Instruction

Getting started with Z-Wave®

Document No.:	INS11469
Version:	4
Description:	This document covers a basic introduction of the Z-Wave Home Control Kit
Written By:	CHL;JFR;CST;AIK HONG
Date:	2011-06-01
Reviewed By:	CHL;CST;JFR;PHU
Restrictions:	Public

Approved by:

Date	CET	Initials	Name	Justification
2011-06-01	13:57:40	CHL	Chong Li	on behalf of NTJ

REVISION RECORD				
Doc. Rev	**Date**	**By**	**Pages affected**	**Brief description of changes**
1	20100310	CHL	All	Initial draft
2	20100708	CHL JFR	4.4.2.3 & 5.2	Changed names of development mode hex files
3	20110126	JFR	4.4.2.3	Indicated pin 1 position on socket correct.
4	20110601	AIK		Optimized flow of getting started instructions

Table of Contents

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
OTP	One Time Programmable
SIS	SUC Id Server
SUC	Static Update Controller
SDK	Software Developer's Kit

2 INTRODUCTION

The Z-Wave® Home Control Kit enables developers and Original Equipment Manufacturers (OEMs) to design, and develop products within the Home Control segment to network wirelessly using the RF based Z-Wave Technology.

The Z-Wave Home Control Kit is based on Sigma Designs' Z-Wave 300 and 400 series based modules, a mixed signal chip integrating RF transceiver, memory, IR generation and learning, AES Security Engine and a MCU capable of handling both the Z-Wave Protocol Stack and the OEM's application software storage in one single chip/module. The Z-Wave Protocol Stack and Z-Wave Modules deliver low-cost, low power consumption, wireless networking solution targeted for home control, home security & monitoring and home energy management.

The Z-Wave Home Control Kit contains all the software and detailed documentation necessary to design and write OEM application software on top of the Z-Wave Protocol API, as well as to test and debug the final Z-Wave product. In addition, this kit contains Z-Wave ZM4101, ZM4102 and ZM3102 modules that can directly be integrated into an OEM's product.

This document describes guidelines on how to get started with the Z-Wave Home Control Kit.

2.1 Purpose

The Z-Wave modules included in the Z-Wave Home Control Kit are not pre-programmed enabling the developer to select which Z-Wave sample application should be the base for evaluation, testing or other objectives.

The purpose of this document is to provide the reader a step-by-step instruction on how to get started with the Z-Wave Home Control Kit in terms of programming the Z-Wave modules, creating a simple network and getting familiar with the most commonly used Z-Wave developer tools.

2.2 Audience and prerequisites

This document is targeted to developers designing and developing Z-Wave enabled products and also for general interest in learning how a Z-Wave network is being created, from Z-Wave chip programming to controlling a Z-Wave device.

The reader of this document is expected to have read the introductory Z-Wave document, "**Z-Wave Node Type Overview and Network Installation Guide**" which is available for download at http://support.zen-sys.com.

3 WHAT'S IN THE Z-WAVE HOME CONTROL KIT?

The Z-Wave Home Control Kit consists of the following packages:

Z-WAVE BASE KIT		Z-WAVE REGION KIT		Z-WAVE SDK
The Z-Wave Base Kit consists of development platform modules, power supply units, cables etc. that are necessary to program the Z-Wave modules with a Z-Wave application.		The Z-Wave Region Kit comes in different frequency variants and consists of Z-Wave 300 and 400 series based reference design modules that are RF matched for the specific region. The Regional kit also contains ZM4101, ZM4102 and ZM3102 modules that are ready for integration directly into the OEM product.		The Z-Wave SDK is available through download center and access to the download center is included with the Z-Wave Home Control Kit. From the download center the developer will be presented with all relevant technical documentation, latest SDK release, developer's FAQ, known issues database and much more.
Qty	Product Name	Qty	Product Name	Contains:
4	ZDP03A, Z-Wave Programmer / Development Platform	10	ZM4101, Z-Wave SiP module (QFN64)	Z-Wave® Protocol Stack
2	ZM4125-S, Socket based module for ZM4101	2	ZM4102, Z-Wave 400 series module (12.5x13.6mm)	Z-Wave® API function descriptions and technical documentations
3	Flexi Antenna	3	ZM4125, ZM4101 ref. design module	Embedded and PC based sample applications
3	USB Cable	2	ZM3102, Z-Wave 300 series module (12.5x13.6mm)	Development tools, Z-Wave Programmer, Z-Wave Zniffer, Build environments, And tools to build your Z-Wave application.
1	Battery pack	2	ZM3120, ZM3102 ref. design module.	
4	Power Supply unit	All Z-Wave modules included are not programmed and must be programmed ONLY with the correct frequency settings i.e. US based modules must only be programmed with US frequency based applications.		
1	Getting Started with Z-Wave			
1	Z-Wave Brand Kit			

* Not included in region kit for Japan.

The Z-Wave Technical Service website is an on-line resource designed to give all Z-Wave Developers access to the latest Z-Wave SDK, Z-Wave Device & Command Class development projects, Z-Wave Certification program and much more technical documentation.

NOTE! Access to this web service is for Z-Wave licensees ONLY and do not share the login data registered on the Sigma Designs or the Z-Wave Alliance homepage.

How to get access?

Having purchased a Z-Wave Home Control Kit and accepted the Z-Wave Developer's Kit Agreement, you are entitled to be granted access to the Z-Wave Technical Service website. We strive to get the information about new Z-Wave Partners from our distributors as quickly as possible, so you will probably already have been created as a user of the website with your e-mail address as username.

In case you have not yet receive your login information then please write an email to support@zen-sys.com with your name, email address, company data and purchase information.

NOTE : There is no CD-ROM included in the Z-Wave Home Control Kit. Please visit http://support.zen-sys.com to download the latest Z-Wave SDK.

For additional kits and/or Z-Wave modules, please contact Sigma Designs 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@digikey.com
Web: www.digikey.com/Zensys

4 GETTING STARTED

4.1 Identifying your Z-Wave modules and development boards

ZDP03A, Z-Wave Programmer / Dev. Platform. This board is used to connect ZM4125-S, ZM4125 or ZM3120. The Z-Wave Programmer PC app will interface to the programming software of the ATmega128 processor to program the Z-Wave module with a Z-Wave embedded application.

ZM4125-S, is frequency independent i.e. it can run any of the Z-Wave frequencies, because it has no SAW filter. The ZM4101 module fits into the socket for the Z-Wave Developer to program its Z-Wave app and run it either directly on the ZM4125-S or to mount the ZM4101 into OEM product samples.

ZM4101. Z-Wave SiP module (QFN64).

ZM4102. Z-Wave 400 series module (12.5x13.6mm).

ZM4125 is the ZM4101 reference design module RF matched for a specific Z-Wave frequency. It contains SMA connector, PCB antenna, all necessary external components and SAW filter for optimal RF performance.

ZM3102, Z-Wave 300 series module (12.5x13.6mm).

ZM3120 is the ZM3102 reference design module RF matched for a specific Z-Wave sub-1GHz frequency. It includes a PCB antenna, is ready for SMA connector and has all necessary external components including a SAW filter for optimal RF performance.

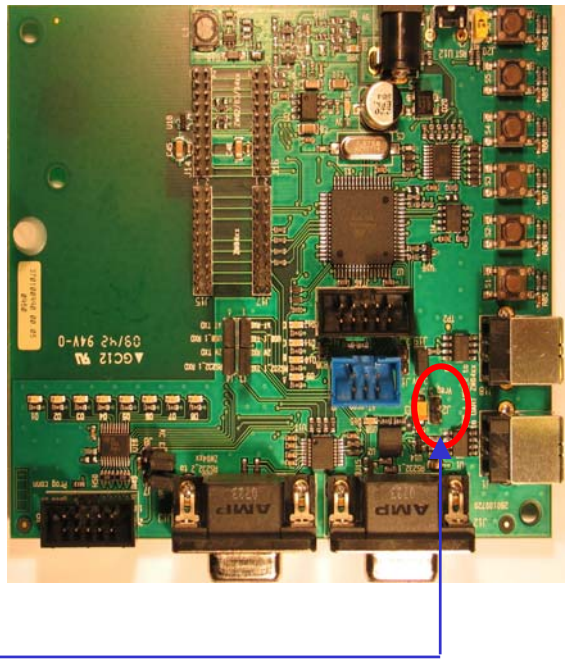
4.2 Development Tools

4.2.1 Z-Wave Programmer & Development Platform

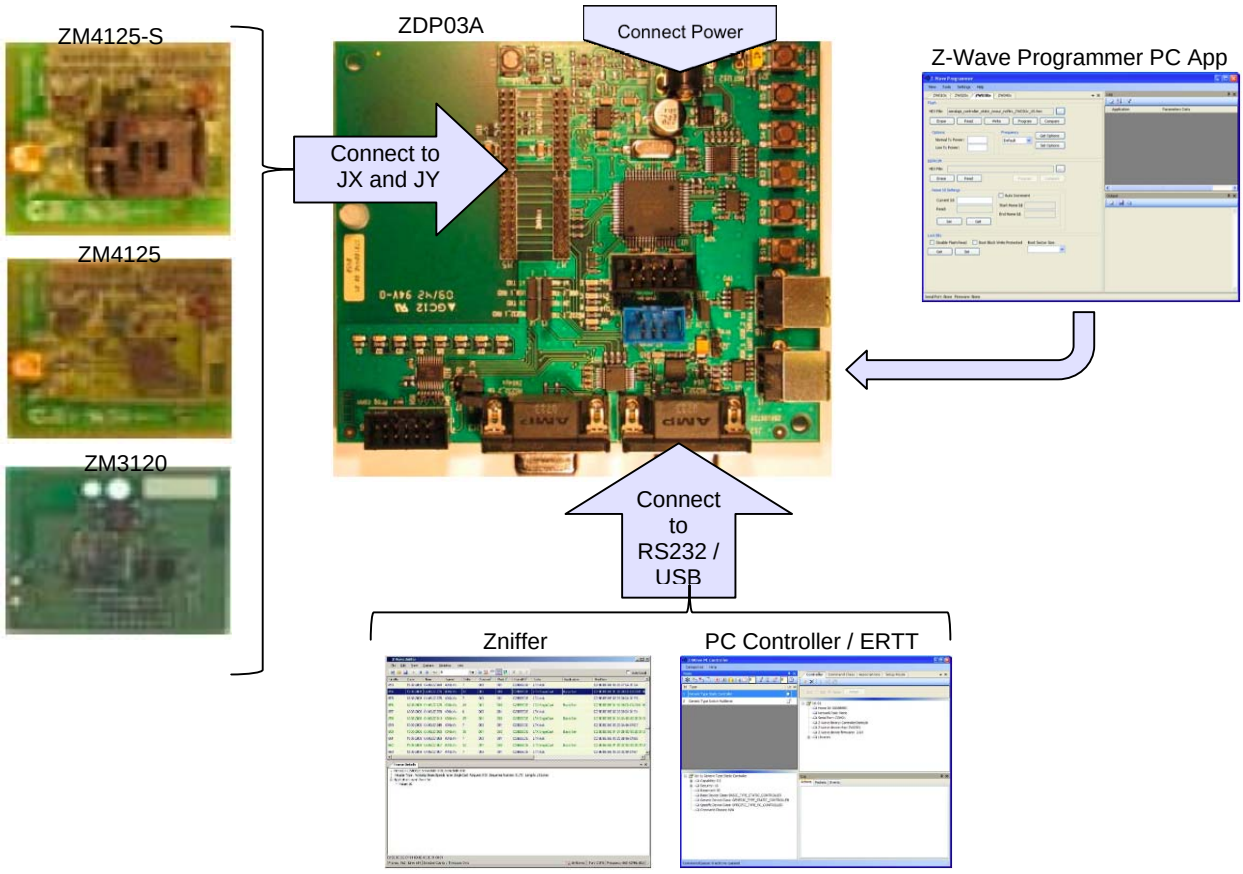
The ZDP03A is both a development platform and Z-Wave programmer. The ZDP03A can program all generations of the Z-Wave ASIC and modules, the external EEPROM / Serial FLASH in e.g. ZM4125 and configuration of the RF output power of the ASIC.

The programming set-up required when using ZDP03A, Z-Wave Programmer consists of:

- ZDP03A, Z-Wave Programmer
- 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 board can also be powered via USB with the connected PC, note when powered from the USB, make sure J2 is shorted)



As a development platform, the ZDP03A board is used to connect necessary development tools such as Zniffer, PC Controller/ERTT and any embedded application utilizing the implemented components.



4.2.2 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 “Z-Wave Zniffer User Guide” [INS10249] for additional information about the Zniffer tool.

4.3 Download and install the Z-Wave Software Developer's Kit (SDK)

NOTE! The Z-Wave SDK runs on a Microsoft Windows environment.

Step 1 : Download a copy of Z-Wave SDK from the Z-Wave Technical Service website. Navigate to the "SOFTWARE" menu and download the latest release that fits your development platform i.e. 300 or 400 series platform. Once download is completed, unzip the SDK into your selected folder.

Step 2 : Install the SDK into your local hard drive. The content that you have unzipped contains a "start here.html" file that will display the SDK contents using an internet browser e.g. Windows Internet Explorer. Open the file "start here.html", navigate to "Developer's Kit SW" and click on the icon. This action is linked to the setup.exe file located in the \data folder and will copy the Z-Wave Developers Software into your local hard drive.

For detailed description of SDK contents, refer to the specific software release note.

4.4 Programming Z-Wave modules and create a simple Z-Wave network

This section will take you through basic steps that prepare your PC environment for programming the Z-Wave modules and to learn how a simple Z-Wave network is created by using the tools provided by the Z-Wave SDK and the Z-Wave modules from the Z-Wave Home Control Kit.

4.4.1 Prepare Z-Wave module programming environment

In order to program a Z-Wave module using the tools and apps from the Z-Wave Home Control Kits you must prepare your pc environment by installing USB drivers and the Z-Wave Programmer PC app. For detailed instruction refer to "Z-Wave Programmer User Guide" [INS10679].

a. **Install the USB driver** - make sure the ZDP03A board is not connected to your PC and run the SiLabs USB driver program "CP210x_VCP_Win2K_XP.exe" located in C:\DevKit_X_XX…\Tools\Programmer\PC. This action will result in a \SiLabs folder is created with USB driver data files.

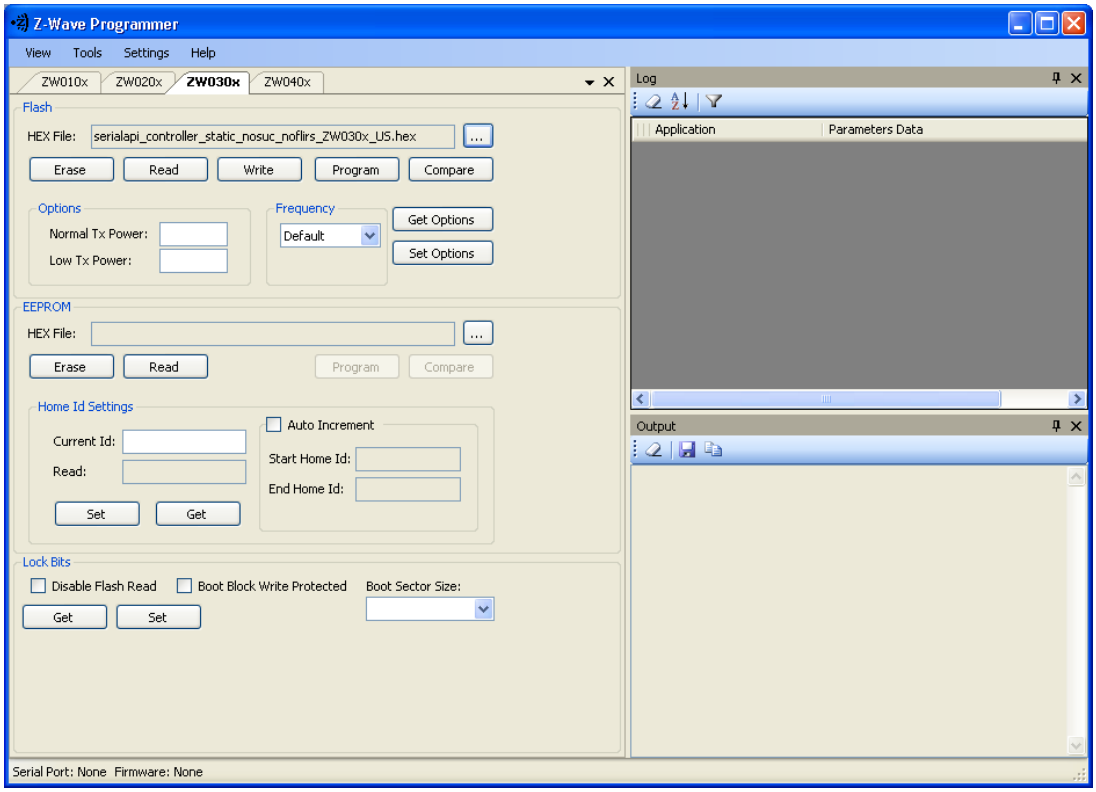
b. **Install the Z-Wave Programmer PC app** - make sure the ZDP03A board is not connected to your PC and run the setup program "setup.exe" located in C:\DevKit_X_XX…\Tools\Programmer\PC and follow the installation instructions on the screen.

c. **Connect the ZDP03A board to the PC** - Apply power to the ZDP03A using one of the power supply units included in the Z-Wave Home Control Kit then connect a USB cable between ZDP03A and PC. Alternatively, just connect a USB cable between ZDP03A and the PC (and short jumper J2). The Windows

environment will start a "Found New Hardware Wizard" and you must make sure not to allow Windows to auto-search/select a USB driver. Instead direct the wizard to use the driver installed from above step a. This step is required to complete ***twice*** to avoid being prompted ***every time*** you connect.

d. Restart the PC!

e. Launch the Z-Wave Programmer PC app - Click on Start - navigate to \All Programs\Zensys\Z-Wave Programmer\ and click on "Z-Wave Programmer". When the Z-Wave Programmer has been launched, click on the "Settings" menu and select the correct COM port and verify communication between the PC app and ZDP03A.



4.4.2 Program Z-Wave modules and install PC tools

With the Z-Wave module programming environment prepared, you are now ready to program the Z-Wave modules, install other Z-Wave PC applications and tools.

KEEP IN MIND Z-WAVE 400 SERIES USES OTP (ONE-TIME-PROGRAMMABLE) MEMORY; HENCE IT CAN BE PROGRAMMED ONLY ONCE!

Z-WAVE 300 SERIES USES FLASH MEMORY AND CAN BE PROGRAMMED MULTIPLE TIMES!

4.4.2.1 Z-Wave Zniffer PC app & Z-Wave Zniffer module

Always visit Z-Wave Technical Service website, http://support.zen-sys.com for latest revisions of the Z-Wave Zniffer tool and more.

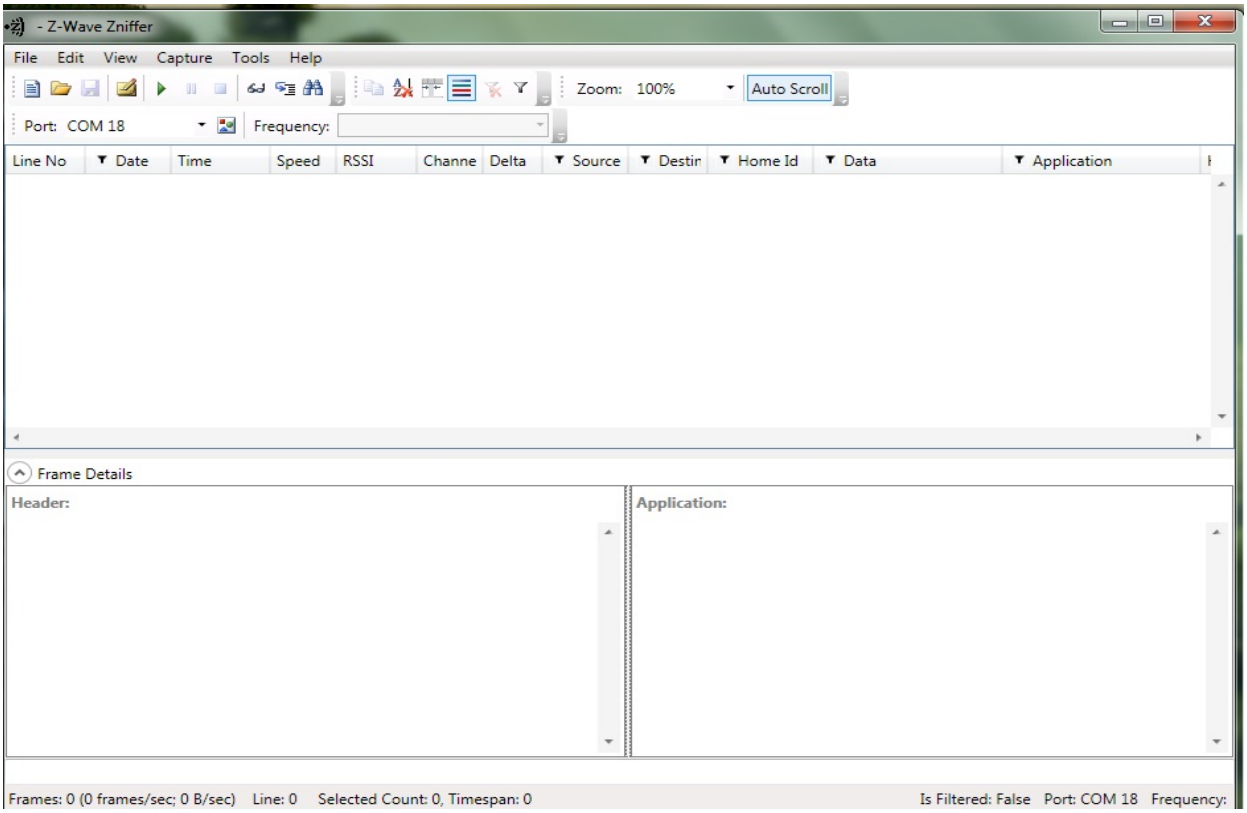
a. **Program a Z-Wave Zniffer module** - Connect a ZM4125 module to the ZDP03A board on the Z-Wave Module Connectors, J14 & J15. Connect ZDP03A to the PC, launch the Z-Wave Programmer PC app and click on "Detect Target Ctrl+D" in the "Tools" menu and verify correct detected platform i.e. ZW040x.

In the "OTP Memory" field select the Zniffer firmware "sniffer_ZW040x.hex" file you have downloaded from the Z-Wave Technical Service website. Click on "Program" and re-confirm when being asked to program the OTP memory. Reset the module by a single press on the white push-button next to the DC power plug of ZDP03A and verify LED1 is blinking, which is an indication the Z-Wave module is programmed with Z-Wave Zniffer firmware.

b. **Install the Z-Wave Zniffer PC app** - run the "setup.exe" located in C:\DevKit_X_XX\Tools\Zniffer\PC or the one you have downloaded from the Z-Wave Technical Service website, whichever is the latest revision. You can verify the SDK Zniffer PC app revision by the software release note of the SDK in question.

c. **Connect the Z-Wave Zniffer module to PC** - connect a serial cable on the RS232-1 port of ZDP03A to the PC's COM port or use a RS232 to USB converter. NOTE! Guidelines in how to install RS232 to USB converter is not covered by this "Getting Started with Z-Wave" document.

d. **Launch the Z-Wave Zniffer PC app** - Click on Start - navigate to \All Programs\Zensys\Z-Wave Zniffer\ and click on "Z-Wave Zniffer". When the app has been launched, click on the "Capture" menu and "Port Selection" and "Auto". Allow the Zniffer PC app to detect the connected Z-Wave Zniffer. Now select the frequency you are working on also in the "Capture" menu. For detailed instructions in how to use the Z-Wave Zniffer PC app, refer to the "Z-Wave Zniffer User Guide" [INS10249].


- Z-Wave Zniffer
File Edit View Capture Tools Help
Zoom: 100%
Auto Scroll
Port: COM 18
Frequency:
Line No
Date
Time
Speed
RSSI
Channe
Delta
Source
Destir
Home Id
Data
Application
Frame Details
Header:
Application:
Frames: 0 (0 frames/sec; 0 B/sec) Line: 0 Selected Count: 0, Timespan: 0
Is Filtered: False Port: COM 18 Frequency:

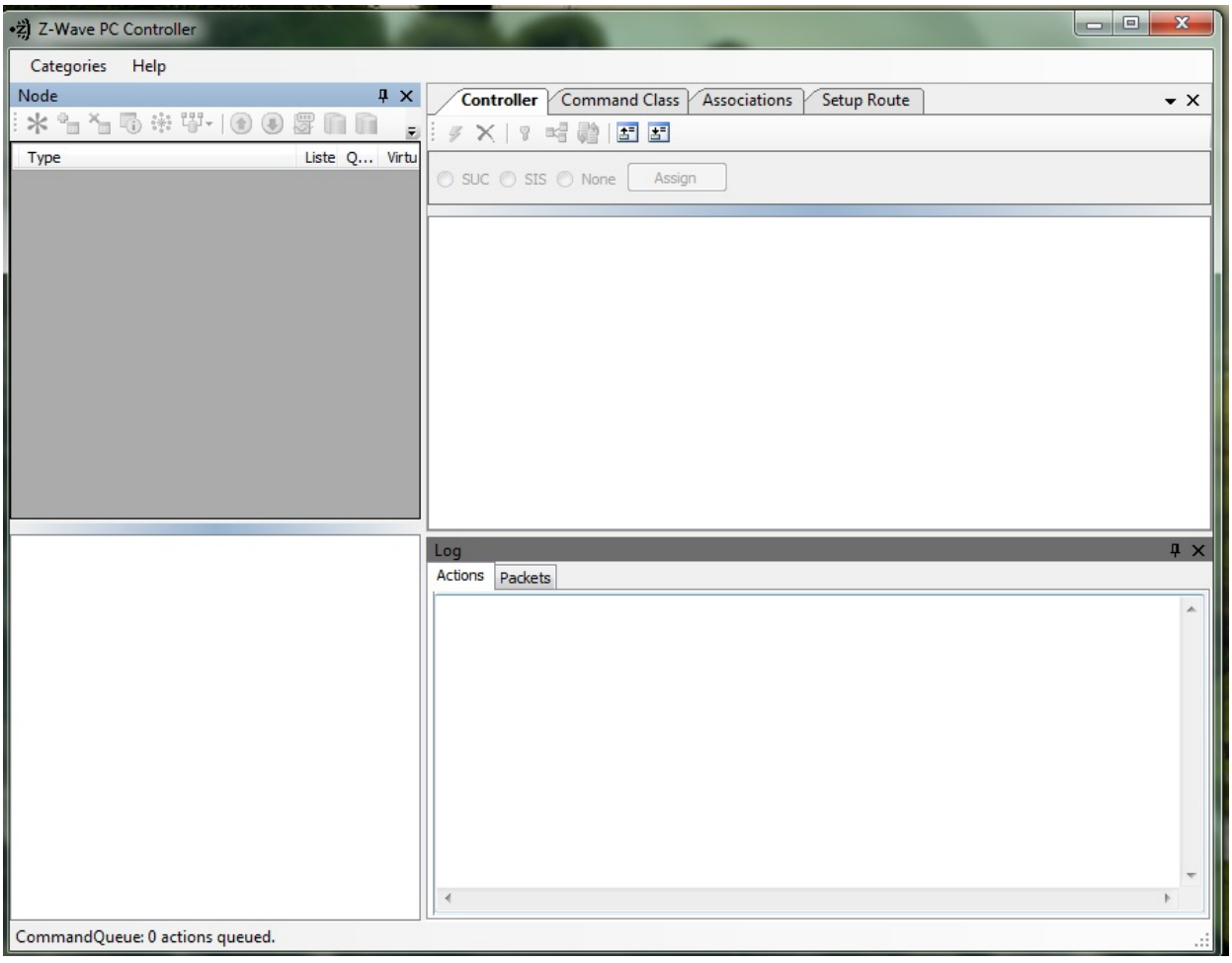
4.4.2.2 Z-Wave PC controller app & Z-Wave Serial API Static Controller

Perform same steps as in 4.3.2.1, only this time program another ZM4125 module on another available ZDP03A board with "serialapi_controller_static_ZW040x_xx.hex" located in C:\DevKit_X_XX\Product\Bin\SerialAPI_Controller_Static\ and in addition erase the external NVRAM by click on "Erase" under "External Non-Volatile Memory" on the Z-Wave Programmer PC app. If you prefer to program the controller module with a unique Z-Wave home id this is also the time to do so.

USE CAUTION WHEN PROGRAMMING THE Z-WAVE MODULES!
MAKE SURE TO ONLY PROGRAM THE MODULES WITH THE BINARY FILES THAT ARE BUILD WITH THE CORRESPONDING FREQUENCY SETTING
E.G. AN US IMAGE MUST BE PROGRAMMED INTO A Z-WAVE MODULE RUNNING US FREQUENCY.

On the PC side you must install the Z-Wave PC controller app located in C:\DevKit_X_X\PC\Bin\ZWavePCController\.

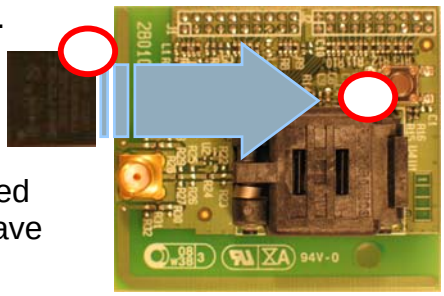
Connect the Z-Wave Serial API Static Controller to your PC following the same steps as the Z-Wave Zniffer setup and launch the newly installed Z-Wave PC Controller app. Refer to "PC based Controller User Guide" [INS10240] for details on to use this app.



4.4.2.3 Z-Wave LED dimmer

Mount a ZM4101 on the socket of a ZM4125-S module.

Make sure the dot on the upper right corner of the ZM4101 is aligned with the push button on the module.



To verify the ZM4101 module has been correctly inserted to the socket, use the detect target function of the Z-Wave Programmer PC app.

Perform same steps as in 4.3.2.1, only this time program the above prepared ZM4125-S module mounted on another available ZDP03A board with "leddimmer_ZW040x_**xx**_devmode.hex" located in C:\DevKit_X_XX\Product\Bin\LED_Dimmer\.
Remember to use the correct frequency images on the Z-Wave modules!

Note that this file includes the naming "_devmode". This is the file that must be programmed into the OTP memory area of the 400 series chip to run development mode which is explained in chapter 5 "Building your own Z-Wave app".

When programming is done, remove power from the ZDP03A board.

4.4.3 Creating a simple Z-Wave network

This chapter will take you through the basic steps on how to set-up a Z-Wave network using the PC tools and a Z-Wave module you have prepared from previous chapters.

1. Launch the Z-Wave Zniffer PC app if not already done according to chapter 4.3.2.1, for recording of Z-Wave RF frames.

2. Launch the Z-Wave PC Controller app if not already done according to chapter 4.3.2.2 and press the “Add” button  on the Z-Wave PC Controller app (verify on the Z-Wave Zniffer PC app frames are being captured continuously with a pre-defined interval). This frame is named “Transfer Presentation” and is an indication from the Z-Wave controller device that it is now ready to include a new device into the network.

3. Apply power to the module programmed with the LED Dimmer app from chapter 4.3.2.3 and verify on the Z-Wave Zniffer frames are being captured.

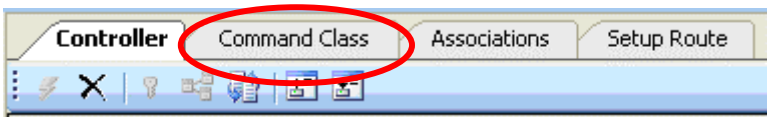
   The expected frame is a “Node Information Frame” that is being broadcasted. This frame is the identity of a Z-Wave app and contains important information about the profile of the device.

   While the Z-Wave PC Controller is in “Add” mode and a Node Info is received it verify whether the device is in default state i.e. node id = 0 and in such case it will start the inclusion process.

4. Verify the node list has been incremented with a new device on the Z-Wave PC Controller app and study the frames captured on the Z-Wave Zniffer.

5. Now control the newly included LED dimmer by clicking on the “Basic Set On” and “Basic Set Off” buttons  and verify the LED Dimmer is being switched On/Off respectively and also the captured frames.

   You should also get familiar with the command class fields of the Z-Wave PC Controller app:

   On this menu you will be listed with all the command classes the device has advertised support for.

   Try playing around with the Multilevel Switch Command Class and get a feel of how to dim up/down the LED Dimmer and most importantly study the frames captured by the Z-Wave Zniffer.

6. To remove the LED Dimmer from the network simply press the “Remove” button  and press three times within 1.5 seconds on the button of the ZM4125-S module of the LED Dimmer.

   Notice the Z-Wave Zniffer and verify the LED Dimmer is now set back to node id 0.

5 BUILDING YOUR OWN Z-WAVE APP

When you are ready to program a Z-Wave module with your own developed embedded Z-Wave application, there are three main steps to execute:

1. First you need to compile and link your application (source code) using the **KEIL** compiler to create the binary HEX file. The Z-Wave SDK includes sample apps for your reference.

2. If you are creating a controller or enhanced slave application, you must initialise/erase and program the external EEPROM / Serial FLASH with a unique home id using the Z-Wave Programmer. This action only needs to be done once. However, if the application is based on SDK 4.5x or SDK 6.0x, programming the home id is optional since the Z-Wave stack will create a random home id whenever the controller application is set back to default.

3. The final step is to program the Z-Wave module using the Z-Wave Programmer.

5.1 KEIL™ Software Development Tool

The KEIL™ Software development tool for 8051 micro controllers is a third party software tool that is required to build your embedded Z-Wave application and is **not supplied with the Z-Wave SDK or Z-Wave Home Control Kit.**

Refer to the SDK software release note for details on which revision of the Assembler, C compiler, Linker and Librarian you should use to assure seamless integration with the Z-Wave protocol APIs.

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

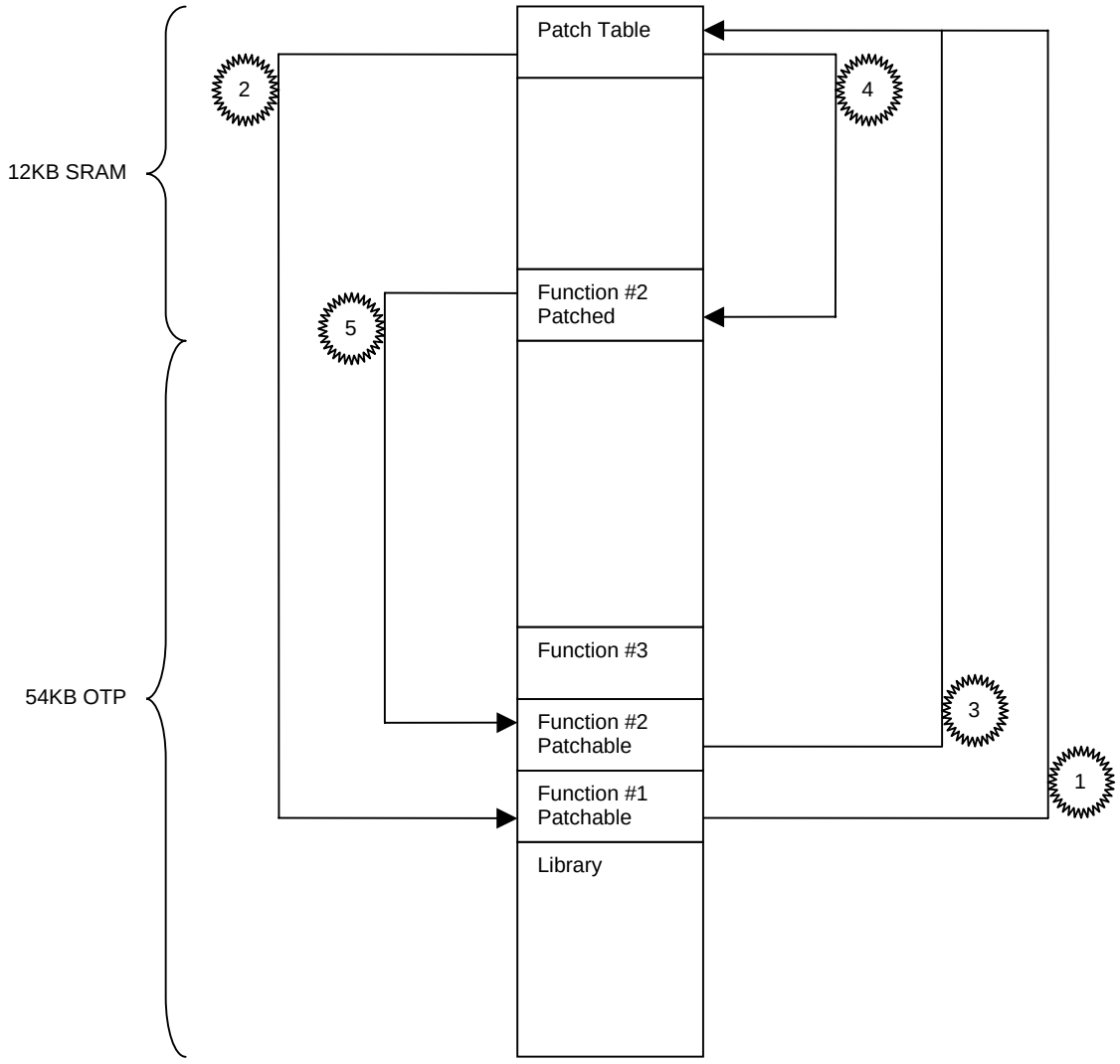
KEIL Software Inc.
1501 10th Street, Suite 110
Plano, TX 75074
USA
Toll Free: +1 800-348-8051
Phone: +1 972-312-1107
Fax: +1 972-312-1159
Sales: sales.us@keil.comn

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

5.2 400 series Development Mode

The 400 Series build environment is different compared to previous 100/200/300 Series due to the ASIC being based on 64KB OTP instead of 32KB Flash. Nevertheless, the 400 series platform supports a Flash-like development mode enabling Z-Wave developers on the 400 series platform to re-iterate the software build over and over again without the necessity of programming more ASICs. The supported modes are shown on the figure below. Refer to "Z-Wave 400 Series Developer's Kit v6.0x.xx Contents" [INS11442] for details on the 400 series development mode.

The 400 series development mode is essentially a patchable system that utilizes the RAM area to run program code. The illustration below shows how an application function #2 configured as patchable is being processed.



The patchable system means that, there must be separate images programmed into the RAM area which contains the patchable functions, and into the OTP area containing the state machine.

In order to do so, different source code files must exists; one main application file and another containing the patched function codes. An example is below two files from the LED Dimmer sample app:

- **LEDdim.c** - contains the source codes for the LED Dimmer application state machine.
- **LEDdim_patch.c** - contains the patchable source code for the LED Dimmer app.

When you build your Z-Wave application using the MAKE file system, there will be a couple of generated binary files for both the normal OTP mode and development mode files:

1) ***application-name*_ZW040x_xx.hex**
2) ***application-name*_ZW040x_xx_devmode.hex**
3) *application-name_ZW040x_xx_devmode_OTP.hex*
4) *application-name_ZW040x_xx_devmode_patch.hex*
5) ***application-name*_ZW040x_xx_devmode_patch_RAM.hex**
6) *application-name_ZW040x_xx_devmode_RAM.hex*

Not all files are being programmed into the Z-Wave module; hence caution must be taken when selecting the files for the specific area.

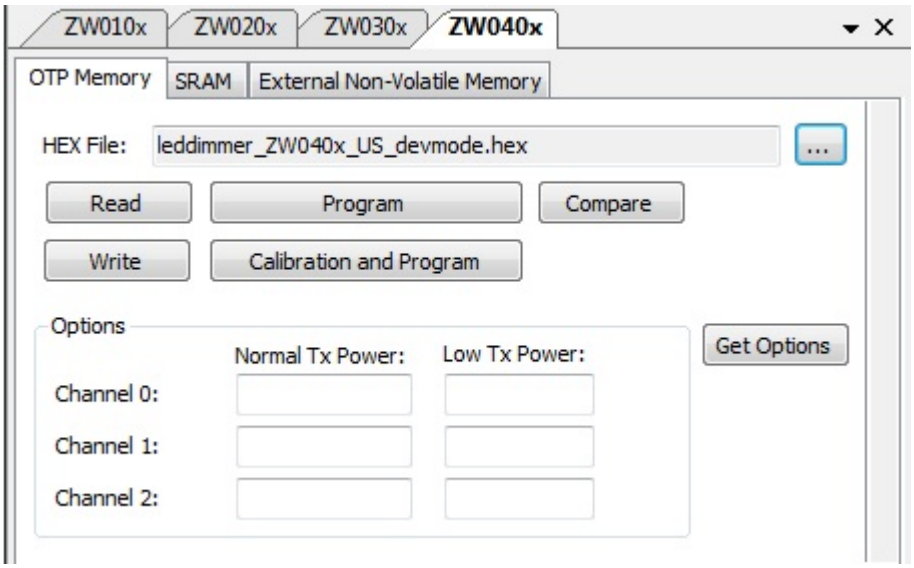
File no. 1 "***application-name*_ZW040x_xx.hex**" is the normal file i.e. normal OTP mode and no development mode. This mode will not search for any patchable functions.

File no. 2 "***application-name*_ZW040x_xx_devmode.hex**" is the OTP file in development mode. The patchable functions existing in this file expects to find patchable functions in the SRAM area, but in case it is not found it will execute its own function.

File no. 5 "***application-name*_ZW040x_xx_devmode_patch_RAM.hex**" is the RAM file in development mode. This file contains the patchable functions and can be re-written multiple times.

File no. 3, 4 and 6 are NOT to be programmed into the Z-Wave ASIC.

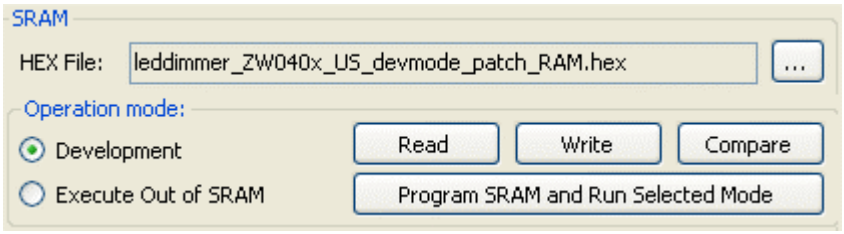
From previous chapter 4.3.2.3, Z-Wave LED Dimmer, you was instructed to program the ZM4101 with the file named "leddimmer_ZW040x_xx_devmode.hex".



By default all sample applications provided with the 400 series platform SDK comes with patchable Z-Wave functions: ApplicationInitHW, ApplicationInitSW, ApplicationTestPoll, ApplicationPoll, ApplicationCommandHandler, ApplicationNodeInformation, ApplicationSlaveUpdate or ApplicationControllerUpdate.

To program the SRAM section of the chip, modify the source code of the patchable functions in LEDdim_patch.c and compile it.

Select "leddimmer_ZW040x_xx_devmode_patch_RAM.hex" and click on "**Program SRAM and Run Selected Mode**" into the ZM4125-S module one had prepared. The changes will effect accordingly.



The 400 series development platform makes it more flexible, comfortable and convenient for the Z-Wave developer when coding in an OTP environment.

IMPORTANT!
LIMITATIONS WHEN WORKING WITH
Z-WAVE 400 SERIES DEVELOPMENT MODE:

- A patchable function and the patch function for it *must* have exactly the same declaration. This constraint goes for both the functions parameters and the functions local variables.

- The functions *must* be declared reentrant to make the parameters and local variables allocated on the "Reentrant Stack". Also because of this, static variables are not allowed in these functions.
which resides in RAM.

- The patchable function and corresponding patch function *must* NOT be empty.

- It is not possible to have declaration of initialized static variables in the part of the program, which resides in RAM. Only variables in the OTP-part of the program can have declared initialized static variables.
A linker error stating ‘segment mismatch’ will be generated when this rule is violated. All static variables *must* be initialized in the initialization part of your code.

- It is not possible to use IDATA, PDATA and BIT data variables in the patch part of the program,

6 Z-WAVE OVERVIEW

6.1 Z-Wave Network Node Types

The Z-Wave network consists of two basic network node types; controller and slave.

The controller node is able create the network and calculate routes based on feedback from every node it adds to the network.

The slave node is generally an input and output unit which replies to a message when being addressed. Both types exist in different variants described in below sub-chapters.

A Z-Wave network can consist up to 232 nodes, which can be freely shared between controller and slave nodes.

6.1.1 Controller Nodes

A Z-Wave network is established by a controller. The controller that creates a Z-Wave network is a **Primary Controller** which has the capability of including and excluding other Z-Wave nodes.

The Primary Controller is also used to include all subsequent nodes in the network including other controller nodes which then becomes **Secondary Controllers**. A controller node in default state is always the Primary Controller until being included into a network in which it will learn its new role.

A Static Controller can be enabled to become a **Static Update Controller (SUC)**, which adds advanced network management functionalities yet transparent end user friendly network management functionality to the end user.

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 is the controller device that assigns the next network node id to new nodes. Furthermore it enables other controller devices to become Inclusion Controller enabling these to also being capable of including/excluding nodes on behalf of the SIS.

The common functionalities for all controller nodes include:

Controller devices

- is by default the Primary Controller and by that the ability to create a new Z-Wave network
- can become Secondary Controllers when being added to a network
- will become Inclusion Controllers when a SIS node is present
- can activate the SUC/SIS role of a Static Controller
- have full network topology awareness

6.1.1.1 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.

As 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.

6.1.1.2 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.

6.1.1.3 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.

6.1.1.4 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 vise versa.

6.1.2 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.

6.1.2.1 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.

6.1.2.2 Routing Slave

The Routing Slave can host a number of routes for reaching other slaves or controllers. Such routes are called “Return Routes”[1]. 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.

[1] ”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.

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.

6.1.2.3 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.

6.1.2.4 Enhanced Slave 232

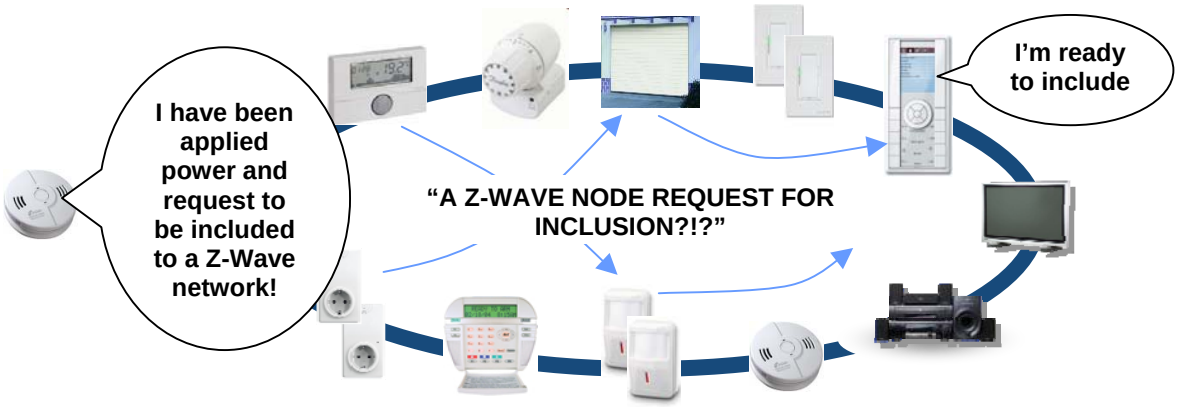
The Enhanced Slave 232 has the same capabilities as the “normal” enhanced slave with extended number of routes it can host to address other controller and slave devices. The Routing Slave and Enhanced Slave have a limitation of maximum 5 destination nodes which it can address. The Enhanced Slave 232 can address up to 232 destination nodes. For all three slave types the routes must be assigned by a controller device.

6.2 Network Wide Inclusion

With the explorer frame, the Z-Wave technology supports 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.

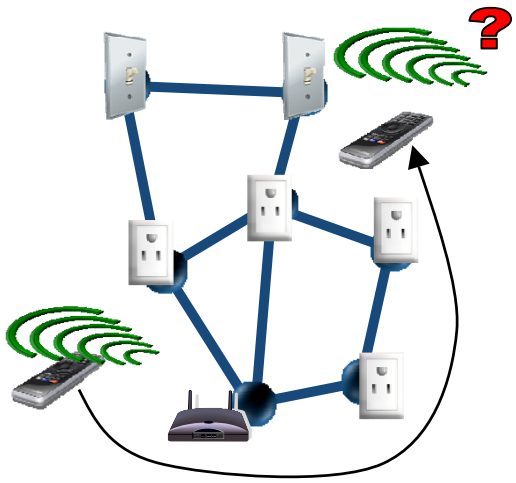
All embedded sample applications provided on SDK releases supporting the explorer frame, will initiate Network Wide Inclusion when being applied power and in case not already being part of a network.



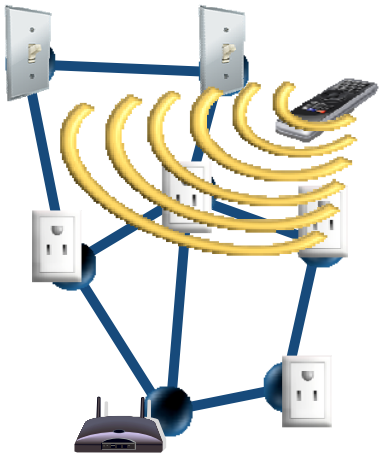
Other explorer frame capable products are already part of the network and will help the new device to transfer the message to the controller unit. The only user-trigger necessary is at the controller unit where the home owner must set the controller into a learning state. Once the auto inclusion request has been received by the controller the new unit will be assigned the home id of the network and an available node id.

6.3 Dynamic Route Resolution

With the explorer frame, the Z-Wave technology provides a true self-healing mesh network through Dynamic Route Resolution. 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.



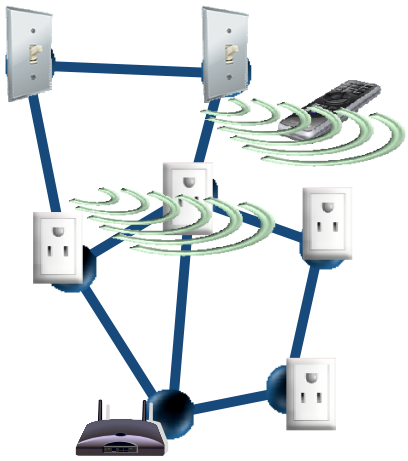
A simple example of when and where Dynamic Route Resolution is beneficial could be for a Z-Wave enabled Remote Controller (RC) which is mobile and is never at a static position.



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 there is a small possibility that the static routes may not be valid no longer when the RC is relocated.

The explorer frame based RC will ask other explorer frame based products to help address the target node. The RC initiates an explorer frame containing protocol information including a Search Request command and the application payload and 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.

When the Explorer frame reaches its destination the target node is provided with the required protocol data to acknowledge the package and carry out the application command immediately the app payload was included in the Search Request.



Dynamic Route Resolution is a self-healing mechanism that enables the Z-Wave application and the Z-Wave developer to focus on the application layer and rely on the Z-Wave protocol to handle complex network management.

7 SAMPLE APPS

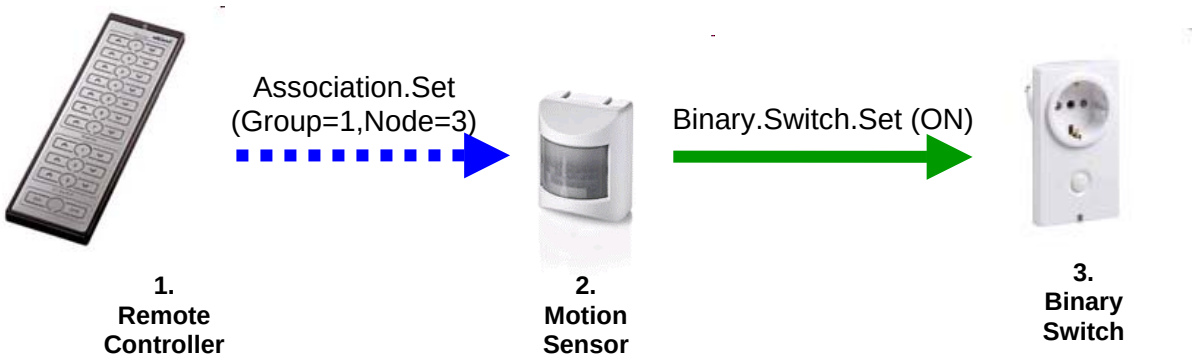
7.1 Embedded Sample Applications

The SDK contains sample codes to help the Z-Wave Developer getting started with developing a Z-Wave application. The source code, make files and µVision project files are located in the appropriate project

The following chapters provide a short introduction of the most commonly used sample applications. For details of all applications refer to the SDK software release note.

7.1.1 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 motion 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.

7.1.2 Development Controller

The Development Controller sample application simulates a simple Z-Wave RF remote. The application user interface is based on the pushbuttons of the Z-Wave development platform module.

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

7.1.3 LED Dimmer

The LED Dimmer sample source code is built for a slave library application on a Z-Wave module, which uses the LED's of the Z-Wave development platform 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

7.1.4 Serial API

The SDK includes Serial API embedded applications linked to all available library types of the SDK. 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 SDK:

- 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 SDK or it can be modified to fit specific needs. The UART on the Z-Wave Module is pre-configured as 115200 baud, no parity, 8 data bits and 1 stop bit.

7.1.5 MyProduct

To help you getting started with your Z-Wave application the SDK includes the MyProduct project files.

MyProduct contains the minimum framework to begin developing a slave application.

7.2 PC Sample Apps

The SDK contains 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 "Z-Wave DLL User Guide" [INS10250] for details.

7.2.1 Z-Wave PC Controller incl. ERTT (Enhanced Reliability Test Tool)

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 and in addition the PC controller of SDK 6.0x includes the ERTT application for RF range test. For detailed description refer to "PC based Controller User Guide" [INS10240].

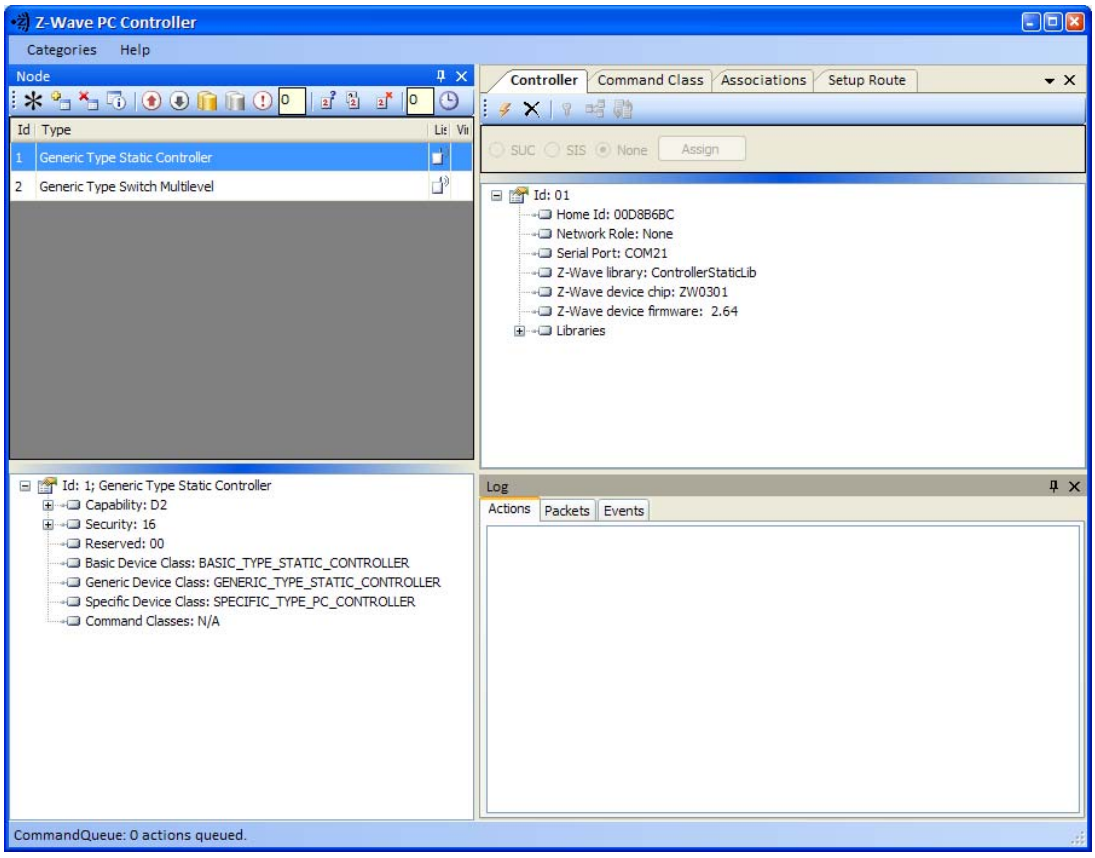


Figure 1, Screenshot of Z-Wave PC Controller

7.2.2 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 “Z-Wave Bridge User Guide” [INS10245] 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.

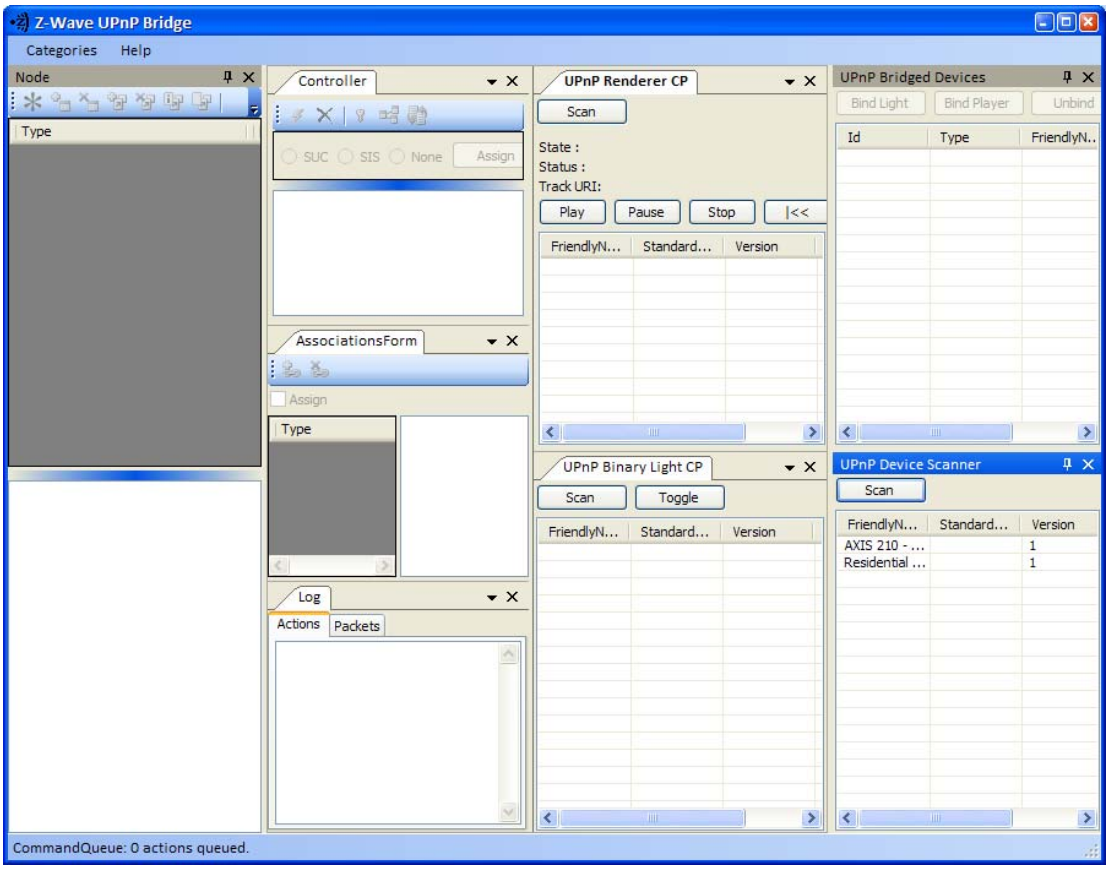


Figure 2, Screenshot of Z-Wave UPnP Bridge