttons of the Development Controller Unit
3. Include the “LED Dimmer” unit to the Development Controller³
 - a. Apply power to the LED Dimmer and notice the Auto-Inclusion requests on the Zniffer
 - b. Press and hold PB1 on the Dev.Ctrl Unit
 - c. The correct response from the Dev.Ctrl should be a short flash of LED0 indicating the unit has been added to the network.
 - d. Study the trace from the Zniffer program and learn how an inclusion of a slave unit to the network is being handled by the Z-Wave protocol. Notice how the “LED Dimmer” unit is assigned the HomeID of the Dev.Ctrl Unit and the new NodeID.
4. Associate the “LED Dimmer” to a button of the Dev.Ctrl Unit
 - a. Press and hold PB0 on the Dev.Ctrl Unit
 - b. Press three times within 1.5 seconds on the button of the “LED Dimmer” unit to trigger a transmission of the Node Information frame
 - c. Now the “LED Dimmer” unit can be controlled by the Dev.Ctrl Unit. By either single press or press and hold PB0 you should see the “LED Dimmer” either switch ON/OFF or dim down/up respectively.
5. Reset or Remove a “LED Dimmer” unit from the network
 - a. Press and hold PB2 on the Dev.Ctrl Unit
 - b. Press three times within 1.5 seconds on the button of the “LED Dimmer” unit
 - c. The correct response from the Dev.Ctrl should be a short flash of LED0 indicating the unit has been removed from the network.
 - d. Study the trace from the Zniffer program and learn how units are being excluded from a network. Notice how the “LED Dimmer” is assigned a new HomeID and NodeID.

You have now been through the basic steps on how to create a simple Z-Wave network, associating a unit to a controller device and how to reset a slave unit. It is recommended that you do some more attempts on how to include and exclude nodes into a Z-Wave network to get familiar with the protocol.

² All embedded sample applications included in ZDK 4.5x will request for inclusion when being applied power after having verified not already being part of a network.

³ In order for a slave application to be included into a network it must not be part of another network i.e. home id. Perform step 5 to reset/remove a device from a network.

5 Z-WAVE NETWORK NODE TYPES AND LIBRARIES OVERVIEW

The Z-Wave network consists of two different types of network nodes; controllers and slaves. The controller nodes are able to calculate routes (and alternative routes). The second node type is the slave node, which generally acts as input and output units. Both types exist in different versions as described below. The Z-Wave protocol supports networks of up to 232 nodes, which can be freely shared between controller and slave nodes.

5.1 Controller Nodes

Z-Wave networks are established around a controller. The controller used to include the first node is by default configured to act as Primary Controller with the capability to include/exclude nodes. The Primary Controller is used to include all subsequent nodes in the network including other controller nodes.

A Static Controller can be enabled to become a Static Update Controller (SUC), which adds advanced network management functionalities. A SUC can furthermore be enabled to become a SUC Id Server (SIS), which adds more flexibility to the installation process. The SIS is by default a Primary Controller because it can include/exclude nodes. Furthermore it enables other controller to become Inclusion Controller enabling these to include/exclude nodes on behalf of the SIS.

The common functionalities for all controller nodes include:

- Is by default the Primary Controller or becoming a Secondary Controller when added to a network
- Can become an Inclusion Controller if a SIS node is present in network
- Has the ability to activate the SUC/SIS role of a Static Controller having enabled this network role
- Has a full network topology awareness

The following chapters provide an overview of the available node types and libraries of ZDK 4.5x including a short feature list. Refer to [2] for general description of Z-Wave node types and to [1] for full library capabilities and detailed API functions description of the specific library.

5.1.1 Portable Controller

The Portable Controller has the ability to discover its own position in the network, when it needs to communicate with other nodes. An example of a device using this type could be a remote control unit, e.g. for controlling light or HVAC systems. Because the Portable Controller can be carried around in the network, it is also typically used to include/exclude nodes and maintaining the Z-Wave network. Portable controllers are typically battery powered.

Portable Controller		
Z-Wave Library:	<ul style="list-style-type: none">➤ ZW_controller_portable_ZW020xs.lib➤ ZW_controller_portable_ZW030xs.lib	
Protocol Features:	Supported:	<ul style="list-style-type: none">➤ Network Wide Inclusion➤ Dynamic Route Resolution➤ Issue Wake Up Beams and calculation of routes to FLiRS nodes➤ Create manual routing
	NOT supported:	<ul style="list-style-type: none">- SUC/SIS functionality- Repeater functionality
Typical applications:	<ul style="list-style-type: none">➤ AV Remote Controller➤ Scene Controller➤ Battery operated actuator	

5.1.2 Installer Controller

The Installer Controller is essentially a Portable Controller node, which incorporates extra functionality that can be used to implement professional installer tools, which need extended network diagnostics. Like the Portable Controller, Installer Controller is typically also battery powered.

Installer Controller		
Z-Wave Library:	<ul style="list-style-type: none">➤ ZW_controller_installer_ZW020xs.lib➤ ZW_controller_installer_ZW030xs.lib	
Protocol Features:	Supported ⁴ :	<ul style="list-style-type: none">➤ Retrieve detailed network information➤ Restore home id and node id➤ Store node information frame of a specific node
Typical applications:	<ul style="list-style-type: none">➤ Handheld Z-Wave Installer tool➤ Professional Installer applications connected to Serial API	

5.1.3 Static Controller

The Static Controller is typically in a fixed position in the network, meaning that it should not be physically moved when it has been included in the network. The “always listening” advantage of the Static Controller allows other nodes to transmit frames to it whenever needed, both for uploading purposes as well as for consulting purposes.

The main difference between Static and Portable Controller is the routing algorithm used. Since a Portable Controller is a mobile device it will always attempt to address the target by direct range. A Static Controller will only use direct range communication if the target node in fact is a neighbor node.

The Static Controller comes in variants supporting different applications requiring specific needs. The capabilities of each Static Controller library are listed below:

Static Controller with NWI, DRR and Wake Up Beam capabilities		
Z-Wave Library:	<ul style="list-style-type: none">➤ ZW_controller_static_nosuc_norep_ZW020xs.lib➤ ZW_controller_static_nosuc_norep_ZW030xs.lib	
Protocol Features:	Supported	<ul style="list-style-type: none">➤ Network Wide Inclusion Center➤ Dynamic Route Resolution➤ Issue Wake Up Beams and calculation of routes to FLIRS nodes➤ Create manual routing
	<i>NOT supported</i>	<ul style="list-style-type: none">- <i>SUC/SIS functionality</i>- <i>Repeater functionality</i>
Typical applications:	<ul style="list-style-type: none">➤ NWI Gateway addressing Frequently Listening battery operated devices	

⁴ In addition to the supported features from a Portable Controller

Static Controller with NWI, DRR and SUC/SIS capabilities		
Z-Wave Library:	➤ ZW_controller_static_norep_nomr_ZW020xs.lib ➤ ZW_controller_static_norep_nomr_ZW030xs.lib	
Protocol Features:	Supported	➤ Network Wide Inclusion Center ➤ Dynamic Route Resolution ➤ SUC/SIS functionality
	<i>NOT supported</i>	- Repeater functionality - Issue Wake Up Beams and calculation of routes to FLiRS nodes - Create manual routing
Typical applications:	➤ NWI Gateway applications associated with multiple remote controller devices	

Static Controller with NWI, DRR and REPEATER capabilities		
Z-Wave Library:	➤ ZW_controller_static_nosuc_ZW020xs.lib ➤ ZW_controller_static_nosuc_ZW030xs.lib	
Protocol Features:	Supported	➤ Network Wide Inclusion Center ➤ Dynamic Route Resolution ➤ Repeater functionality ➤ Create manual routing
	<i>NOT supported</i>	- SUC/SIS functionality - Issue Wake Up Beams and calculation of routes to FLiRS nodes
Typical applications:	➤ Applications to be located in a central position e.g. generic display receiving reports from sensor nodes	

5.1.4 Bridge Controller

The Bridge Controller is essentially a Static Controller node, which has the additional capability of representing devices from other network types like X10 or TCP/IP as virtual nodes in a Z-Wave network. This enables control of Z-Wave nodes from e.g. an X10 controller or vice versa.

Bridge Controller		
Z-Wave Library:	➤ ZW_controller_static_nosuc_norep_ZW020xs.lib ➤ ZW_controller_static_nosuc_norep_ZW030xs.lib	
Protocol Features:	Supported	➤ Network Wide Inclusion Center ➤ Dynamic Route Resolution ➤ Create up to 128 virtual nodes ➤ Create manual routing
	<i>NOT supported</i>	- SUC/SIS functionality - Repeater functionality - Issue Wake Up Beams and calculation of routes to FLiRS nodes
Typical applications:	➤ NWI Gateway addressing Frequently Listening battery operated devices	

5.2 Slave Nodes

The slave nodes are devices that do not contain routing tables. The so-called Routing Slave nodes and Enhanced Routing Slave nodes however can contain a number of pre-configured routes (assigned to them by a controller).

Any slave node can act as repeater for frames going to other nodes. The only requirement for being able to act as repeater is that the node is in listening state. This requires that the node is permanently powered, and in order to limit battery consumption, this means that only mains-powered nodes will act as repeaters in most practical installations.

Battery operated slaves that do not listen continually are disregarded by controllers when they calculate routes. Battery operated slaves must therefore set the node in non-listening state.

5.2.1 Slave

The Slave node type is able to receive frames and reply if necessary. The slave node cannot host pre-configured routes to other nodes. The slave node is typically used for devices that only require input (and report status if polled) and do not generate frames unsolicited. An example of a device using this type could be a power outlet.

5.2.2 Routing Slave

The Routing Slave can host a number of routes for reaching other slaves or controllers. Such routes are called "Return Routes"⁵. Routing Slaves can use these routes to communicate with either controllers or other slave nodes. The Routing Slave can either be A/C powered or battery powered. If the Routing Slave is A/C powered it is used as a repeater in the Z-Wave network, otherwise it will be disregarded when routes are calculated. The Routing Slave functionality is used for devices that need to report unsolicited status or alarms.

An example of a routing slave node could be a Passive Infra Red (PIR) movement sensor. A wall switch might also be a routing slave and could then be used to control small lighting scenes, or to establish a kind of "virtual" 3-way switching.

A special case of a battery powered routing slave is the Frequently Listening Routing Slave (FLiRS). This is a routing slave configured to listen for a wakeup beam in every wake-up interval. This enables other nodes to wake up the FLiRS node and send a message to it.

One example of a FLiRS node is as chime node in a wireless doorbell system or a door lock system is battery operated but requiring real time operation.

5.2.3 Enhanced Slave

The Enhanced Slave has the same basic functionality as a routing slave node, but more software components are available because of more features on the hardware. Enhanced slave nodes have software support for External EEPROM.

An example of a device using this type of Slave nodes could be a Thermostat.

⁵ "Return Route" is a controller-centric term that was originally referring to a route going back to the controller. Seen from the routing slave, "Controller Assigned Route" might be a better term. However, "Return Route" is the established term in all Z-Wave documentation.

6 DEVELOPER'S KIT SAMPLE CODES

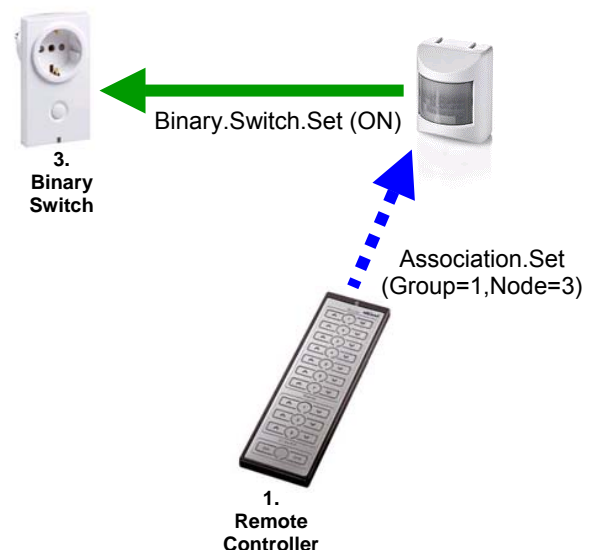
6.1 Embedded Sample Applications

The Developer's Kit contains sample codes to help the Z-Wave Developer getting started with developing a Z-Wave application. The source codes, make files and µVision project files are located in the appropriate project folders in ..\DevKit_4_5\Product\. The following chapters provide a short introduction of each sample applications. For details of all applications refer to [1].

6.1.1 Binary Sensor

The Binary Sensor sample applications can result in two different products: AC operated Binary Sensor and DC operated Binary Sensor. This device is in effect a binary sensor where the sensor input is the pin also used as a button input on the device module.

In addition the push-button will simulate a detected movement to trigger a control command transmitted to the associated target device i.e. this sample application includes an example of how to set-up an association between two Z-Wave nodes with the help from a controller application. Further the DC operated Binary Sensor includes the "I'm Lost" procedure which is a protocol function to ask other nodes in the network to help find the Static Update Controller for updates due to target nodes have been relocated.



6.1.2 Development Controller

The Development Controller sample application is designed for the Development Controller Unit, which is an assembly of the Development Module and the Z-Wave Module. The application user interface is based on 5 pushbuttons and 2 LED's on the ZW010x Development Module. Refer to [4] for user guide.

The Development Controller sample application shows the following features of the Z-Wave protocol:

- Include Slave and Controller Nodes to the Z-Wave Network
- Associate a Node to a local group
- Switching a Group of Nodes on or off
- Dimming a Group of multilevel switches (e.g. LED Dimmers)
- Assigning a Route to a routing slave (e.g. a Binary Sensor)
- Excluding a Node from the Z-Wave Network
- Resetting the Development Controller
- Requesting network updates to a Secondary Controller from a Static Update Controller (SUC) in case it is present in the network
- Work as Inclusion Controller when a SUC ID Server (SIS) is present in the network

6.1.3 LED Dimmer

The LED Dimmer sample source code is built for a slave library application on a Z-Wave module, which uses three LED's on the interface module to simulate a light switch with a built in dimmer. The LED Dimmer sample application is typically the base for simple appliance modules.

The LED Dimmer advertises support for the following command classes via the node information frame:

- Switch Multilevel Command Class
- Switch All Command Class
- Manufacturer Specific Command Class
- Version Command Class
- Protection Command Class
- Powerlevel Command Class



6.1.4 Door Bell

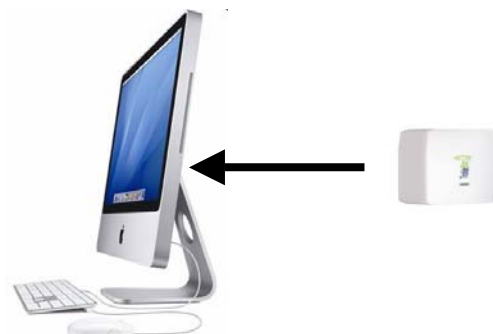
The ZDK contains sample code for a Door Bell sample application. This device is an example of how a battery operated chime in a door bell system could be build. The Door Bell is based on the Routing Slave Library and uses the Frequently Listening Routing Slave (FLiRS) mode where it powers up the radio for a short period every 1000ms and if it receives a command it will power up entirely and turn on the LED's.

The Door Bell can be included to the network by using e.g. the PC Controller application. Once included, the Door Bell will enter sleep mode and wake up every 1000ms. The protocol will know in advance that the destination node is in frequently listening mode, when the PC Controller request to communicate with the Door Bell, and will transmit a Wake-Up Beam, prior to e.g. a Basic.Set command. The Wake-Up Beam will trigger the Door Bell to power up and execute the received frame if supported.

6.1.5 Serial API

The ZDK includes Serial API embedded applications linked to all available library types of the ZDK. The Serial API presents the enabled embedded API functions to the UART interface. The host application can vary but is typically a PC or WEB application running on a Linux environment connected to either the Z-Wave module implemented on board or through a USB dongle as depicted below. The following host based PC applications are available on the ZDK:

- PC based Controller application to be connected with a Serial API application based on a static controller library.
- PC based Installer Tool application to be connected with a Serial API application based on an installer controller library.
- PC based Z-Wave Bridge application to be connected with a Serial API based on a bridge controller library.



The Serial API can be used as provided with the ZDK or it can be modified to fit specific needs. The UART on the Z-Wave Module is initialized for 115200 baud, no parity, 8 data bits and 1 stop bit.

6.1.6 MyProduct

To help you getting started with your Z-Wave application the ZDK includes the MyProduct project files. MyProduct contains the minimum framework to begin developing a slave application.

6.1.7 Production Test Software

6.1.7.1 Production Test DUT

During production testing the Device Under Test (DUT) sample code can be loaded into the product sample for semi-auto testing. The Production Test DUT sample application is based on the ZW_slave_proctest_dut library and has two functions that can be used during production test:

- The radio will start to transmit continuously if the P1.5 (SS_N) pin on the ZM2102/ZM3102 is pulled low during power up.
- If the pin isn't pulled low the radio will go into receive mode and send acknowledge to all frames sent to the module.

6.1.7.2 Production Test Generator

The Developer's Kit contains sample code that demonstrates 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 to verify the TX / RX circuits on Z-Wave enabled products.

The generator consists of an Interface Module and a ZM3120 Z-Wave Module to which a push-button and two LED's, a red LED and a green LED are connected. After connection to power, the red LED will be on. The green LED is the LED position 'D6' on the Interface Module and the red LED is the LED position 'D1'. The push button is the 'S1' push button on the ZM3120 Z-Wave Module.

When the button is pressed, 10 NOP frames will be transmitted through the RF-connector of the generator. The DUT is expected to verify the reception of each NOP with an ACK. During transmission, the red LED will blink.

If all NOP frames are acknowledge by the DUT, the red LED will turn off and the green LED will turn on and stay on until the next test is conducted. If the DUT does not reply correct, the red LED will stop blinking and turn on.

6.2 Z-Wave Security enabled Sample Applications

ZDK 4.5x includes the following sample applications enabled with the Z-Wave Security framework:

- Secure Binary Sensor
- Secure Development Controller
- Secure Door Lock
- Secure LED Dimmer

The Z-Wave Security framework for ZW0301 based applications consist of a software implementation where the encryption/decryption cipher block must reside in a host processor for controller applications such as the Secure Development Controller. Refer to [1] for details.

NOTICE: Binaries and AES128 encryption/decryption engine are not distributed on due to export restrictions. Contact support via support@zen-sys.com for further information.

6.3 PC Sample Applications

The ZDK contains three PC sample applications utilizing the Z-Wave DLL presenting an API to the Z-Wave protocol for PC based applications. The Z-Wave DLL uses the Microsoft .NET Framework which provides a stable and reliable operation of the DLLs. Refer to [15] for details.

6.3.1 Z-Wave PC Controller

The PC Controller Application is an example of a controller, where the application code is running on a PC using Microsoft .NET Framework, and not on the Z-Wave Module itself. The Z-Wave Module facilitates the communication between the PC Controller application and the other Z-Wave enabled nodes in the network. To run the PC Controller Application, the PC must be connected via a serial cable to a Z-Wave Module running a Serial API Static Controller application.

You can use the Z-Wave PC Controller application to configure the Z-Wave Network using Network Wide Inclusion. For detailed description refer to [7].

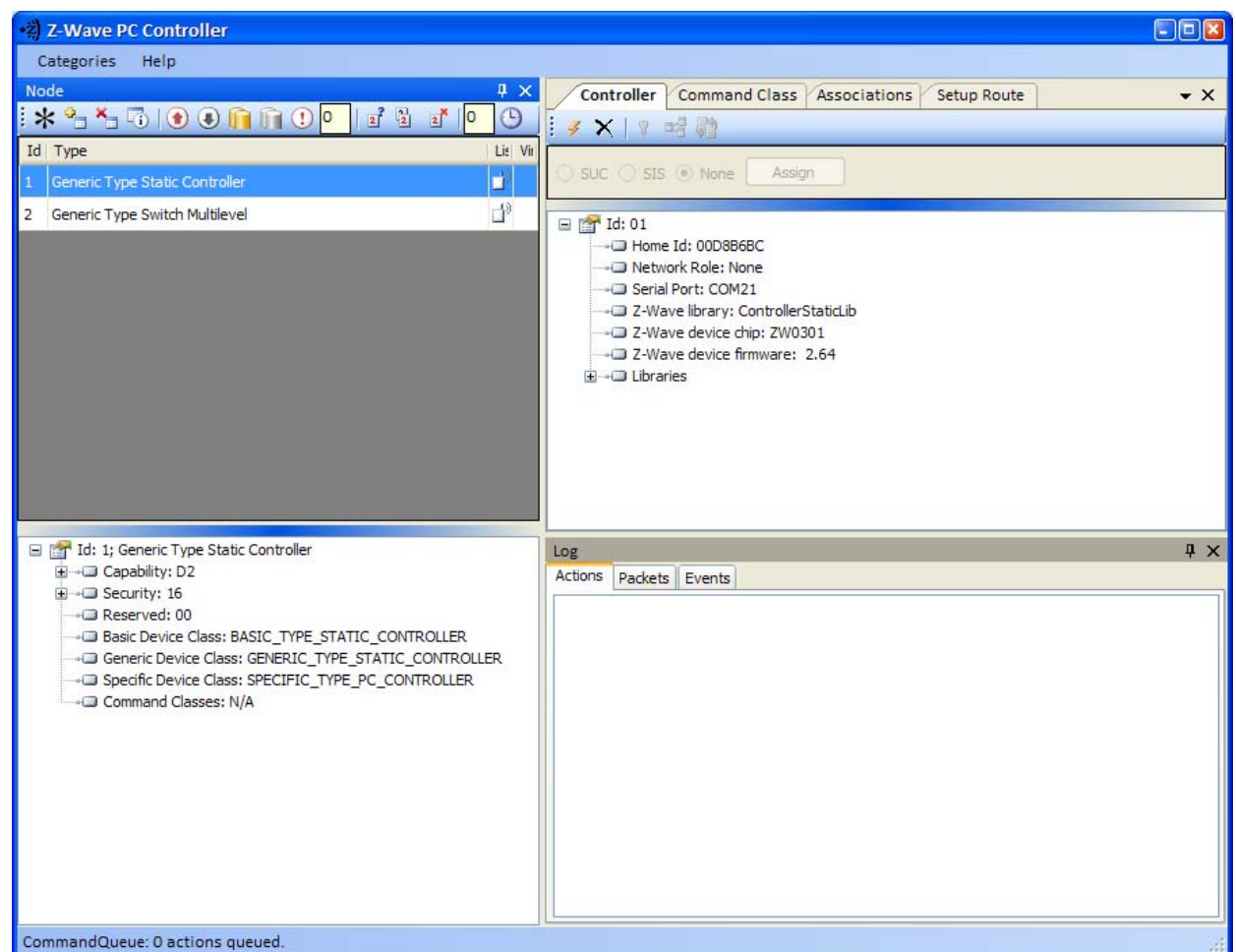


Figure 4, Screenshot of Z-Wave PC Controller

6.3.2 Z-Wave UPnP Bridge

The Z-Wave UPnP Bridge application is an example on how features of the Bridge Controller API can be used to implement a Z-Wave to UPnP Bridge. To run the application it is required that the PC is connected via a serial cable to a Z-Wave Module running the Serial Bridge API SW. Refer to [8] for more details.

The features of the Z-Wave UPnP Bridge application are:

- Adding and resetting nodes.
- Adding and removing Virtual Z-Wave Slave nodes (up to 128 virtual slave nodes)
- Sending Basic SET ON/OFF and GET commands to known Z-Wave nodes.
- Bridging a Z-Wave Switch node to UPnP as an UPnP BinaryLight.
- Bridging an AV Rendered device to a Z-Wave network as a Z-Wave Switch node.

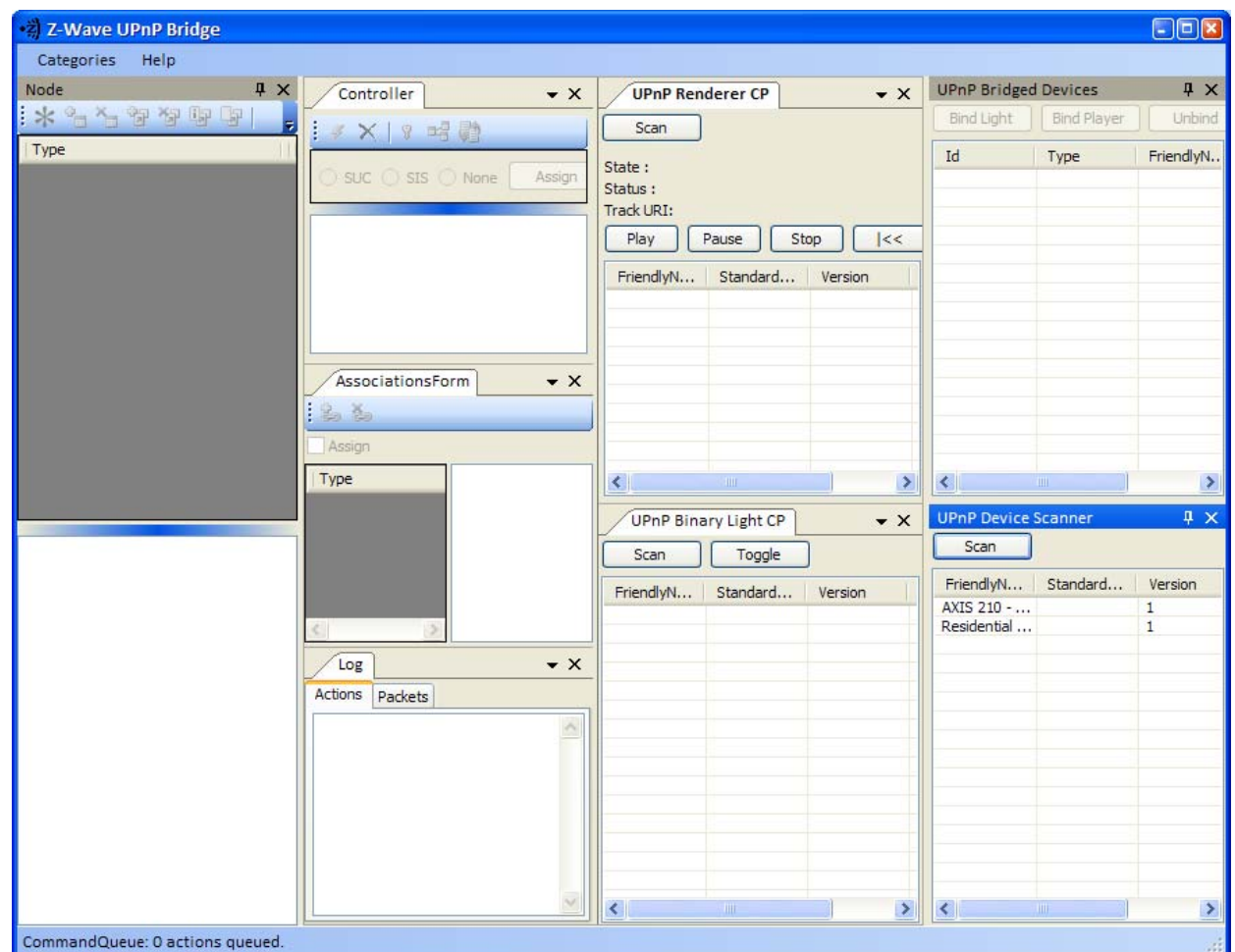


Figure 5, Screenshot of Z-Wave UPnP Bridge

6.3.3 Z-Wave Installer Tool

The Z-Wave Installer Tool application is a PC sample application demonstrating the possibilities of using the features available in the Serial API Installer Tool sample code. These features include the possibility of having the entire Network Topology presented on a Graphical User Interface and checking the RF communication quality between two nodes of your own choice. To run the application it is required that the PC is connected via a serial cable to a Z-Wave Module running the Serial API Installer SW.

The features of the Installer Tool are:

- Adding and resetting nodes
- Testing link quality between nodes
- Rediscovering of network
- Displaying network topology
- Set the Static Controller to automatic Update

You can either use the Installer Tool to set-up your entire Network, or you can copy the Network information from e.g. a Development Controller. To learn more about how to use the Z-Wave Installer Tool application refer to [12].

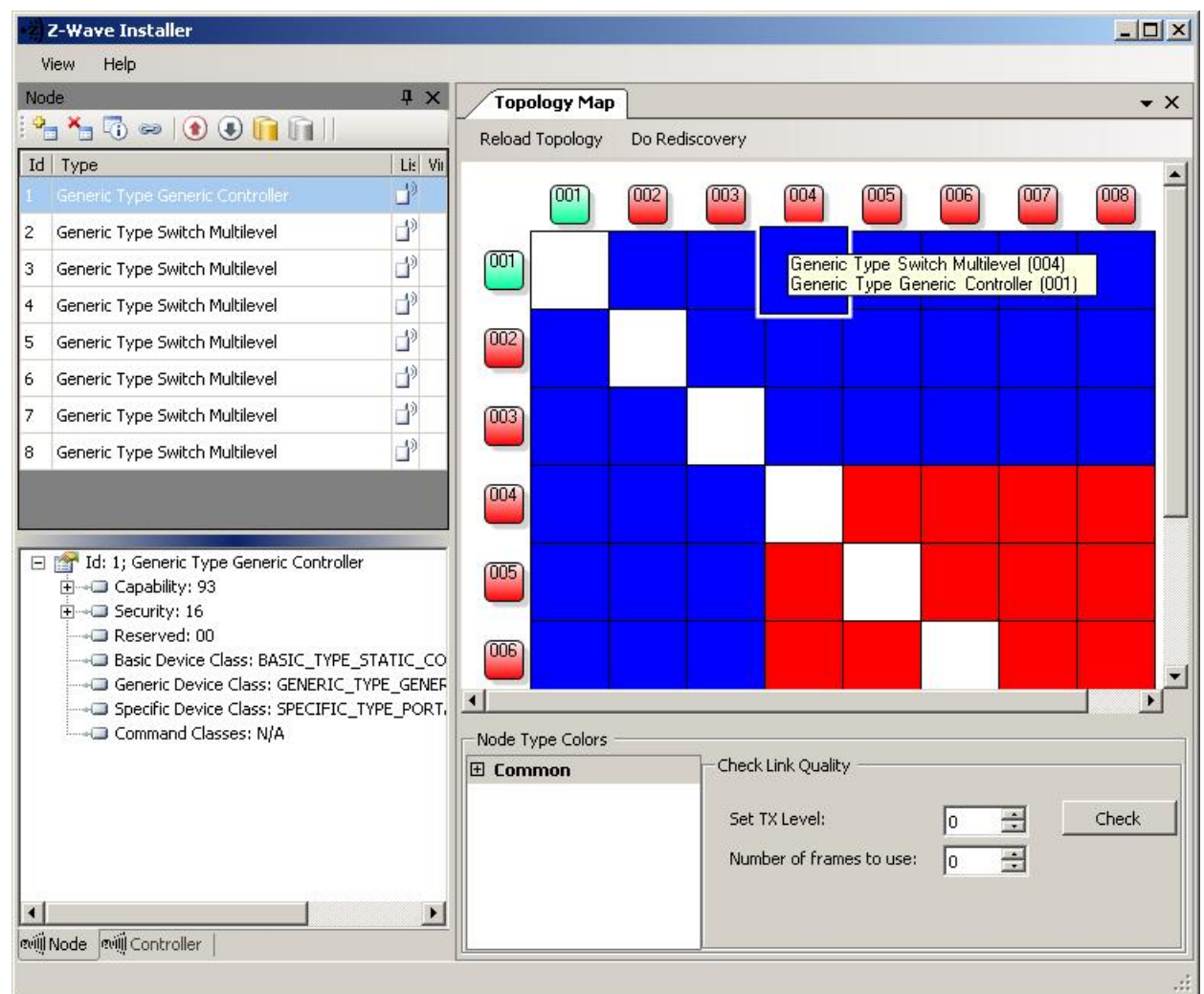


Figure 6, Screenshot of Z-Wave Installer Tool

7 DEVELOPER'S KIT TOOLS

7.1 Z-Wave Zniffer

The Z-Wave Zniffer is a development tool for capturing Z-Wave RF traffic and presenting the frames in a graphical user interface on a PC. The Zniffer allows you to monitor and record all communication that takes place between the Nodes in a Z-Wave Network. You can investigate the ID of the Source and Destination for the communication, the type of Frame being sent, and the Application Content, i.e. the specific command, which is being sent.

The Zniffer tool is a passive “listener” to the Z-Wave Network traffic, and will only display the RF communications taking place within direct RF range. Refer to [9] for additional information about the Zniffer tool.

LineNo	Date	Time	Speed	Delta	Source	Dest	HomeID	Data	Application	HexData
653	13-03-2009	04:06:27.688	40Kbit/s	7	003	001	C29BBEDE	LTX.Ack		C2 9B BE DE 03 23 07 0A 01 EA
654	13-03-2009	04:06:27.735	40Kbit/s	32	001	003	C29BBEDE	LTX.SingleCast	Basic Set	C2 9B BE DE 01 61 08 0D 03 20 01 00
655	13-03-2009	04:06:27.735	40Kbit/s	7	003	001	C29BBEDE	LTX.Ack		C2 9B BE DE 03 23 08 0A 01 E5
656	13-03-2009	04:06:27.735	40Kbit/s	34	001	003	C29BBEDE	LTX.SingleCast	Basic Set	C2 9B BE DE 01 61 09 0D 03 20 01 00
657	13-03-2009	04:06:27.798	40Kbit/s	6	003	001	C29BBEDE	LTX.Ack		C2 9B BE DE 03 23 09 0A 01 E4
658	13-03-2009	04:06:27.813	40Kbit/s	35	001	003	C29BBEDE	LTX.SingleCast	Basic Set	C2 9B BE DE 01 61 0A 0D 03 20 01 00
659	13-03-2009	04:06:27.845	40Kbit/s	7	003	001	C29BBEDE	LTX.Ack		C2 9B BE DE 03 23 0A 0A 01 E7
660	13-03-2009	04:06:27.860	40Kbit/s	35	001	003	C29BBEDE	LTX.SingleCast	Basic Set	C2 9B BE DE 01 61 0B 0D 03 20 01 00
661	13-03-2009	04:06:27.860	40Kbit/s	7	003	001	C29BBEDE	LTX.Ack		C2 9B BE DE 03 23 0B 0A 01 E6
662	13-03-2009	04:06:27.907	40Kbit/s	32	001	003	C29BBEDE	LTX.SingleCast	Basic Set	C2 9B BE DE 01 61 0C 0D 03 20 01 00
663	13-03-2009	04:06:27.907	40Kbit/s	7	003	001	C29BBEDE	LTX.Ack		C2 9B BE DE 03 23 0C 0A 01 E1

Frame Details

HomeID: C29BBEDE SrcNodeID: 001 DstNodeID: 003
Header Type: WakeUp Beam Speed: none SingleCast Request ACK Sequence Number: 8 LTX Length: 13 bytes
Application Layer: Basic Set
Value: 00

C2 9B BE DE 01 61 08 0D 03 20 01 00 81

Frames: 962 Line: 654 Selected Count: 1 Timespan: 0 ms Unfiltered Port: COM1 Frequency: 868.42MHz (EU)

Figure 7, Screenshot of Z-Wave Zniffer

7.2 Z-Wave ERTT

The Z-Wave Enhanced Reliability Test Tool (ERTT) is used to measure the performance of the Z-Wave modules. It sends packets to a Device Under Test (DUT), which acknowledges the reception back to the ERTT. The ERTT counts the amount of acknowledgements received and compares it to the amount of packages sent. The ERTT has to be connected to a Z-Wave module loaded with a special version of the serial API application code and is a valuable tool for the Z-Wave Certification program as well as general RF range testing. For detailed information about the ERTT please refer to [10].

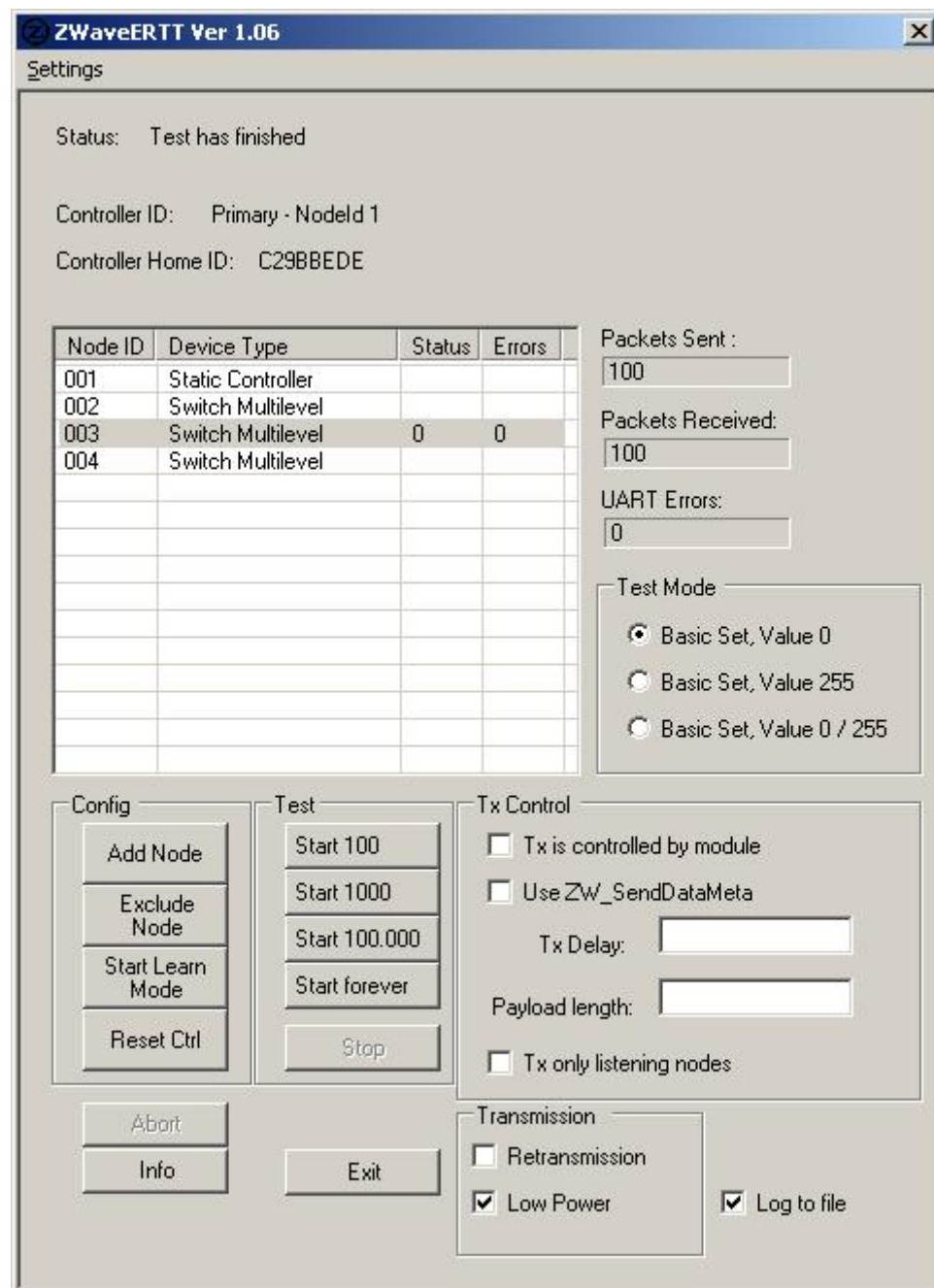


Figure 8, Screenshot of Z-Wave Enhanced Reliability Test Tool

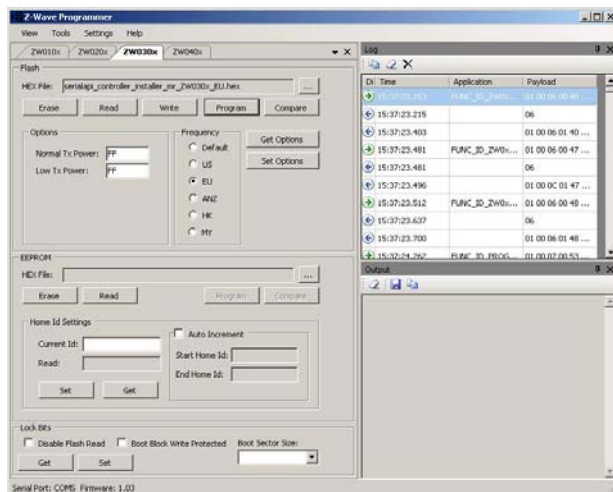
7.3 Z-Wave Programmer

Zensys has developed its own USB based Z-Wave ASIC Programmer that will significantly reduce the programming time and increase the ease of use compared to the EQUINOX, EPSILON5 programmer included on older Developer's Kit revisions.

The Z-Wave Programmer can besides programming the Z-Wave ASIC's, ZW0102, ZW0201, ZW0301 in the Z-Wave modules, also directly program the external EEPROM in e.g. ZM3120. In addition the GUI can configure the RF output power of the ASIC, which formerly was handled by the ZetupRF tool. Please refer to [13] for detailed information.

The programming set-up using the Z-Wave Programmer consists of:

- The Z-Wave Programmer HW.
- A Z-Wave module mounted on top of the programmer in the module socket (alternatively connected via the ISP cable).
- A PC running the Z-Wave Programmer GUI on windows XP.
- USB cable.
- ISP Cable
- One power supply connected to the programmer.



The Z-Wave Programmer GUI supports the following features:

- Write/verify HEX files contents to the flash
- Erasing the flash
- Read the flash contents to a file.
- Compare the flash contents with a hex file.
- Write the RF settings to the flash
- Write HEX files contents to the EEPROM
- Clear the EEPROM
- Read the EEPROM contents to a file.
- Compare the EEPROM contents with a file
- Write home ID to the EEPROM
- Program the Lock Bits
- Prepared for ZW0401 based modules

8 PROGRAMMING A Z-WAVE ASIC

When you require downloading new application to the Z-Wave ASIC any of the Z-Wave modules, there are three main steps to execute:

1. First you need to compile and link your application using KEIL compiler to create the HEX file.
2. If you are creating a controller application, you must initialise and program the external EEPROM with a unique home id using the Z-Wave Programmer. This action is only necessary once. If the application is based on ZDK 4.5x, programming the home id is optional as this Z-Wave release will random generate a home id whenever the controller application is set back to default.
3. The last step is to program the Z-Wave ASIC using the Z-Wave Programmer.

8.1 KEIL™ Software Development Tool

The KEIL™ Software development tool for 8051 micro controllers is a third party software tool that is required to work with the ZDK SW and is not supplied with the ZDK . **The ZDK uses the PK51 kit.**

Refer to the [2] chapter 8, for details about which revision of the Assembler, C compiler, Linker and Librarian that you should use to assure seamless integration with the Z-Wave protocol APIs.

The KEIL™ Developer's Kits can be purchased either through Zensys or directly from KEIL™ or from one of their local distributors. Visit <http://www.keil.com> for details.

To configure the KEIL™ µVision3 IDE for development of Z-Wave applications refer to [3] or use the pre-defined µVision project files provided in the ZDK.



KEIL Software Inc.

1501 10th Street, Suite 110
Plano, TX 75074
USA
Toll Free: +1 800-348-8051
Phone: +1 972-312-1107
Fax: +1 972-312-1159
Sales: sales.us@keil.com

KEIL Elektronik GmbH

Bretonischer Ring 15
D-85630 Grasbrunn
Germany
Phone: +49 089 45 60 40 0
Fax: +49 089 46 81 62
Sales: sales.intl@keil.com
Support: support.intl@keil.com

9 REFERENCES

- [1] Zensys, INS11095, Instruction, Z-Wave ZW0201/ZW0301 Appl. Prg. Guide 4.50
- [2] Zensys, INS10244, Instruction, Z-Wave Node Type Overview and Network Installation Guide
- [3] Zensys, INS10246, Instruction, Keil uVision3 IDE User Guide
- [4] Zensys, INS10236, Instruction, Development Controller User Guide
- [5] Zensys, SDS10243, Specification, Z-Wave Protocol Overview
- [6] Zensys, SDS10242, Specification, Z-Wave Device Class Specification
- [7] Zensys, INS10240, Instruction, PC based Controller User Guide
- [8] Zensys, INS10245, Instruction, Z-Wave Bridge User's Manual
- [9] Zensys, INS10249, Instruction, Z-Wave Zniffer User Guide
- [10] Zensys, INS10336, Instruction, Z-Wave Reliability Test Guideline
- [11] Zensys, INS10637, Instruction, Z-Wave Certification form
- [12] Zensys, INS10241, Instruction, Z-Wave PC Installer Tool Application User Guide
- [13] Zensys, INS10679, Instruction, Z-Wave Programmer User Guide
- [14] Zensys, SDS11060, Specification, Z-Wave Command Class Specification
- [15] Zensys, INS10250, Instruction, Z-Wave DLL User Guide