Instruction

Z-Wave Programmer User Guide (ZDP03A, PC)

Document No.:	INS10679
Version:	15
Description:	This document describes the use of ZDP03A Z-Wave Programmer and the Z-Wave PC Programmer application.
Written By:	JFR;SSE;ANI;MVO;ABR;SRO;SAR;NTJ
Date:	2014-07-01
Reviewed By:	JFR;MVO;ANI;BBR;NTJ;ABR;SRO;SAR;CST
Restrictions:	Partners Only

Approved by:

Date	CET	Initials	Name	Justification
2014-07-01	08:44:18	NTJ	Niels Thybo Johansen	

CONFIDENTIAL

REVISION RECORD

Doc. Rev	Date	By	Pages affected	Brief description of changes
1	20061001	SSE JFR	ALL	Initial draft
2	20070322	JFR	5.8.8 5.10.1	Shortcut keys revised Console frequency parameters revised
3	20070420	SSE	-	Programmer firmware upgrade added
4	20080227	JFR	-	Revised with respect to 100 Series
5	20081212	DDA JFR	ALL	Revised and updated with respect to 400 Series and ZDP03A
6	20090929	JFR	3.2 3.2	ZDP03A Jumper J20 settings clarified Not possible to program external EEPROM on 100 Series based modules when using ZDP03A
7	20100127	DDA	4 & 5	Updated screenshot. MTP description added. Added Hold device in reset, Write on PCB button, Use test frequencies. Description of SRAM updated. Console interface syntax added. Firmware upgrade updated.
8	20110210	DDA	5.4.5	Section about Calibration and Program feature added.
9	20110217	DDA	5.9	Console interface syntax updated.
10	20110816	DDA	-	Added ZW040x RF settings Updated interface
11	20130515	MVO	All	Replaced first section and updated headers/footers Updated references
11	20130527	DDA	3.1, 4.4, 5.2, 5.4 & 5.7	Added ZW050x descriptions.
12	20130607	DDA	1, 3.2, 5.4	Updated Abbreviations, added UART shorting picture, updated screenshots.
12	20130918	ABR	All	Editorial cleanup. Separated on-board and ISP descriptions. Added details on 500 series programming. Added details on 500 series calibration. Updated screen dumps. Console interface syntax updated. Console interface help text updated. Removed 100 series details. Added details on 500 series calibration hardware.
13	20131001	ABR	5.3.2	Added more on USB/UART/SPI programming interfaces
13	20131002	ABR	5.3	Added more on USB/UART/SPI programming interfaces
13	20131112	ABR	Various	Minor editorial cleanup. PC Programmer v2.68 screen dumps added.
13	20131205	ABR	Various	Updated front page Z-Wave logo. Removed ZDP02A and Interface module due to End Of Life. Reordered sections: 500 series details are described first. Added new figure to 5.1: Using DP03A for USB-UART conversion Improved DP03A jumper overview + added details for J3+J4: USB-UART
14	20140220	JFR MVO	5.6 4.4.6 5.10.1	Supports only EEPROM's and via SPI Revised lock bits description Revised console interface description
15	20140603	JFR	5.4.6.3	Revised ZW050x RF settings

Table of Contents

Table of Figures

Table of Tables

1 ABBREVIATIONS

Abbreviation	Explanation
EEPROM	Electrically Erasable Programmable Read Only Memory
Flash	A type of EEPROM memory that can be erased and reprogrammed in blocks.
GUI	Graphical User Interface
HW	Hardware
ISP	In-System Programming
LED	Light Emitting Diode
MTP	Many Time Programmable; a type of NVM
NVM	Non-Volatile Memory
NVR	Non-volatile register
OTA	Over The Air (firmware update)
OTP	One-Time Programmable; a type of NVM
PA	Power Amplifier
PCB	Printed Circuit Board
pF	Pico Farad
RF	Radio Frequency
Rx	Receive
SD34xx	Range of 400 series Z-Wave Single Chips
SD35xx	Range of 500 series Z-Wave Single Chips
SD3502	General Purpose Wireless Z-Wave Chip, QFN48 (Full IO chip)
SD3503	Wireless Z-Wave Modem SOC, QFN32 (Modem chip)
SDK	Software Development Kit
SoC	System on Chip
SPI	Serial Peripheral Interface
SW	Software
Tx	Transmit
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
ZDB5xxx	Range of 500 series Z-Wave Development Modules
ZDB5101	Z-Wave Development Module containing a ZM5101 General Purpose Z-Wave SiP Module
ZDB5202	Z-Wave Development Module containing a ZM5202 General Purpose Z-Wave Module
ZDB5304	Z-Wave Development Module containing a ZM5304 Z-Wave Serial Interface Module with Antenna
ZDP02A	Z-Wave Development Programmer, version 02A. This version is no longer supported by Sigma Designs.
ZDP03A	Z-Wave Development Programmer, version 03A Supports Sigma Designs Z-Wave chips series 100/200/300/400/500
ZM1206	6 cm² Z-Wave Module containing a 100 Series Z-Wave Single Chip
ZM1220	20 cm² Z-Wave Module containing integrated PCB antenna, EEPROM and 100 Series Z-Wave Single Chip
ZM2102	2 cm² Z-Wave Module containing a 200 Series Z-Wave Single Chip
ZM2106	6 cm² Z-Wave Module containing a 200 Series Z-Wave Single Chip
ZM2120	20 cm² Z-Wave Module containing a 200 Series Z-Wave Single Chip

Abbreviation	Explanation
ZM2120C	20 cm² Converter Z-Wave Module, used to convert ZM2106 to ZM2120-formfactor
ZM3102N	2 cm² Z-Wave Module containing a 300 Series Z-Wave Single Chip
ZM3106C	6 cm² Z-Wave Module containing a 300 Series Z-Wave Single Chip
ZM3120	20 cm² Z-Wave Module containing a 300 Series Z-Wave Single Chip
ZM3120C	20 cm² Converter Z-Wave Module, used to convert ZM3106C to ZM3120-formfactor
ZM4225	25 cm² Z-Wave Module containing integrated PCB antenna, EEPROM and Z-Wave SD3402 Single Chip (QFN)
ZM5xxx	Range of 500 series Z-Wave Modules
ZM5101	General Purpose Z-Wave® SiP Module (Full IO module)
ZM5202	General Purpose Z-Wave® Module (Basic IO module)
ZM5304	Z-Wave® Serial Interface Module with Antenna (Modem module)
ZW0102	100 Series Z-Wave Soc
ZW0201	200 Series Z-Wave Soc
ZW0301	300 Series Z-Wave Soc
ZW040x	400 Series Z-Wave Soc
ZW050x	500 Series Z-Wave Soc

2 INTRODUCTION

2.1 Purpose

The purpose of this document is to present the Sigma Designs Z-Wave chip programming environment consisting of the ZDP03A programming hardware and the PC Programmer application.

The Z-Wave PC Programmer can program the complete range of Sigma Designs Z-Wave chips and modules via the ZDP03A programming hardware. The programming environment also supports programming of External NVM on Z-Wave modules. Finally, the programming environment can be used to configure RF settings and Lock Bits in Z-Wave modules.

500 series chip USB and UART programming modes are supported by the Z-Wave PC Programmer.

ZW040x and ZW050x chips may need two types of calibration for the Z-Wave chip to operate perfectly. While **Tx calibration** is handled automatically by the Z-Wave PC Programmer application, **crystal calibration** requires dedicated calibration hardware. For more details on crystal calibration, refer to section 5.4.5.

2.2 Audience and prerequisites

The audience is Z-Wave partners and Sigma Designs.

3 GETTING STARTED

3.1 Installation

The ZDP03A programming hardware can be used for programming Sigma Designs Z-Wave 100/200/300/400/500 Series Chips. The ZDP03A is intended for use during SW and HW development and for small production series. A Z-Wave development module can be mounted in the Z-Wave module socket on the ZDP0xx programming hardware and product specific hardware may be programmed via an ISP cable. The ZDP03A programming hardware is connected to the PC via a USB interface. An external 9VDC 500mA power supply must always be used for the ZDP03A.

ALWAYS install the USB driver firmware in the PC BEFORE connecting the USB cable to the ZDP03A. ALWAYS check that the jumpers are mounted correctly on the ZDP03A programming hardware before applying power.

The Z-Wave PC Programmer supports three programming interfaces:

- SPI, via ZDP03A (ZW020x/ZW030x/ZW040x/ZW050x)
- UART, directly to end product or via ZDP03A (ZW050x)
- USB, directly to end product or via ZDP03A (ZW050x)

The ZDP03A provides USB and UART interfaces mapping directly to the USB and TTL level UART interfaces of Sigma Designs ZW050x Z-Wave modules. Further, the ZDP03A may be configured so that the USB interface acts as a USB to UART converter; connecting the PC USB port to the UART interface of the Z-Wave module. The interfaces may also be configured for direct in-system programming, e.g. for testing the programming of end product firmware before hardware is available for such products.

As the SPI interface is the most versatile, it is recommended to use the SPI interface during general software development in ZDP03A compatible Z-Wave modules.

3.1.1 System Requirements

Minimum PC requirements to run Z-Wave PC Programming application are as follows:

- Windows XP, Service Pack 2
- Microsoft .NET Framework 2.0 or later
- USB
- Serial port (optional for UART programming)

3.1.2 ZDP03A USB Driver installation

The USB driver firmware MUST be installed in the PC before connecting the USB cable to the ZDP03A.

Follow the step-by-step instructions below.

1. Install the CP210x VCP driver.

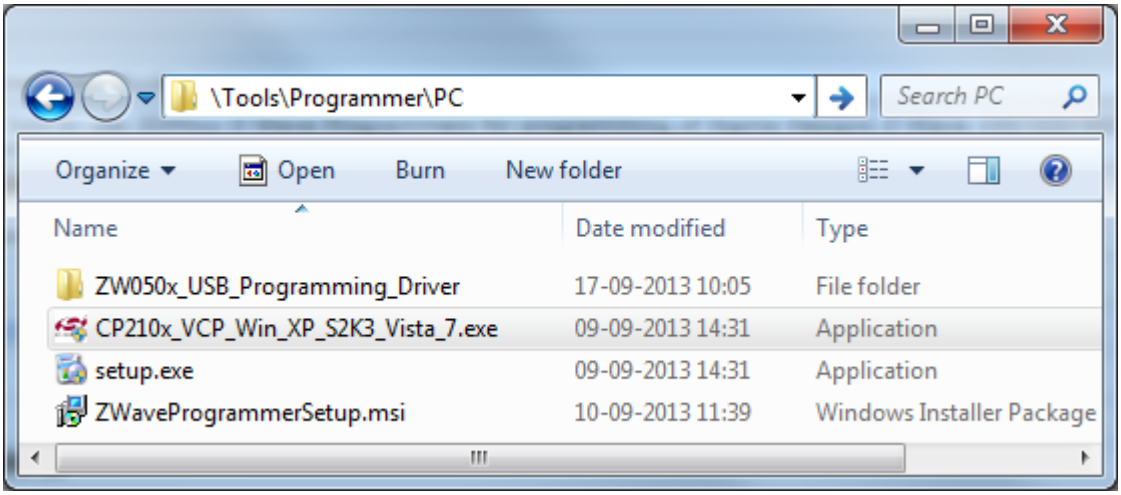


Figure 1. ZDP03A USB driver location

To run the CP210x VCP driver installation wizard, run `C:\DevKit_X_YY\Tools\Programmer\PC\CP210x_VCP_Win_XP_S2K3_Vista_7.exe` and follow the installation wizard. It is RECOMMENDED to not change the suggested destination folder. If changing the destination folder, the new folder name should be recorded for later use.

2. Check the jumper settings on the ZDP03A; refer to section 3.2.
3. Connect the external 9VDC power supply.
4. Connect the ZDP03A to the PC using the USB cable (USB interface “J1” on the ZDP03A).

Windows 7 will detect the new hardware and the new hardware wizard will start. Once completed, the driver installation is confirmed. The reported COM port number may vary.

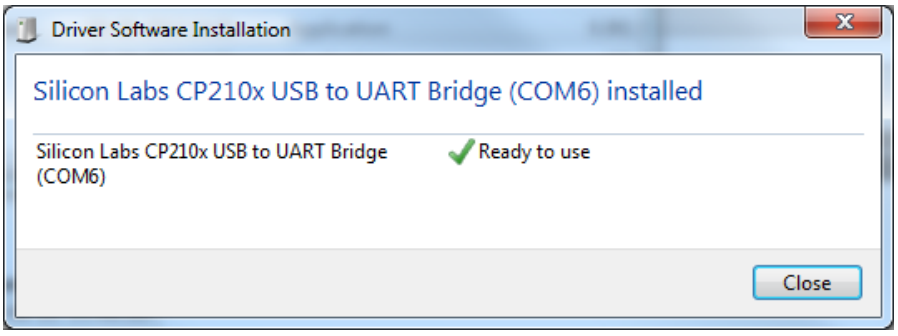


Figure 2. ZDP03A USB driver installation confirmation

5. Restart the PC when the USB driver installation is finished if prompted to do so.

3.2 ZDP03A Programming hardware

The ZDP03A Z-Wave programming hardware can be used for programming Z-Wave 100/200/300/400/500 Series Chips.

ALWAYS install the USB driver firmware in the PC BEFORE connecting the USB cable to the ZDP03A.
ALWAYS check that the jumpers are mounted correctly on the ZDP03A before applying power.

3.2.1 ZDP03A Connectors, LEDs and jumpers

The ZDP03A is connected to the PC using the USB interface “J1” and powered by an external 9VDC power supply. The embedded application may be debugged while the Z-Wave module is mounted in the Z-Wave module connector of the ZDP03A.

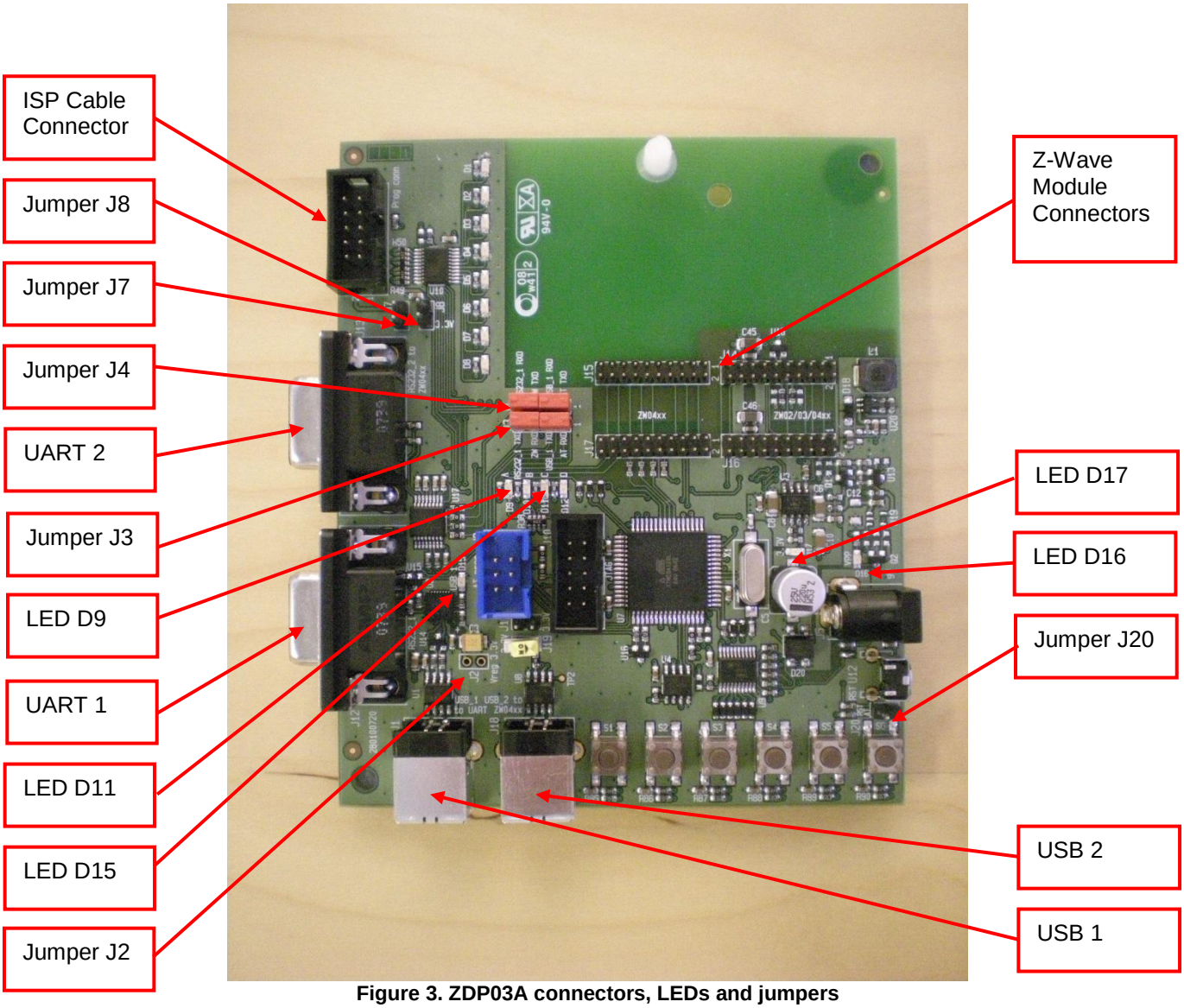


Figure 3. ZDP03A connectors, LEDs and jumpers

The following Jumpers must be correctly configured. Refer to Figure 3.

Table 1. ZDP03A configuration jumpers

LED	Function	Options
J2	ZDP03A Power via USB	: Disable (RECOMMENDED) : Enable
J3+J4	UART connector selection	ZDP03A UART 1 directly to module UART ZDP03A USB 1 via CP210 to module UART
J7	OTP programming enable (ZW0400 only)	: Disable : Enable (RECOMMENDED)
J20	ATmega128 reset (LED D9 turns off)	: No reset (RECOMMENDED) : Reset

The following LEDs indicate the state of the programmer.

Table 2. ZDP03A status LEDs

LED	On State	Off State
D9	ZDP03A firmware is up and running	ZDP03A firmware is halted
D11	ZDP03A is busy	ZDP03A is idle
D15	ZDP03A and host PC are connected via USB	ZDP03A USB un-connected
D16	External power on	External power off
D17	USB power on	USB power off

The figure below shows the ZDP03A with a ZW050x Z-Wave module mounted in the Z-Wave Module connector.

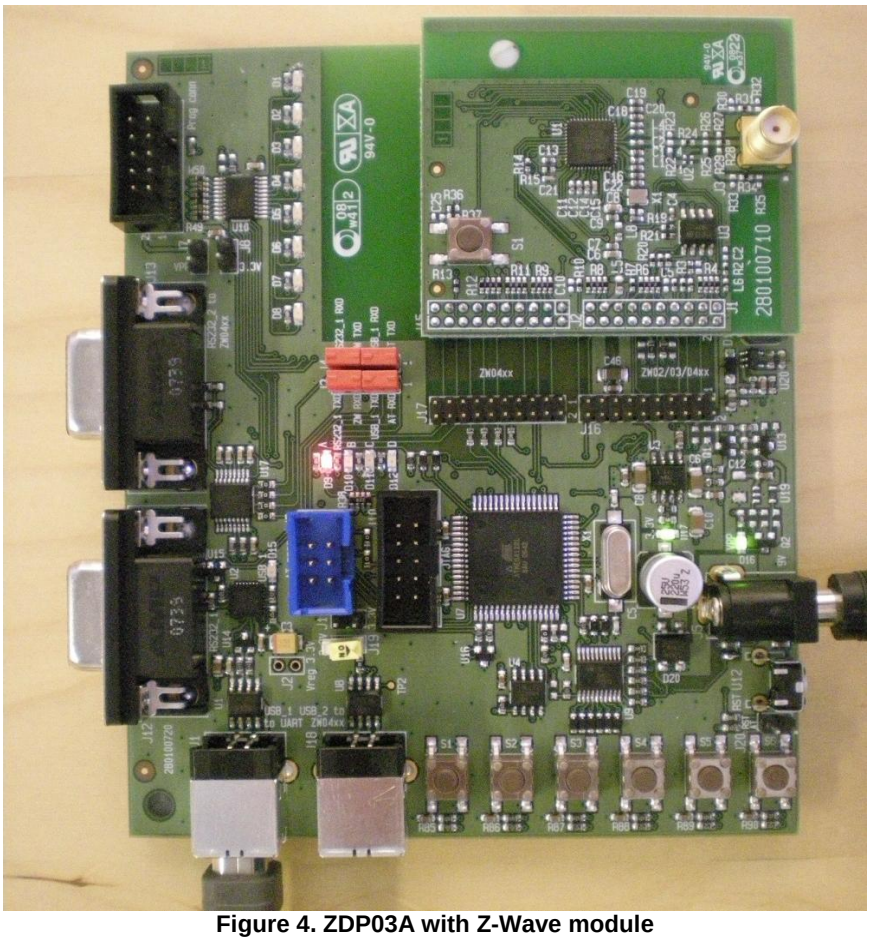


Figure 4. ZDP03A with Z-Wave module

3.3 Z-Wave PC Programmer installation

The Z-Wave Programmer software is installed by browsing to the `C:\DevKit_X_YY\Tools\Programmer\PC\` directory and double click on the **setup.exe** file.

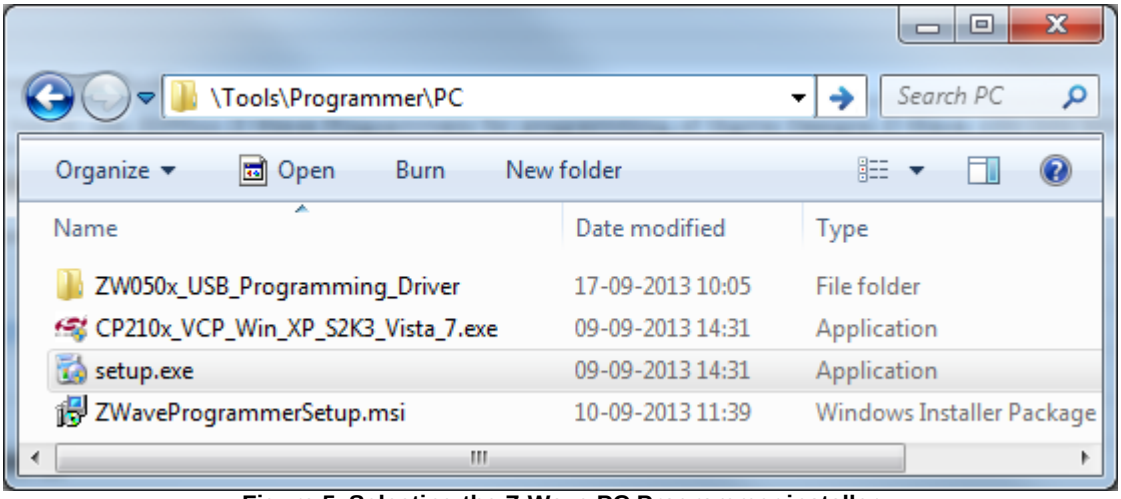


Figure 5. Selecting the Z-Wave PC Programmer installer

The “Welcome” dialog appears.

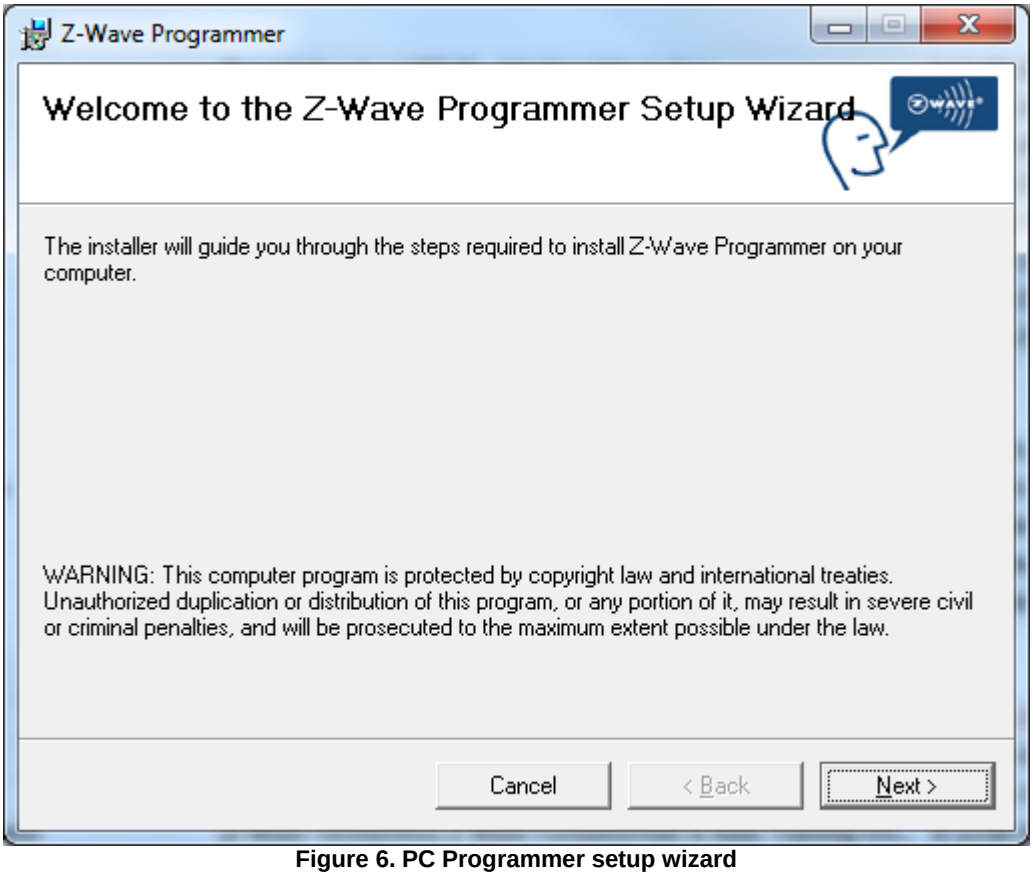
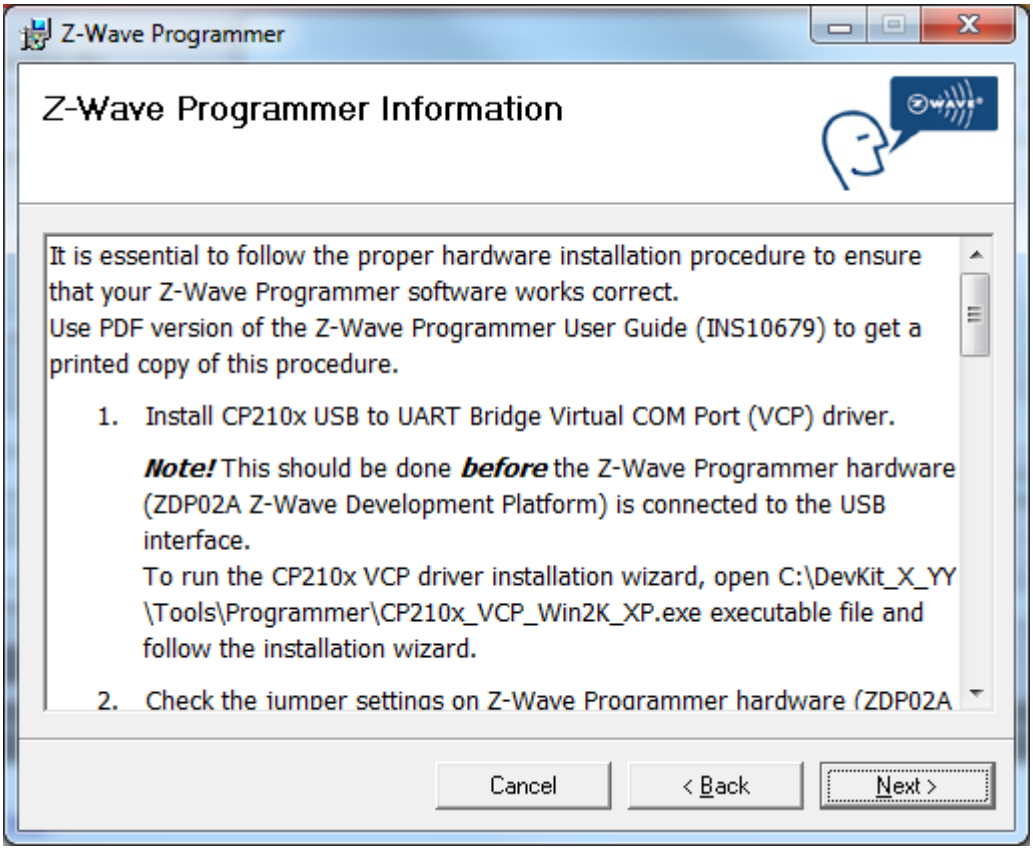


Figure 6. PC Programmer setup wizard

Click on the ‘**Next>**’ button.



Click on the '**Next>**' button.

The Installation Folder dialog appears.

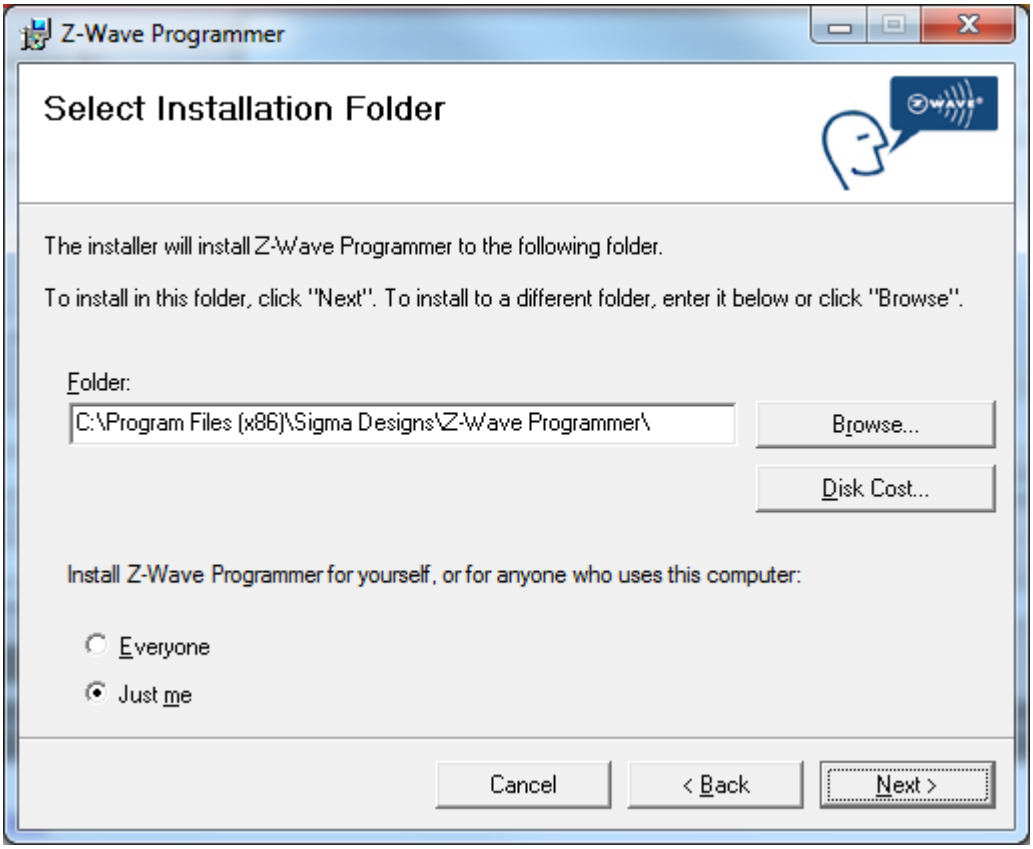
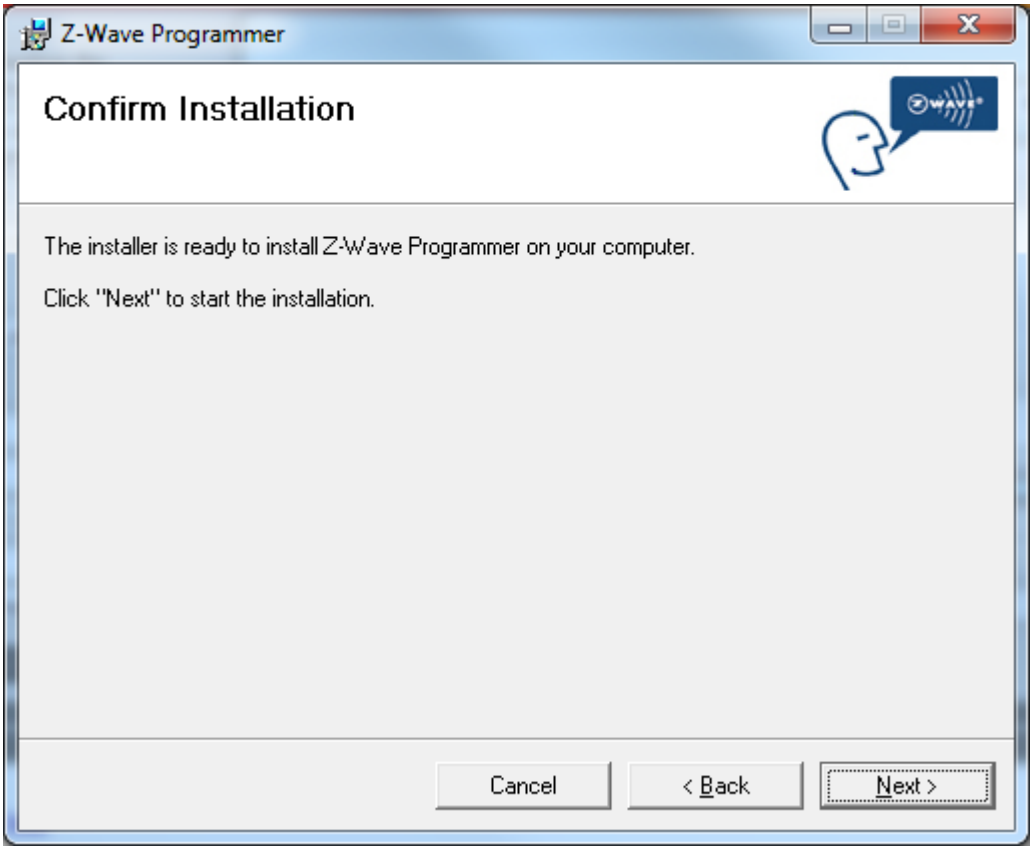


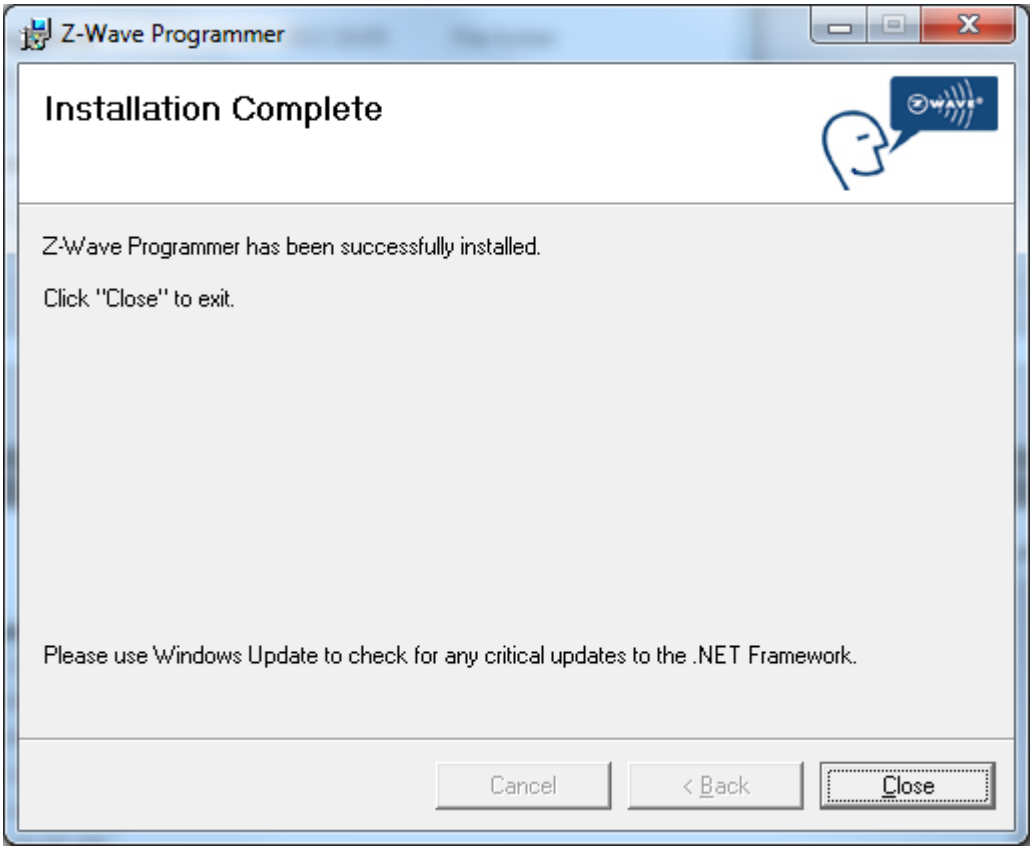
Figure 7. PC Programmer installation folder

It is RECOMMENDED to use the suggested default folder path.

Click on the '**Next>**' button.



Click on the '**Next>**' button. The wizard starts installing the Z-Wave PC Programmerapplication.



Click on the '**Close**' button to exit the installation wizard.

Run the Z-Wave PC Programmer from the Windows start menu:

Start->All Programs->Sigma Designs->Z-Wave Programmer.

From the Z-Wave PC Programmer main menu, choose the USB interface used by the ZDP03A programming hardware:

Z-Wave PC Programmer->Settings.

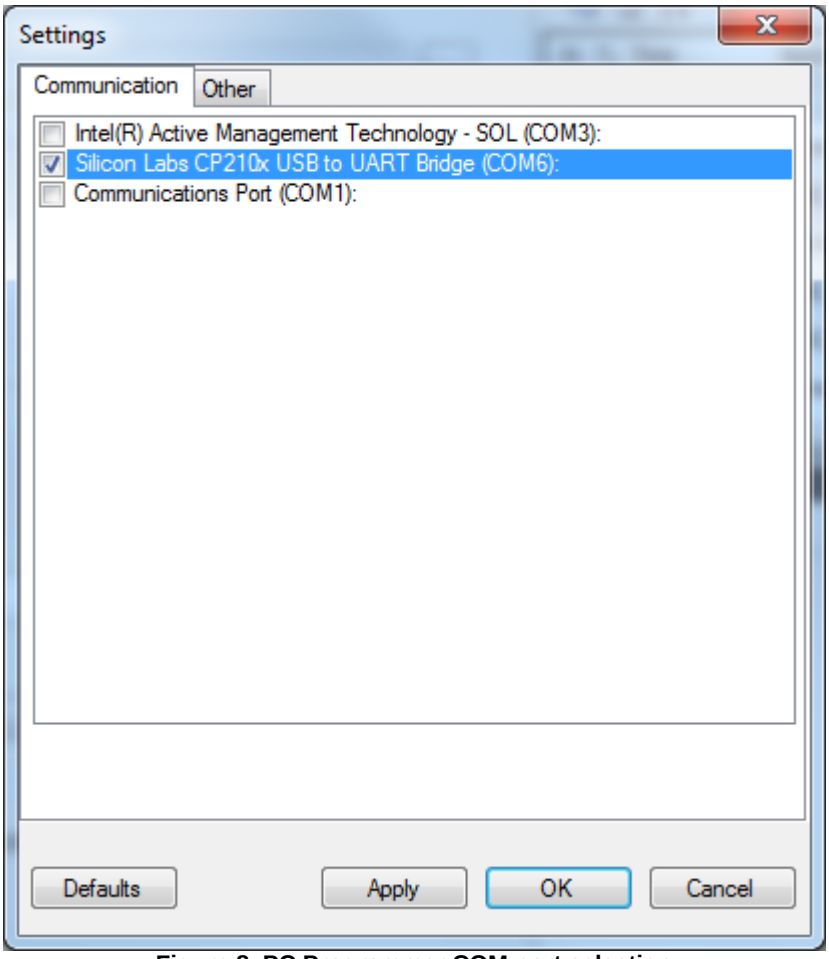


Figure 8. PC Programmer COM port selection

4 Z-WAVE PC PROGRAMMER GUI

The Z-Wave PC Programmer application implements a Graphical User Interface (GUI) as well as a console interface. Both user interfaces may be used for programming the internal and External NVM of Z-Wave chips. This chapter focuses on the GUI. Refer to section 5.10 for a description of the console interface.

Figure 9 shows the Z-Wave PC Programmer main window.

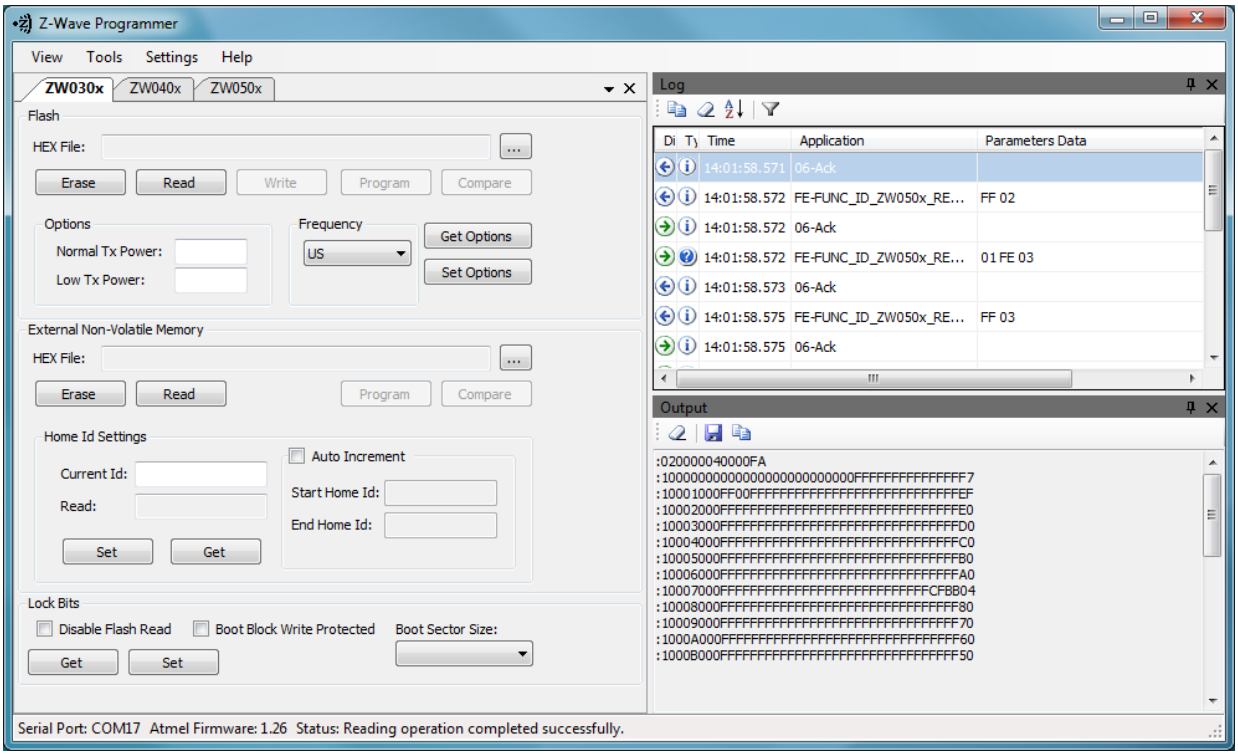


Figure 9. Z-Wave PC Programmer main window

The Z-Wave PC Programmer window is organized in a number of views.

The tab strip in the top of the left view is for selection of the actual chip type. The contents of the left view depend on the selected tab. The “Log” and “Output” views in the right half of the window are the same for all series of Z-Wave chips.

Notice that the Output window MUST be opened before using any read buttons.

4.1 Main Window Menu

The main window offers the following menu:

Table 3. Z-Wave PC Programmer menu

Menu item	Description
View	Configuration of GUI components.
Tools	Tools for • Chip detection • ZDP03A firmware update • Module calibration • Module reset
Settings	Opens up a “Settings” dialog with two tabs: “Communication” and “Other”. The “Communication” tab allows for selection of the USB port used by the ZDP03A programming hardware while the “Other” tab gives access to advanced settings.
Help	Help resources and application version information.

4.2 Log View

All actions are logged here.

4.3 Output View

A user may request that the current NVM contents are read back from the Z-Wave module. The Output view shows the data that were read back. The user may save the contents of the Output View as a HEX file.

4.4 The ZW050x view

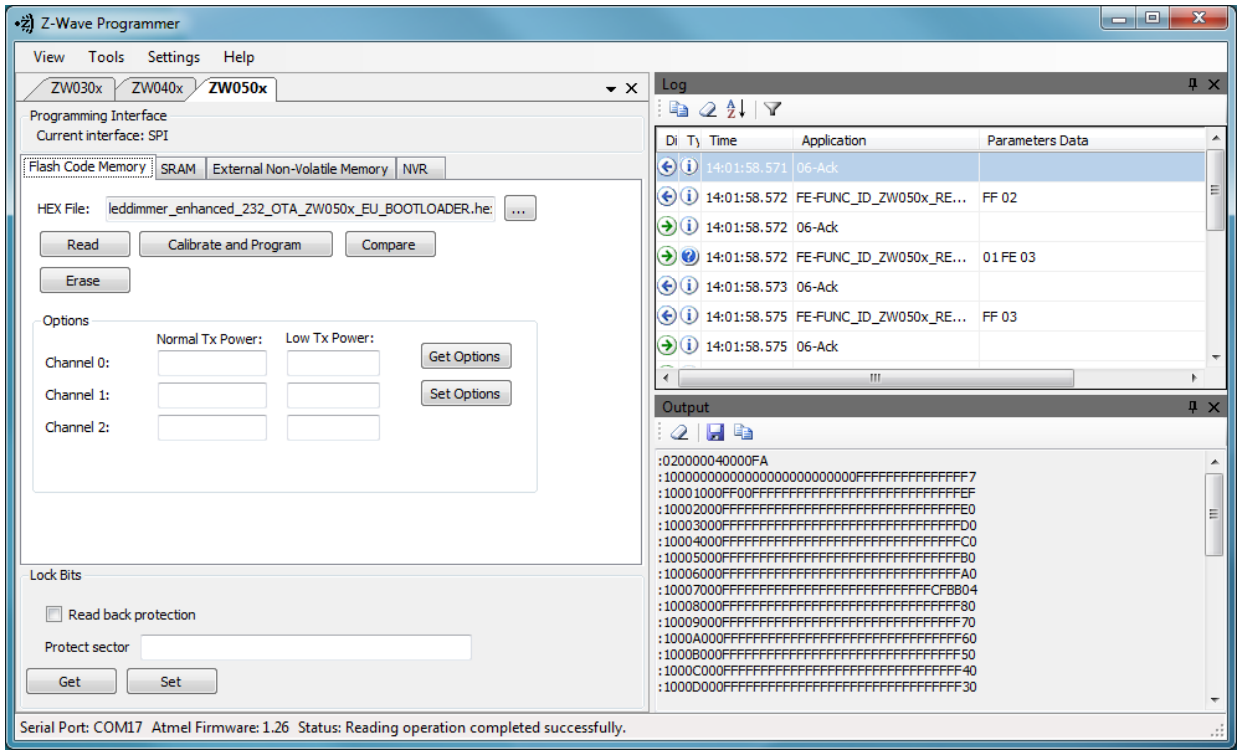


Figure 10. ZW050x::Flash Code Memory view

Note: If using the ZDP03A Z-Wave programming hardware for programming Z-Wave 500 series chips, the ZDP03A MUST run firmware v1.26 or newer.

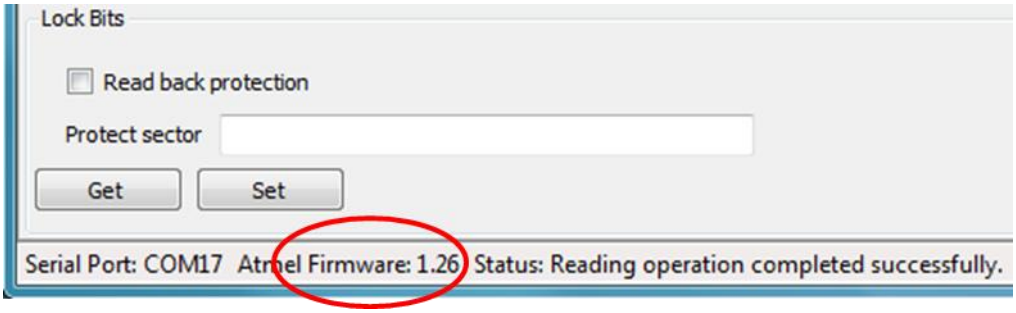


Figure 11. Reading ZDP03A firmware version from status bar of the PC Programmer main window

The ZW050x view is organized as three views, each showing one memory type along with a number of permanently visible GUI blocks. The actual ZW050x view is selected from a second tab strip just under the chip type tab strip. Each block is presented in the following.

4.4.1 Programming Interface Block

The PC Programmer Application supports three programming modes; SPI, UART and USB.
The SPI interface SHOULD be used during general software development.

4.4.2 ZW050x::Flash Code Memory View

The ZW050x Flash Code memory view is outlined in Figure 10. The **Flash** block comprises the following GUI elements:

- **HEX File** textbox (path to HEX file)
- **Erase** button (erase the Internal NVM)
- **Read** button (read the Internal NVM)
- **Calibrate and Program** button (Calibrate Tx frequencies and write HEX file to Internal NVM)[1]
- **Verify** button (compare the Internal NVM contents to a HEX file)
- **Normal TX Power** and **Low TX Power** textbox (Refer to Table 8)
- **Get Options** button (read current TX power and frequency)
- **Set Options** button (write TX power and frequency to the Internal NVM)

The Erase button SHOULD NOT be used prior to the programming of a new software image to a ZW050x Z-Wave module. The erase operation not only clears the Flash code memory block but also the NVR page. The NVR page holds certain module specific data such as crystal calibration constants. If the NVR is cleared, the Z-Wave PC Programmer will need dedicated calibration hardware to perform a chip crystal calibration. Furthermore, it is also necessary to restore the remaining NVR content such as UUID. Refer to section 5.4.5.

4.4.3 ZW050x::SRAM View

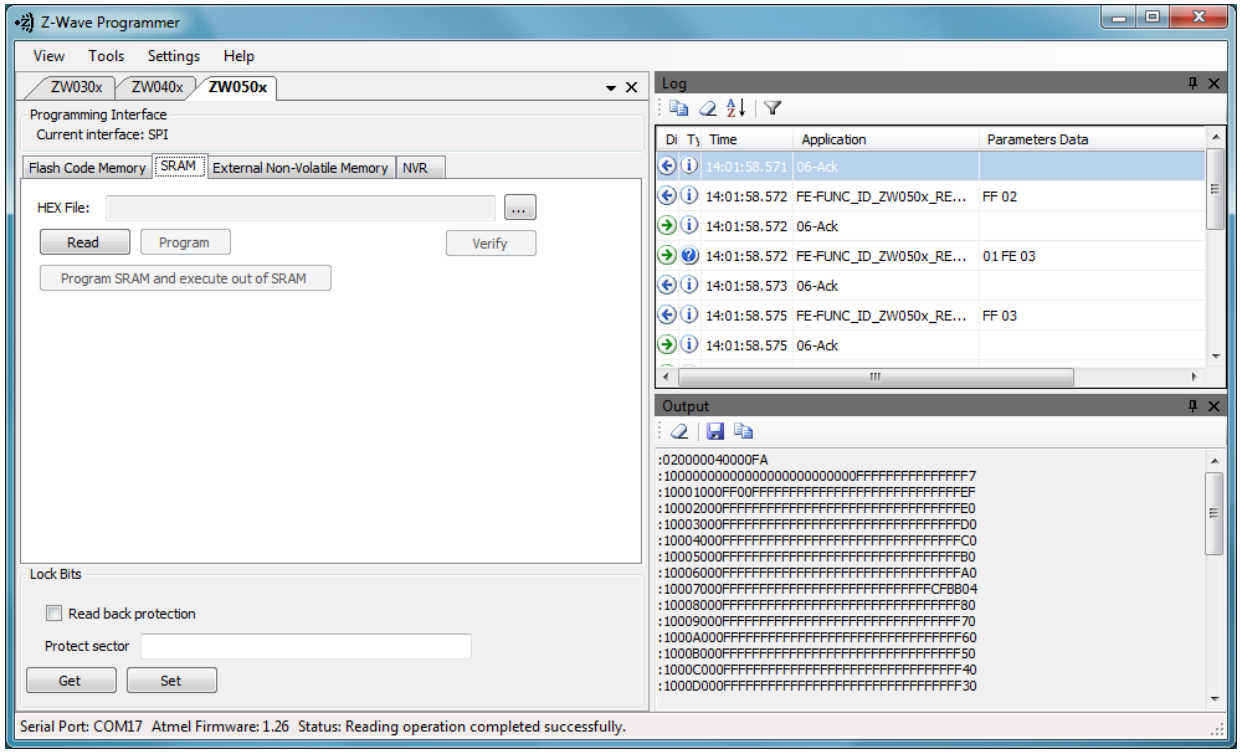


Figure 12. ZW050x::SRAM view

[1] Dedicated calibration hardware may be required for chip crystal calibration. Refer to section 5.4.5.

The SRAM view comprises the following GUI elements:

- **HEX File** textbox (path to HEX file)
- **Read** button (read the SRAM contents)
- **Program** button (write HEX file to the SRAM)
- **Verify** button (compare the SRAM contents to a hex file)

4.4.4 ZW050x::External NVM View

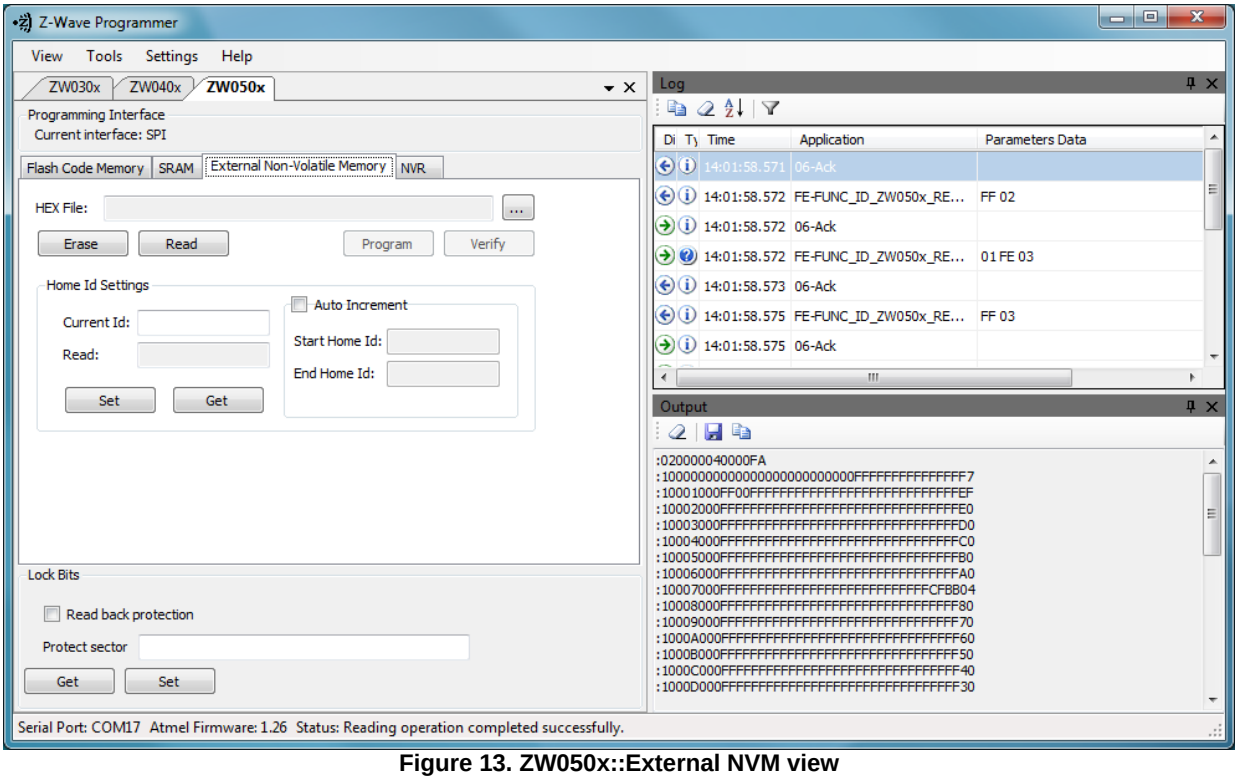


Figure 13. ZW050x::External NVM view

The **External NVM** (Non-Volatile Memory) view comprises the following GUI elements:

- **HEX File** textbox (path to HEX file)
- **Erase** button (erase the External NVM)
- **Read** button (read the External NVM)
- **Program** button (write HEX file to the Internal NVM)
- **Verify** button (compare the External NVM contents to a HEX file)
- **Home ID Settings** (configure the Home ID of the Z-Wave module)

4.4.5 ZW050x::NVR View

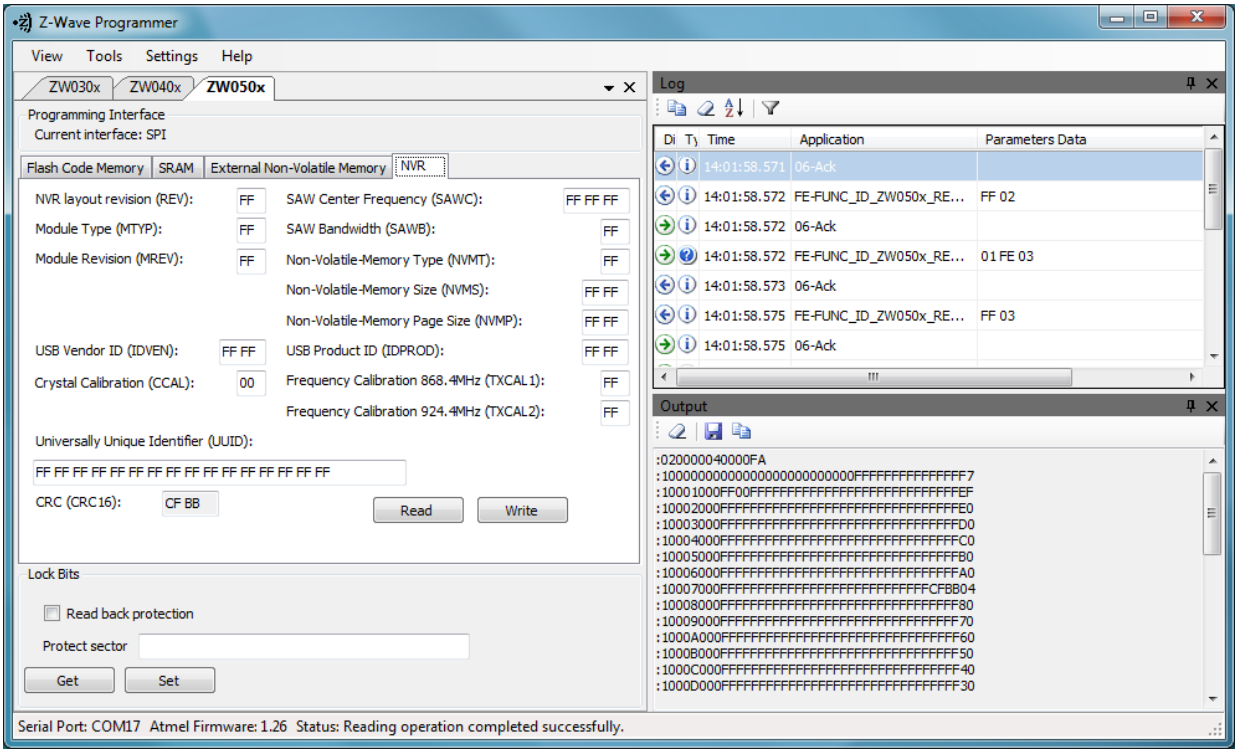


Figure 14. ZW050x::NVR view

The NVR area of the Z-Wave 500 Series chip holds information which can be manipulated via the Z-Wave PC Programmer application. NVR registers available via the PC Programmer includes the Universally Unique Identifier (UUID) used by USB enabled Z-Wave products.

NVR registers are preserved if the Flash Code Memory is reprogrammed, but they are reset to their default values when the Flash Code Memory is erased. The Erase button of the “Flash Code Memory” view SHOULD NOT be used prior to the programming of a new software image to a ZW050x Z-Wave module. The erase operation not only clears the Flash code memory block but also the NVR page. The NVR page holds certain module specific data such as crystal calibration constants. If the NVR is cleared, the Z-Wave PC Programmer needs dedicated calibration hardware to perform a crystal calibration. Furthermore, it is also necessary to restore the remaining NVR content. Refer to section 5.4.5.

4.4.6 Lock Bits Block

The Lock Bits block comprises the following GUI elements:

- **Read back protection** checkbox
- **Protect sector** textbox, contains protected sector numbers (0 - 63)
  - Type sector numbers and/or sector ranges separated by commas. For example: type 0, 1, 3, 6, 17-63
- **Get** button (reads lock bits from the chip)
- **Set** button (writes lock bits to the chip)

The encoding of the lock bits is outlined in **Error! Reference source not found.**.

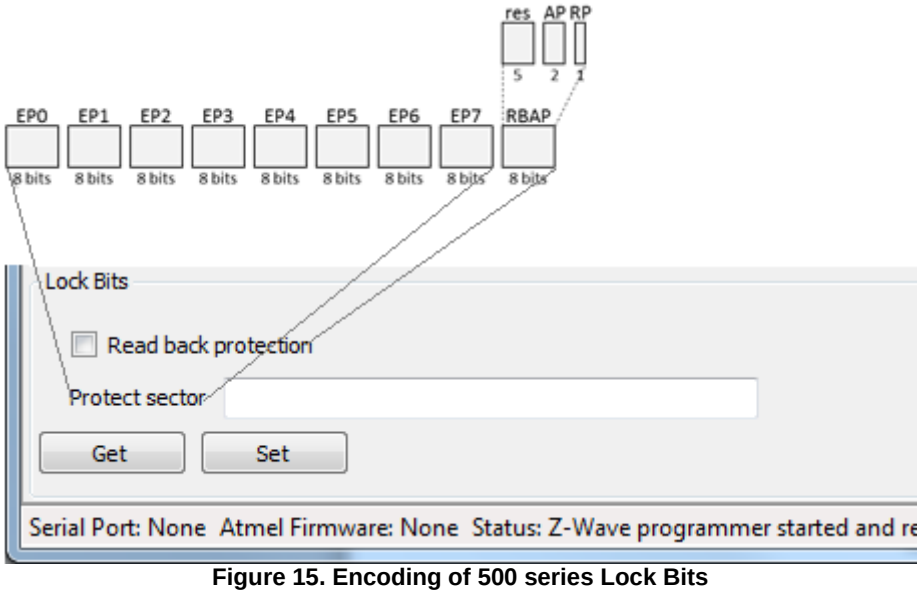


Figure 15. Encoding of 500 series Lock Bits

Details may be found in Table 4.

Lock bits are active low. As an example, setting bit 1 of EP1 to ‘0’ enables Erase Protection for Flash Code sector 9.

Table 4. 500 series Lock Bits

Byte	Long Name	Description
EP0	Erase Protection 0	Protects sectors 0..7
EP1	Erase Protection 1	Protects sectors 8..15
EP2	Erase Protection 2	Protects sectors 16..23
EP3	Erase Protection 3	Protects sectors 24..31
EP4	Erase Protection 4	Protects sectors 32..39
EP5	Erase Protection 5	Protects sectors 40..47
EP6	Erase Protection 6	Protects sectors 48..55
EP7	Erase Protection 7	Protects sectors 56..63
RBAP	Read Back & Auto Programming	RP (bit 0): Read Back Protection - Enables read back protection if set to ‘0’
		AP (bits 2..1): Auto Programming mode - Controlled by the PC Programmer. User input is ignored for these bits. Notice: User can’t see/change these bits in the GUI.
		Res (bits 7..3): Reserved for future use Notice: User can’t see/change these bits in the GUI.

The Set button cannot clear lock bits if they have already been set. A user can only clear lock bits and the Read Back Protection bit by writing an entirely new image to the chip, thus clearing the confidential contents that once resided in the module.

For programming of ZW050x chips in a production setup, refer to [6] & [7].

4.5 The ZW040x view

The ZW040x view is outlined in Figure 16.

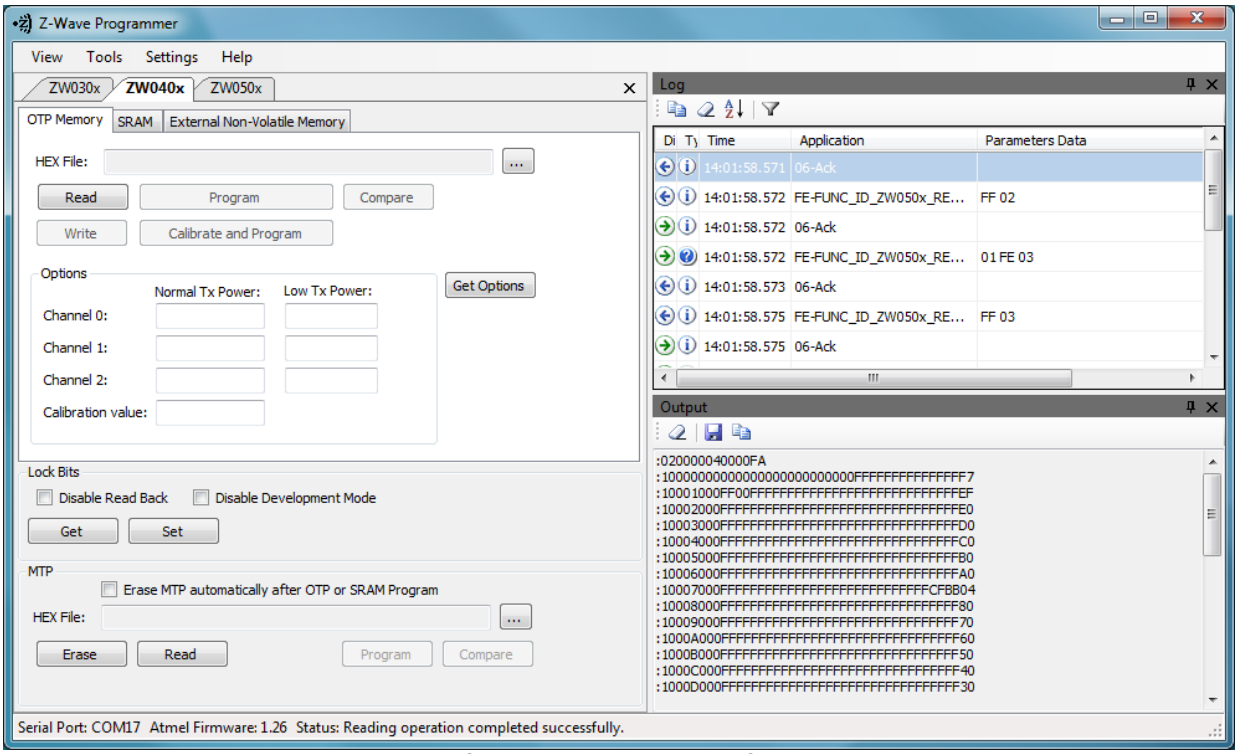


Figure 16. ZW040x::OTP view

The ZW040x view is organized as three sub-views, each showing one memory type along with a number of permanently visible GUI blocks. The actual ZW040x view is selected from a second tab strip just under the chip type tab strip. Each block is presented in the following.

4.5.1 ZW040x::OTP Memory View

The ZW040x::OTP view can be seen in Figure 16. The 400 series chip Internal NVM memory is of the OTP (One-Time-Programmable) type. Therefore, this view has no **Erase** button.

The **OTP Memory** view comprises the following GUI elements:

- **HEX File** textbox (path to HEX file)
- **Read** button (read Internal NVM)
- **Write** button (write HEX file to Internal NVM)
- **Program** button (write and verifies HEX file to Internal NVM)
- **Compare** button (compare Internal NVM contents to a HEX file)
- **Calibration and Program** button (crystal calibrate and write HEX file to Internal NVM)[2]
- **Normal TX Power** and **Low TX Power** textbox (Refer to Table 7);
- **Get Options** button (read current TX power and RF settings);
- **Lock Bits,** the same as in ZW020x-ZW030x tabs. Refer to section 4.6.3.

[2] Dedicated calibration hardware required. Refer to section 5.4.5.

4.5.2 ZW040x::SRAM View

The 400 series chip Internal NVM memory is of the OTP type. To assist debugging in an OTP environment, the 400 series chip also implements a block of SRAM for code execution. The SRAM view can be seen in Figure 17.

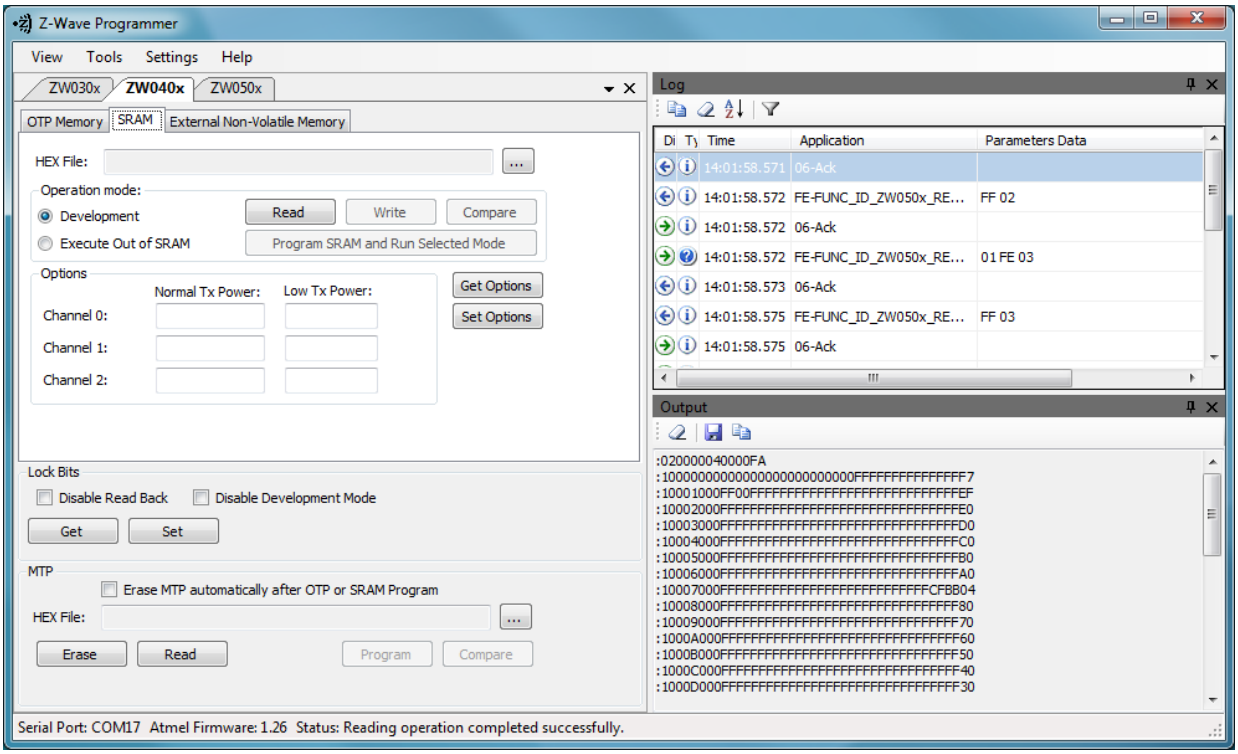


Figure 17. ZW040x::SRAM view

The **SRAM** view comprises the following GUI elements:

- **HEX File** textbox (path to HEX file)
- **Operation Mode** radio buttons:
  - Development
  - Execute out of SRAM
- **Read** button (read SRAM contents)
- **Write** button (write HEX file to SRAM)
- **Compare** button (compare SRAM contents to a HEX file)
- **Program SRAM and Run Selected Mode button**
- **Normal TX Power and Low TX Power** textboxes (Refer to Table 7)

4.5.3 ZW040x::External NVM View

The External NVM View can be seen in Figure 18.

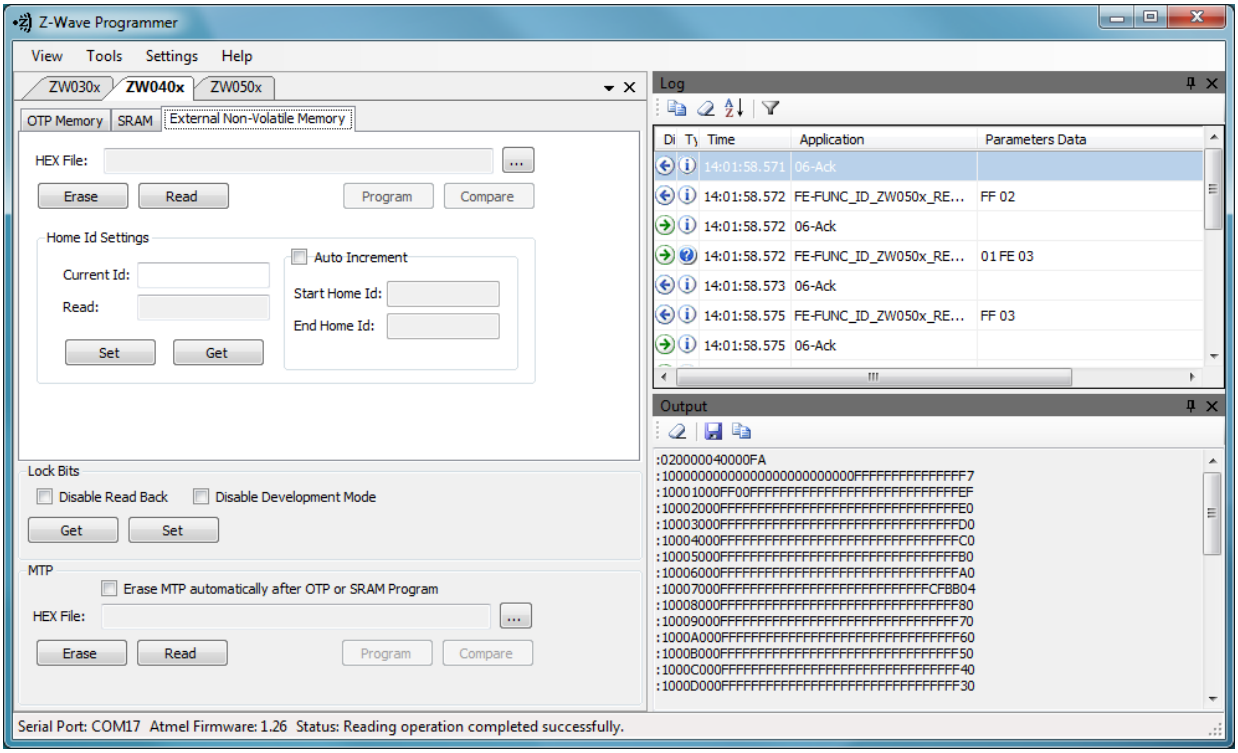


Figure 18. ZW040x::External NVM view

The External NVM (Non-Volatile Memory) view comprises the following GUI elements:

- **HEX File** textbox (path to HEX file)
- **Erase** button (erase External NVM)
- **Read** button (read External NVM)
- **Program** button (write and verify HEX file to External NVM)
- **Compare** button (compare External NVM contents to a HEX file)
- **Home ID Settings** (configure the Home ID of the Z-Wave module)

4.5.4 Lock Bits Block

The Lock Bits block comprises the following GUI elements:

- **Disable Read Back** checkbox (prevent a user from reading back chip memory contents)
- **Disable Development Mode** checkbox (prevent a user from selecting development mode)
- **Get** button (read lock bits from chip)
- **Set** button (write lock bits to chip)

The Set button cannot clear lock bits if they have already been set.
A user can only clear lock bits by writing an entirely new image to the chip, thus clearing the confidential contents that once resided in the chip.

4.5.5 MTP Block

The **MTP** (Many Time Programmable) memory block comprises the following GUI elements:

- **HEX File** textbox (path to HEX file)
- **Erase** button (erase MTP memory)
- **Read** button (read MTP memory)
- **Program** button (write and verify HEX file to MTP memory)
- **Compare** button (compare MTP memory contents to a HEX file)

4.6 TheZW020x and ZW030x views

The ZW020x and ZW030x views have a unified layout as outlined in Figure 19.

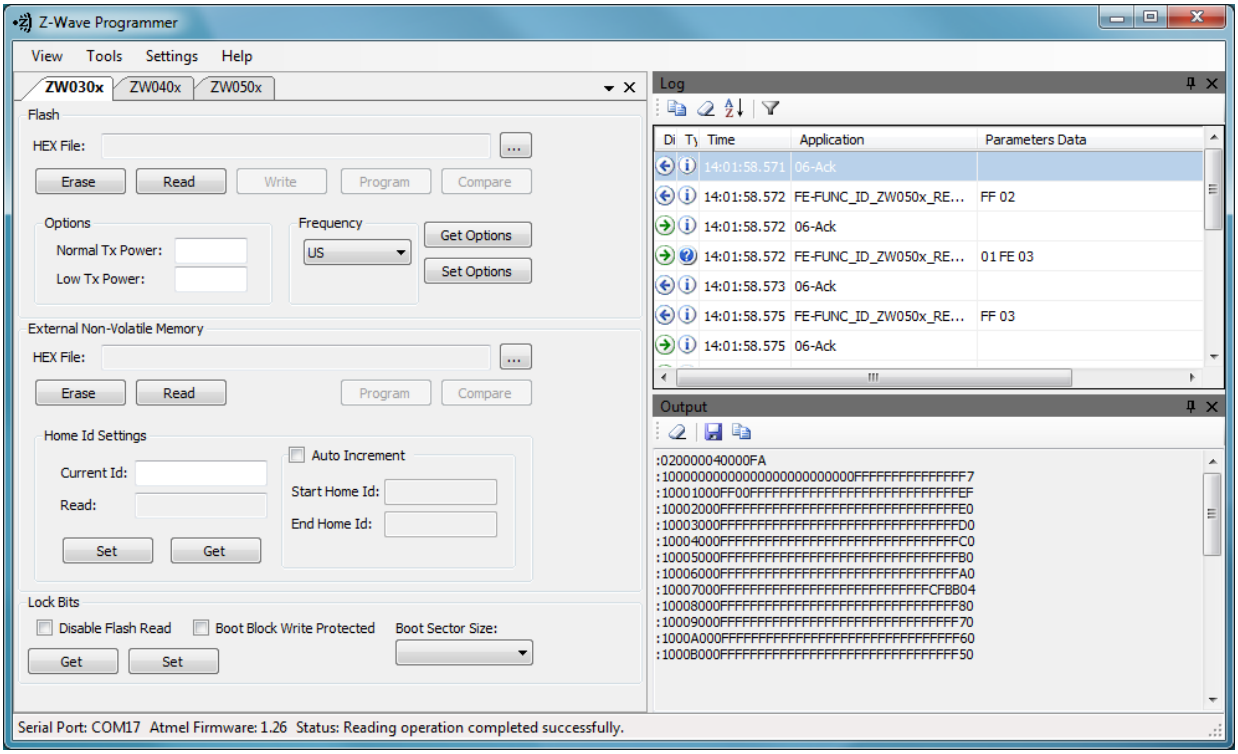


Figure 19. ZW030x view

Each block of the view is presented in the following.

4.6.1 Flash Block

The **Flash** block comprises the following GUI elements:

- **HEX File** textbox (path to HEX file)
- **Erase** button (erase Internal NVM)
- **Read** button (read Internal NVM)
- **Write** button (write HEX file to Internal NVM)
- **Program** button (write HEX file to Internal NVM and verify that data was written correctly)

- **Compare** button (compare Internal NVM contents to a HEX file)
- **Normal TX Power** and **Low TX Power** textbox (refer to Z-Wave API documentation)
- **Frequency** dropdown list (selection of regions)
- **Get Options** button (read current TX power and frequency)
- **Set Options** button (write TX power and frequency to Internal NVM)

4.6.2 External NVM Block

The **External NVM** block comprises the following GUI elements:

- **HEX File** textbox (path to HEX file)
- **Erase** button (clear External NVM)
- **Read** button (read External NVM contents)
- **Program** button (write HEX file to External NVM)
- **Compare** button (compare External NVM contents to a HEX file)
- **Home ID settings** (configure the Home ID of the Z-Wave module)

4.6.3 Lock Bits Block

The Lock Bits block comprises the following GUI elements:

- **Disable Flash Read** checkbox (prevent a user from reading back Internal NVM contents)
- **Boot Block Write Protected** checkbox (prevent a user from overwriting the boot block)
- **Boot Sector Size** dropdown list
- **Get** button (read lock bits from chip)
- **Set** button (write lock bits to chip)

The Set button cannot clear lock bits if they have already been set.
A user can only disable lock bits by writing an entirely new image to the chip, thus clearing the confidential contents that once resided in the chip.

5 PROGRAMMING

5.1 Programming setups

Figure 20 outlines a typical SDK programming setup:

- ZDP03A Z-Wave programming hardware
- A Z-Wave module mounted in the Z-Wave module socket of the ZDP03A
- A Windows PC running the Z-Wave PC Programmer application
- A USB cable

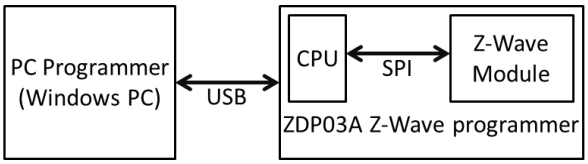


Figure 20. Programming a Z-Wave module via ZDP03A SPI interface

Alternatives include the following configurations:

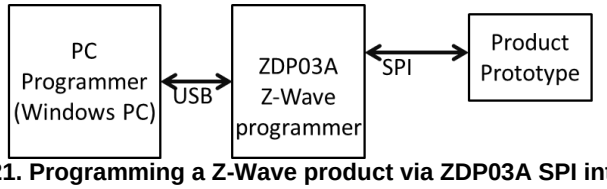


Figure 21. Programming a Z-Wave product via ZDP03A SPI interface

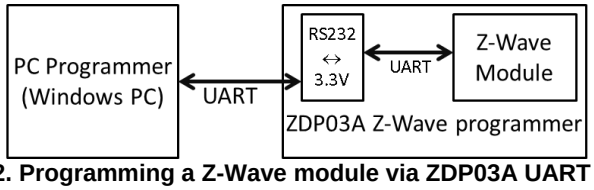


Figure 22. Programming a Z-Wave module via ZDP03A UART interface

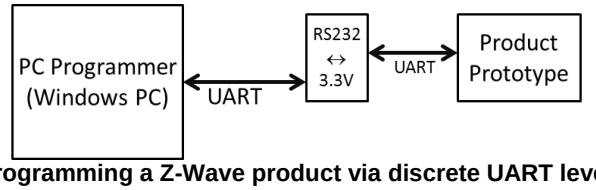


Figure 23. Programming a Z-Wave product via discrete UART level converter

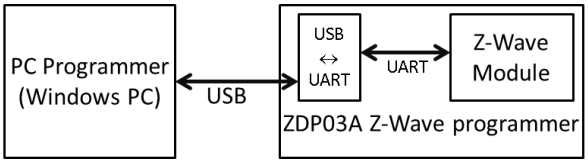


Figure 24. Programming a Z-Wave product via built-in ZDP03A USB to UART converter

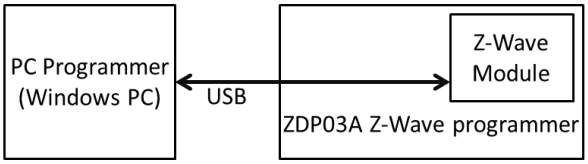


Figure 25. Programming a Z-Wave module via ZDP03A USB interface

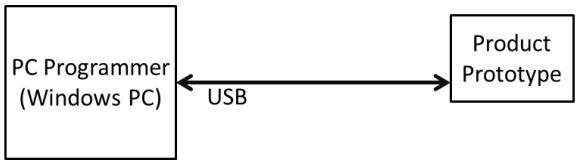


Figure 26. Programming a Z-Wave product via USB interface

5.2 Configuration

5.2.1 Selecting a COM port

Select the PC interface port assigned to the ZDP03A Z-Wave programming hardware by Windows:

Open Settings->Communication and select the appropriate COM port.

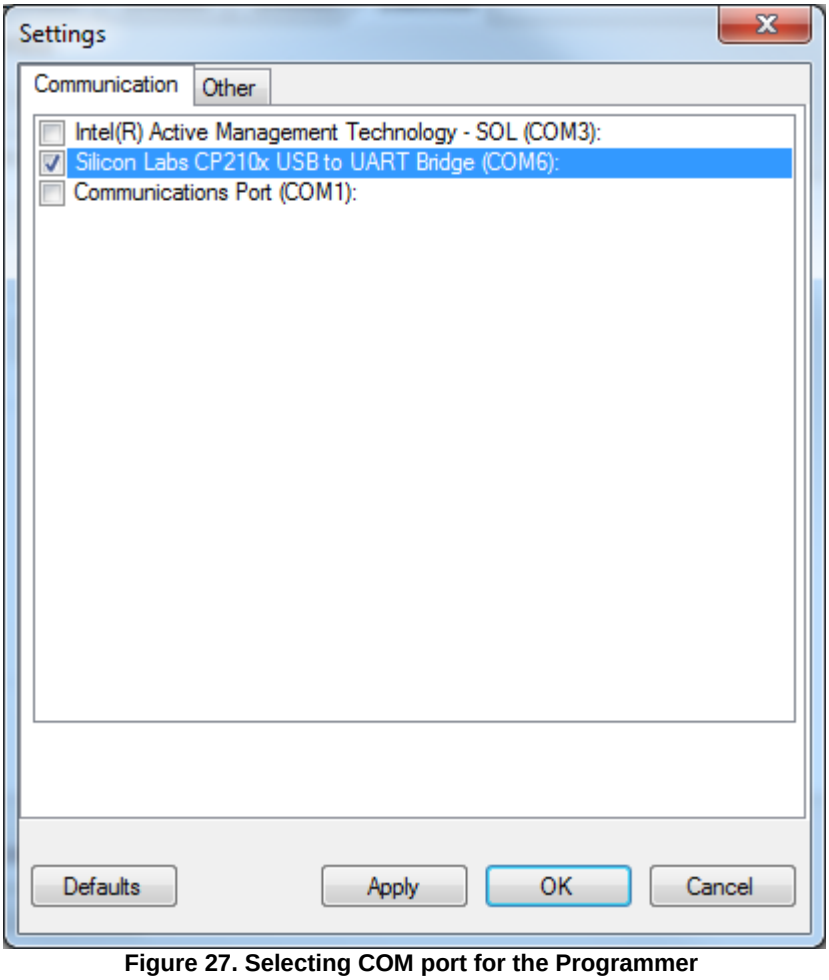


Figure 27. Selecting COM port for the Programmer

The selected COM port is used as the default port every time the Z-Wave PC Programmer is started.

5.2.2 Detecting target

The user may use the built-in detection tool to make sure the right chip type is selected.

Choose **Tools->Detect Target** from the PC Programmer **Tools** menu.

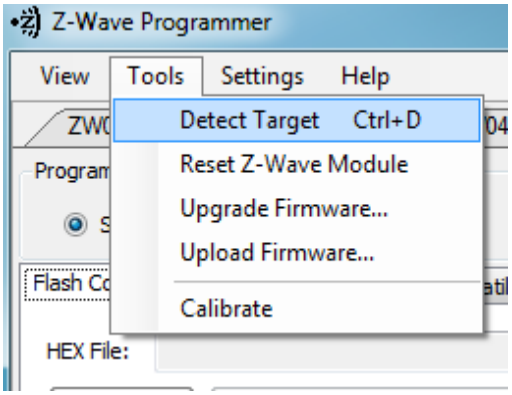


Figure 28. Detect target

The PC Programmer application may be configured to perform this detection during each application startup. Go to Settings->Other to enable automatic target detection on application startup.

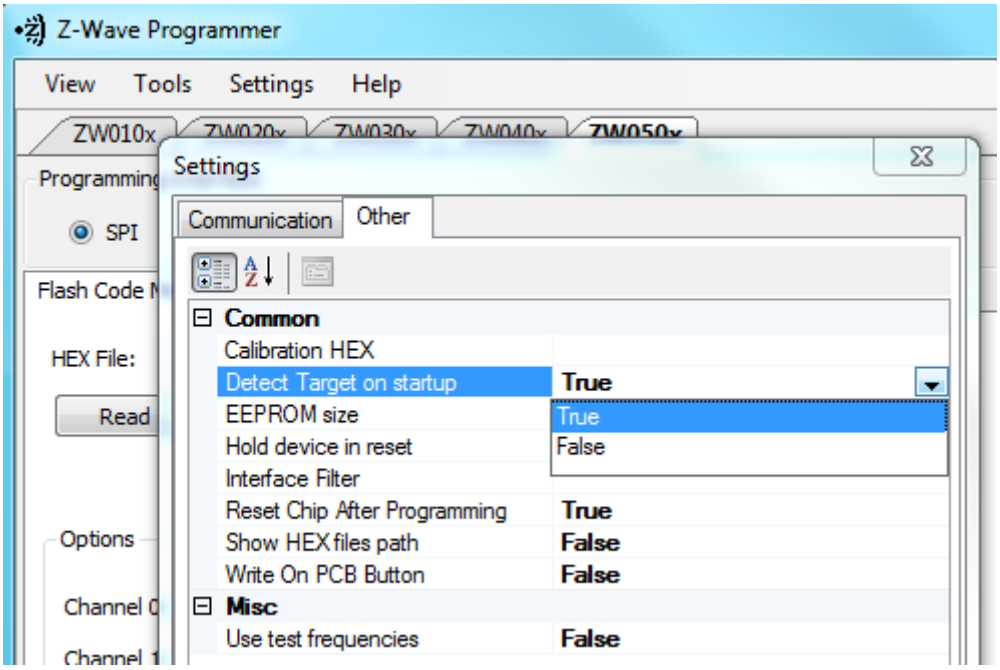


Figure 29. Enabling automatic target detection on startup

5.3 Programming Modes (500 series only)

The PC Programmer supports programming of 500 series chips via three different interfaces. The current interface is displayed in the upper part of the ZW050x view:

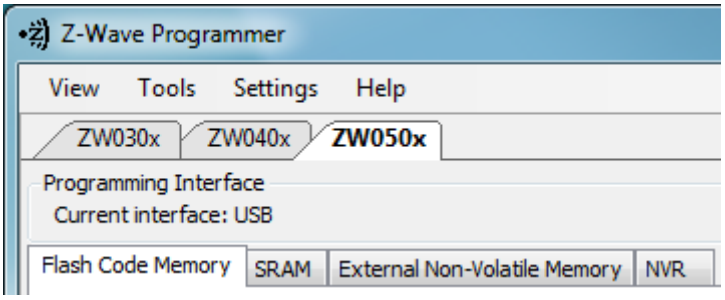


Figure 30. Current 500 series programming interface

The SPI programming interface should be used if at all possible. The other interfaces are intended for programming of end products with no SPI programming connector and for reduced function modules such as the ZM5304.
The ZDP03A does however provide UART and USB programming interface mapping for the module connector and may thus be used for performing product programming experiments via UART and USB using a Z-Wave development module, e.g. before the real end product hardware is available.

Table 5. Available programming interfaces of 500 series modules and chips

Chip/ Module	Description	SPI1	UART0	USB
ZDB5101	Development module with ZM5101 for ZDP03A	✓	✓	✓
ZDB5202	Development module with ZM5202 for ZDP03A	✓		
ZDB5304	Development module with ZM5304 for ZDP03A		✓	✓
ZM5101	General Purpose Z-Wave SiP Module (Full IO)	✓	✓	✓[3]
ZM5202	General Purpose Z-Wave Module (Basic IO)	✓		
ZM5304	Z-Wave Serial Interface Module with Antenna (Modem)		✓	✓[3]
SD3502	General Purpose Wireless Z-Wave Chip, QFN48 (Full IO)	✓	✓	✓[3]
SD3503	Wireless Z-Wave Modem SOC, QFN32 (Modem)		✓	✓[3]

Manipulation of external NVM from the PC Programmer requires access via the SPI interface of the chip.

[3] A product designed for USB programming MUST also provide access to the following interfaces: SPI1/UART0, RESET_N, POWER, GND. These interfaces allow for the initial transfer of a firmware image via UART to enable subsequent USB programming.

5.3.1 Programming via the SPI interface

The ZDP03A SPI interface is also known as “the Z-Wave module connector”.

To program a Z-Wave module via the ZDP03A SPI interface, connect the USB cable to the USB interface “J1” on the ZDP03A and select the ***Silicon Labs CP210x USB to UART Bridge*** port in the **Settings** menu of the PC Programmer application.

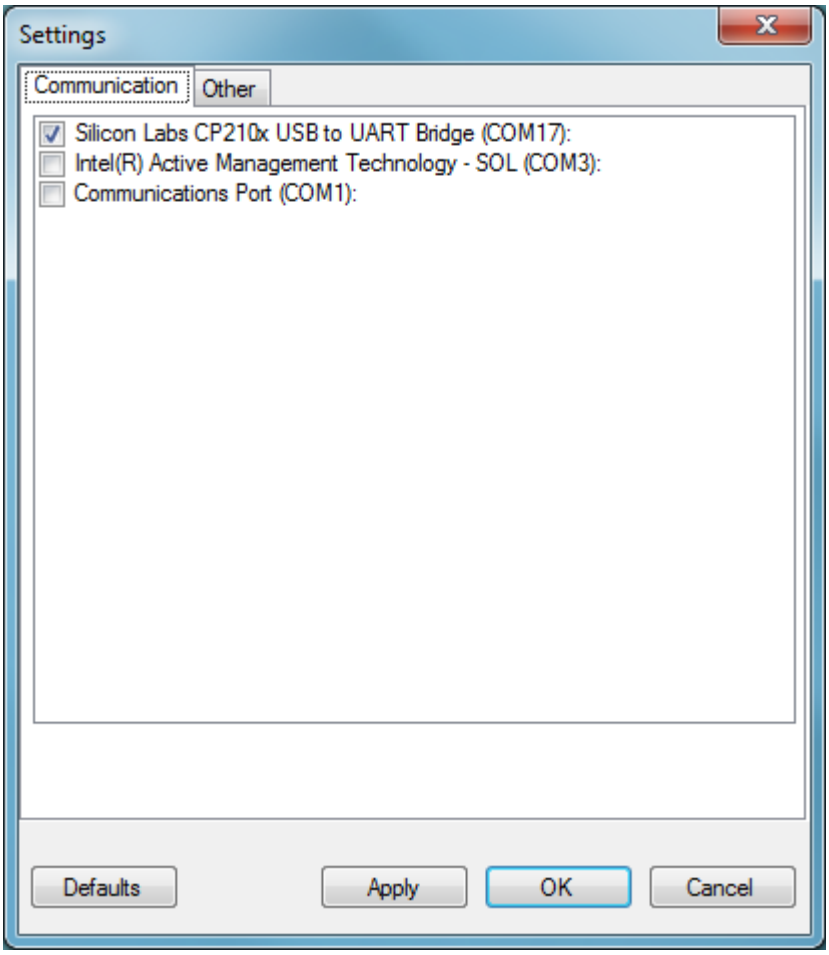


Figure 31. Selecting a COM port for SPI programming

5.3.2 Programming via the UART interface

The UART programming option is intended for in-system programming of modules with reduced I/O set such as the ZM5304. It may also be used for field upgrading of end products.

NOTICE: The SPI programming interface should be used if at all possible.

To program a Z-Wave chip via the UART interface, the chip must either be put into Auto Programming Mode by an application supporting that (e.g. Serial API) or by applying the reset signal to the Z-Wave chip. If experimenting with UART programming with a Z-Wave module mounted in a ZDP03A, the chip may be reset by activating the reset switch on the ZDP03A during the entire UART programming cycle.

The PC Programmer must be configured to use the PC COM port which connects to the ZDP03A COM port “J12”. The PC Programmer is configured via the **Settings** menu. Refer to Figure 32.

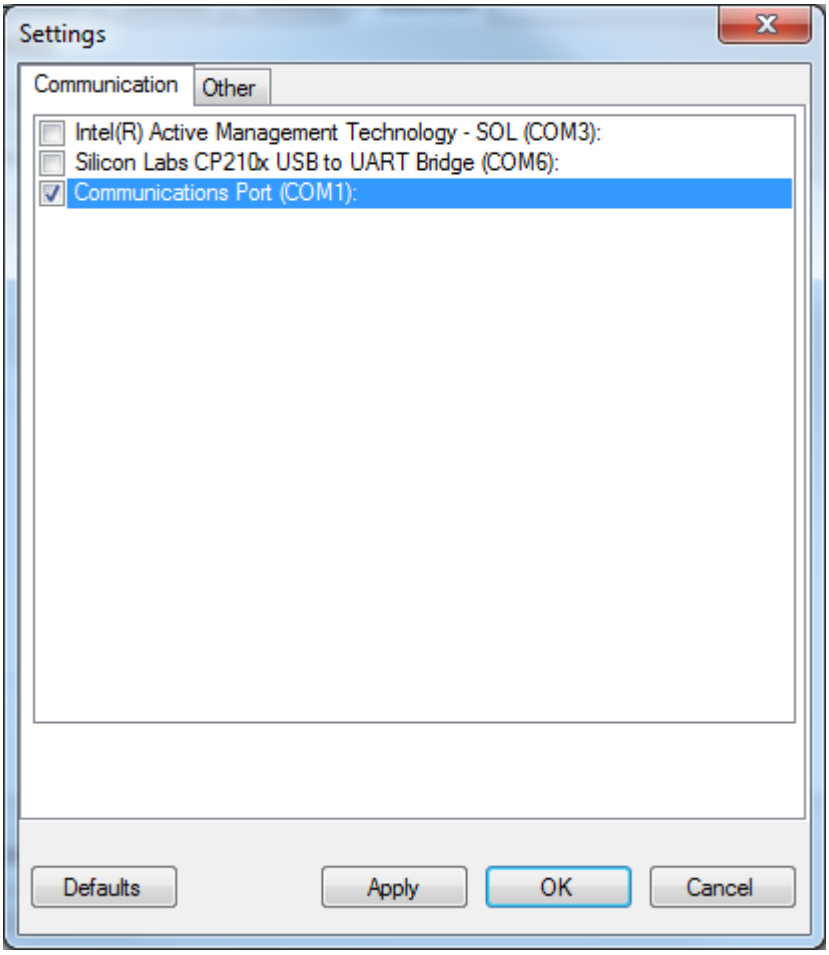


Figure 32. Selecting a COM port for UART programming

The COM port may have another port number than the one displayed in Figure 32.

5.3.3 Programming via the USB interface

The USB programming option is intended for in-system programming of modules with reduced I/O set such as the ZM5304. It may also be used for field upgrade of end products having a USB interface.

NOTICE: The SPI programming interface should be used if at all possible.

Before a Z-Wave module can be programmed via USB it must pre-programmed with a HEX file which enables USB communication.

Any “USBVCP” labeled serial API based application of the SDK 6.5x “\Product\Bin” folder may be used as initial image to enable USB programming during production.
Depending on the actual Z-Wave module, pre-programming may be performed via the SPI or UART interfaces.
For instance, the ZM5304 module must be pre-programmed via UART. This also goes for a dedicated USB device like the UZB Serial API stick provided with the Sigma Designs Z-Wave SDK. Since the end product is enclosed in plastic, pre-programming should be carried out during production PCB testing. All future firmware images should also support USB Programming Mode. Failing to do so will brick the end product and force the customer to return the product for service (pre-programming via internal interface).

In the following example, the SPI interface is used for pre-programming.

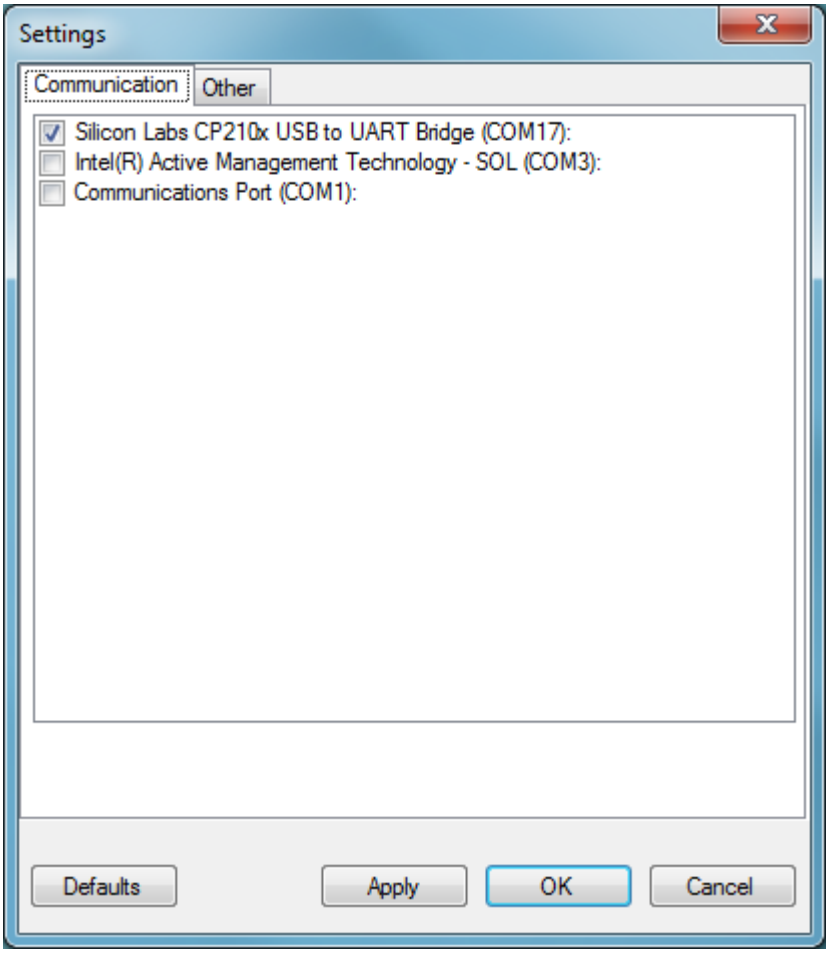


Figure 33. Selecting the SPI interface for pre-programming a USB device

The current interface is listed as “SPI”.

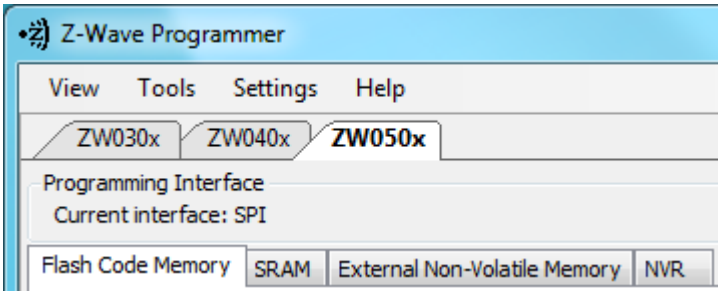


Figure 34. SPI interface selected before pre-programming of USB device

After pre-programming, the USB connector is moved from ZDP03A “J1” to “J18”.

First time a pre-programmed Z-Wave chip is connected to the PC via USB (ZDP03A port “J18”), the standard Windows ”New Hardware Found” dialog appears. Select ”Have Disk” and point to the **zw05xxprg.sys** or **zw05xxprg_x64.sys** driver file in the ZW050x_USB_Programming_Driver folder.

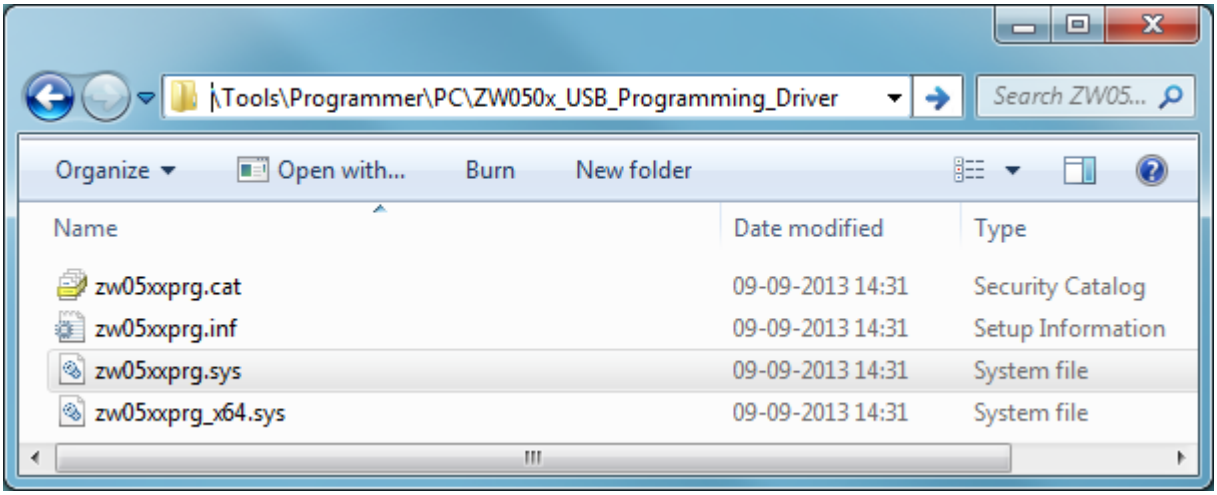


Figure 35. 500 series USB programming interface driver

After pre-programming, the Z-Wave device boots up as a USB Serial API device named “UZB”. Go to the **Settings** menu and select the newly appeared ***UZB*** port.

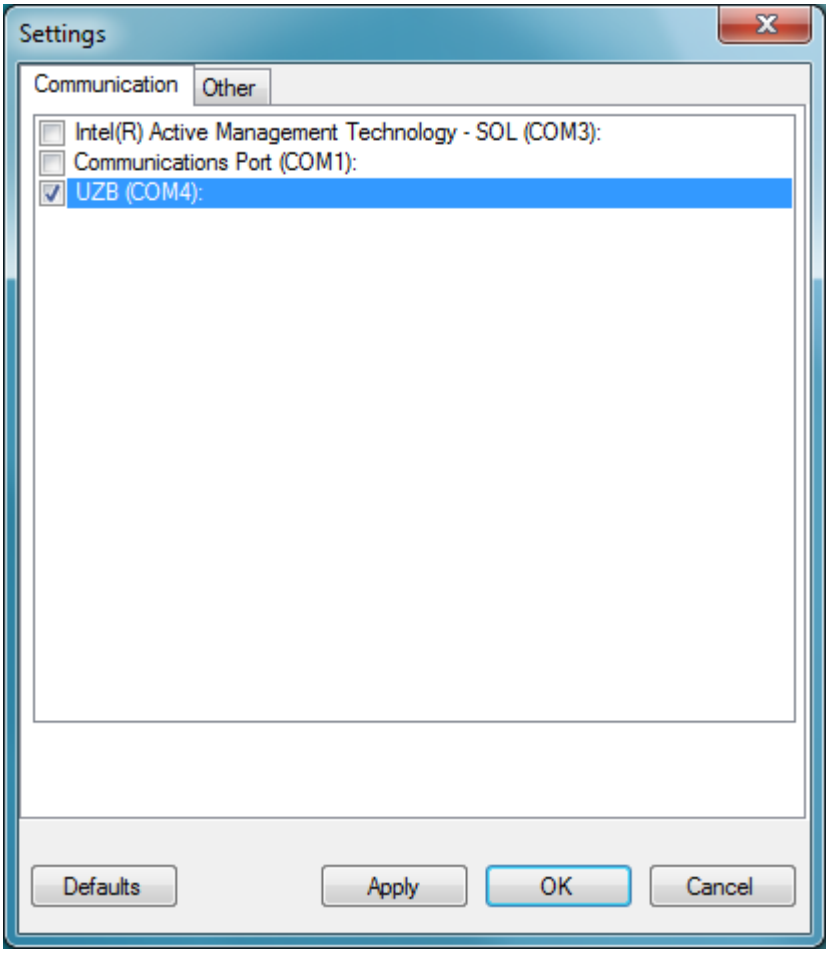


Figure 36. Selecting the new UZB interface of the pre-programmed USB device

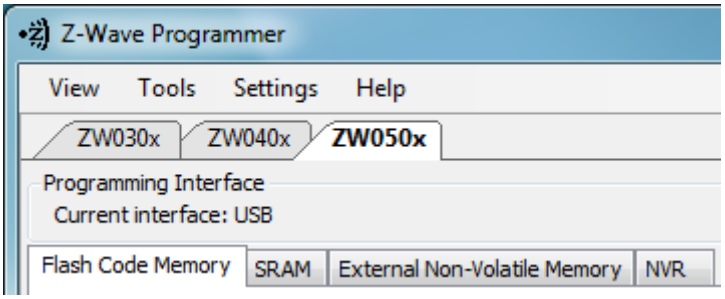


Figure 37. The current interface is USB

After the “UZB” interface is selected, the PC Programmer enables Auto Programming mode. This again makes the USB device change its role to “Programming Interface” device. A look in the **Settings** menu shows that the PC Programmer automatically selected the ***Sigma Designs ZWave programming interface*** port of the USB device.

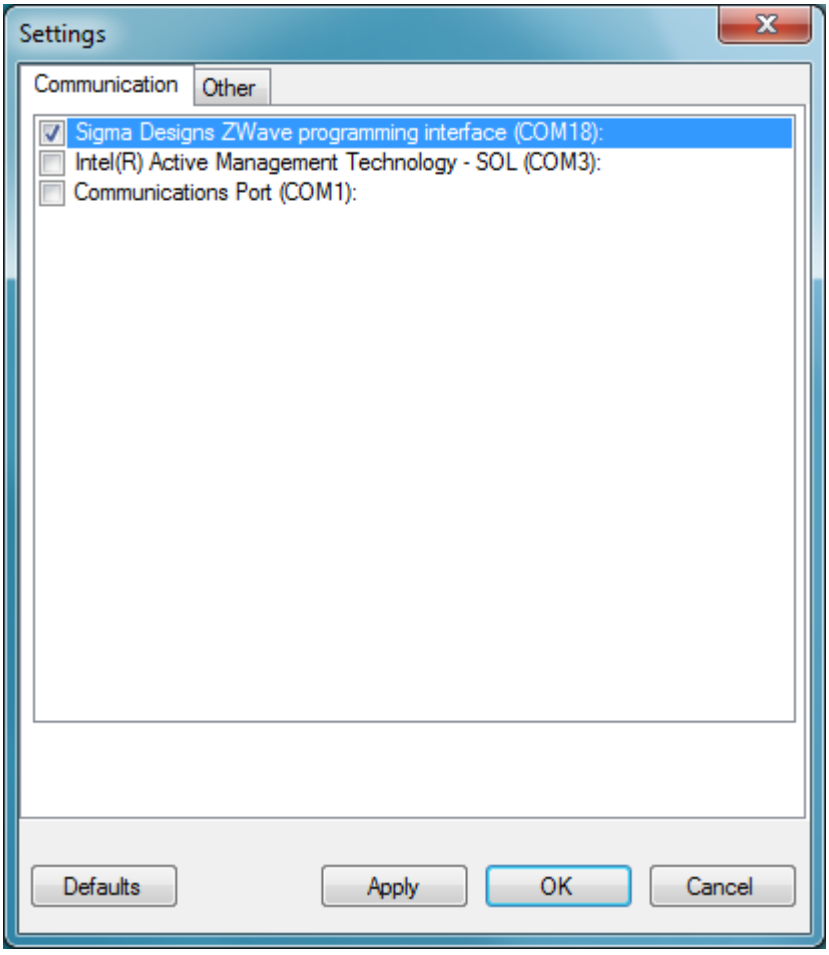


Figure 38. USB device “UZB” changes to “…Programming Interface” in programming mode

Now, the Z-Wave module is ready for programming via USB. Future USB programming does not need pre-programming via SPI or UART.

5.4 Internal NVM Memory

Depending on the chip generation, Z-Wave chips offer OTP or Flash memory technologies. From a PC Programmer GUI perspective, the process is the same when manipulating the internal NVM.

5.4.1 Writing a HEX file to the internal NVM Memory

A HEX file must be specified before the Z-Wave module can be programmed. Click on the open file icon as shown in Figure 39. Browse to the desired HEX file and select it. Select the tab that matches the actual chip type. Refer to [1] regarding the available sample applications and the associated HEX files.

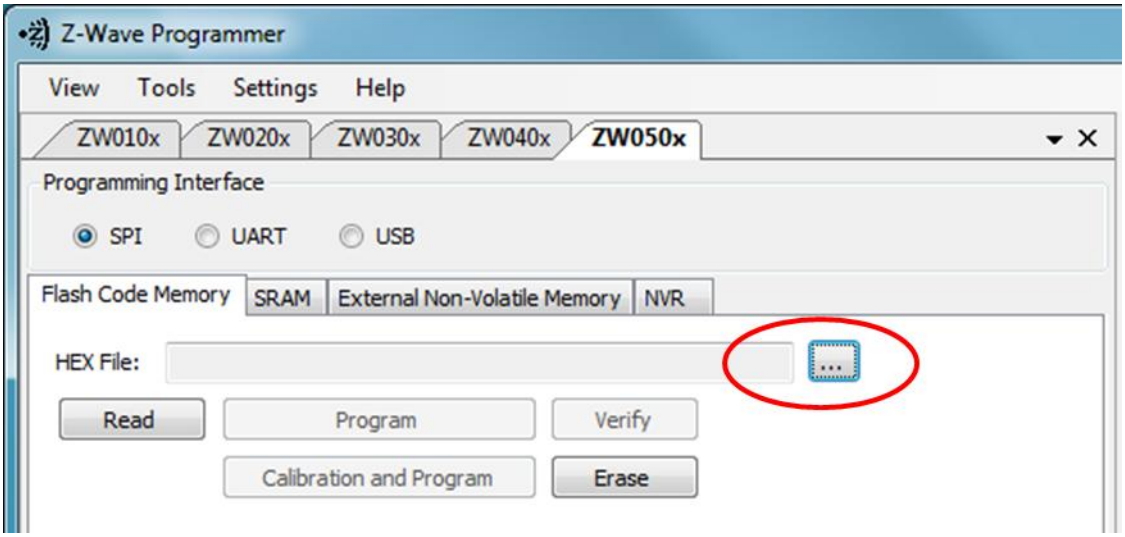


Figure 39. Button for locating a HEX file for internal NVM

The enhanced slave and controller libraries require External NVM. It may be necessary to initialize the External NVM on the Z-Wave module after programming the internal NVM. Failing to do so may cause the protocol or application to read corrupted data and possibly fail. Cases where it is necessary to initialize or update the external NVM include the update to a new application version with a different external NVM layout – or the programming of a completely different library and application to the Z-Wave module.

To Write and verify the content of the HEX file to the internal NVM memory, click on the 'Program' button. Depending on the actual chip generation, the 'Write' button may be used to write the HEX file while skipping the verification step.

In case of rewritable NVM technologies, the process starts by erasing the internal NVM. After the internal NVM is erased, the PC Programmer writes the HEX file as shown in Figure 40.

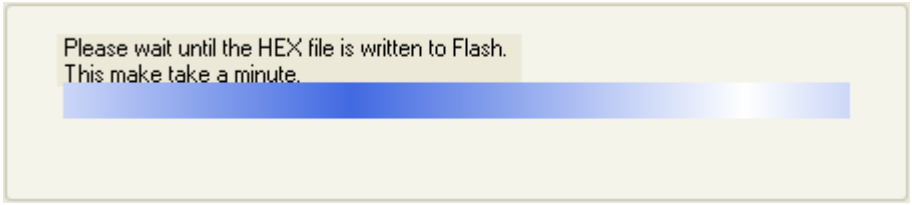


Figure 40. NVM writing process

If the writing process is completed successfully, a "**Program Done**" is shown in the status bar in the bottom of the PC Programmer window. A message box is displayed if any error occurs during programming.

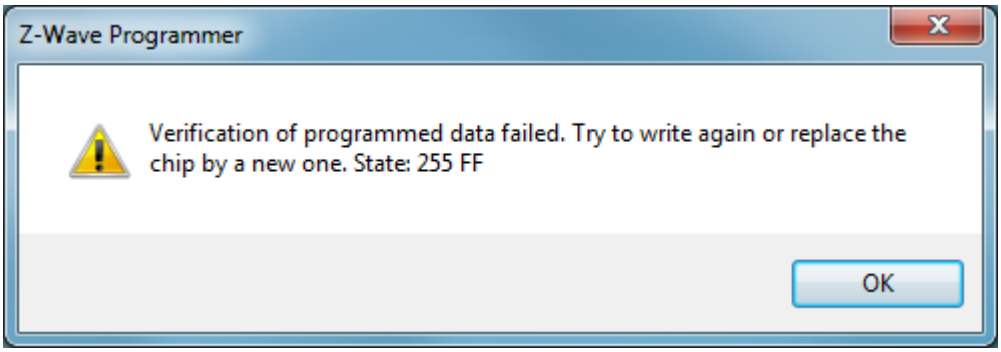


Figure 41. PC Programmer error message

5.4.2 Erasing the NVM

To erase the entire internal NVM memory, simply click on the '**Erase'** button.

The Erase button SHOULD NOT be used prior to the programming of a new software image to a ZW050x Z-Wave module. The erase operation not only clears the Flash code memory block but also the NVR page. The NVR page holds certain module specific data such as crystal calibration constants. If the NVR is cleared, the Z-Wave PC Programmer needs dedicated calibration hardware to perform a crystal calibration. Furthermore, it is also necessary to restore the remaining NVR content. Refer to section 5.4.5.

5.4.3 Reading the internal NVM memory content

To read back the content of the internal NVM of a Z-Wave module do the following:

Open the Output window if not already open. Click on the **Read** button. A progress indicator appears:

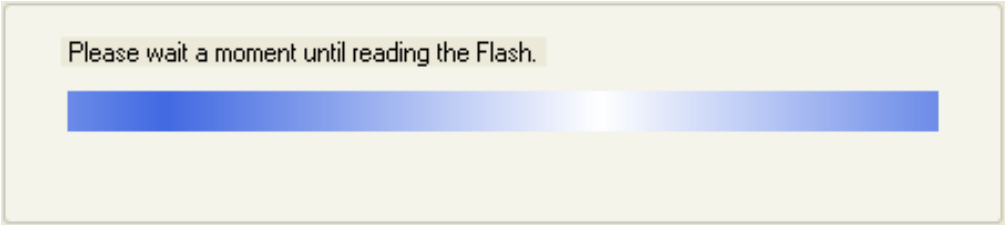


Figure 42. Internal NVM reading

When the internal NVM content has been read, the output is shown in the **Output** view.

To save the data, click on the **Save** icon.

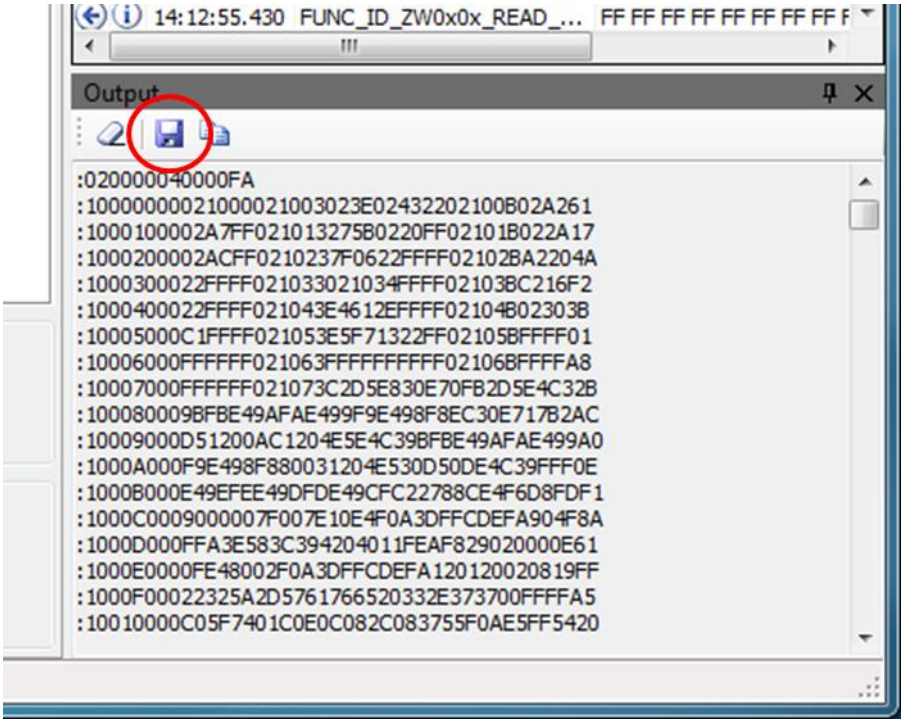


Figure 43. Saving a HEX file via the Output view

Select the destination folder and specify a file name for the HEX file. Then press **Save**.

5.4.4 Comparing internal NVM content with a HEX file

The content of the internal NVM of a Z-Wave module may be compared to a hex file.

Click on the open HEX file button and browse to the location of the HEX file in question.

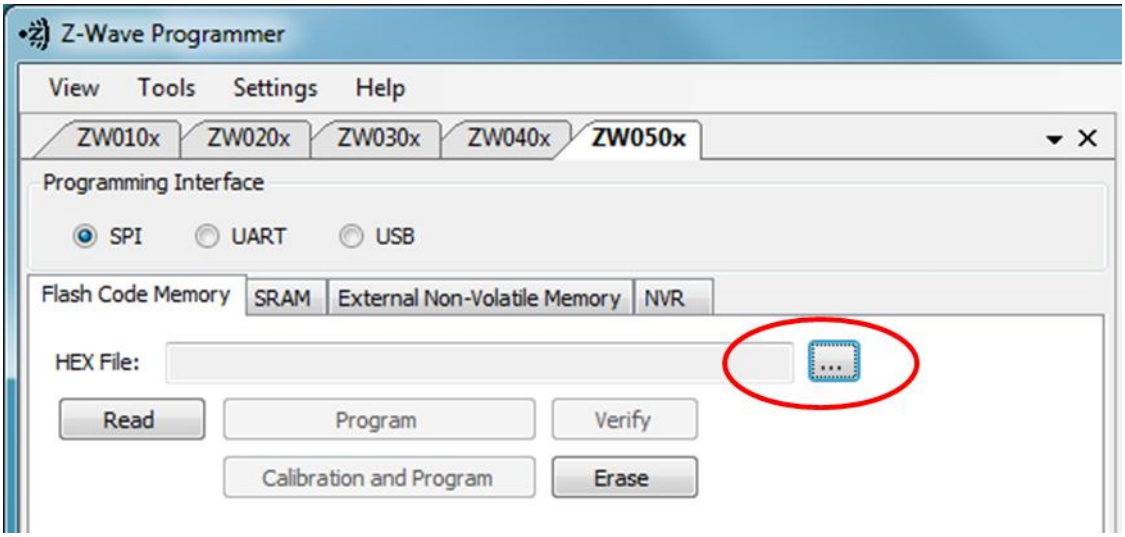


Figure 44. Button for locating a HEX file for internal NVM

Click on the **Compare/Verify** button. A popup message shows the progress.

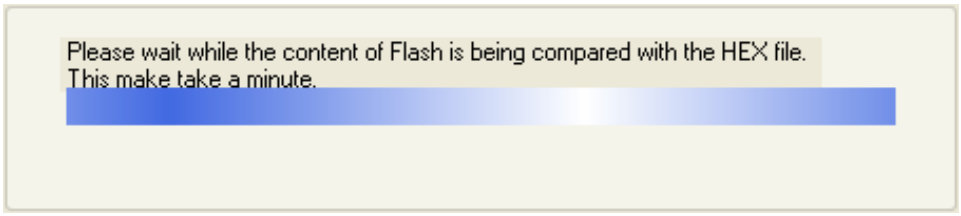


Figure 45. Progress of a HEX file comparison

When completed, the result of the comparison is reported.

5.4.5 Calibration and Program (ZW040x and ZW050x)

This option MUST be used during production of un-calibrated modules/chips. When “Calibration and Program” is selected, the PC Programmer performs crystal calibration and Tx calibration and then writes the specified HEX file to the internal NVM memory.

ZW040x and ZW050x Z-Wave modules which incorporate a clock crystal are already crystal calibrated from the factory and do normally not need to be calibrated again. Crystal calibration data is stored in ZW050x NVR storage and in ZW040x code memory. If the NVR storage is cleared, crystal calibration must be performed again. Crystal calibration requires dedicated calibration hardware connected to the ZDP03A programmer. The Sigma Designs RBK-ZWAVECALIBOX implements the required functionality for crystal calibration.

For the RBK-ZWAVECALIBOX to work together with the ZDP03A, a calibration HEX file must be specified in the Z-Wave PC Programmer. For further details, refer to section 5.8.1.

5.4.6 RF parameters

RF parameters are stored in the internal NVM memory. Depending on the actual technology, the PC Programmer can read back or manipulate the RF parameters of the Z-Wave chip.

The following sections describe how to access and adjust the parameters depending on the actual technology.

5.4.6.1 ZW0201/ZW0301 RF parameters

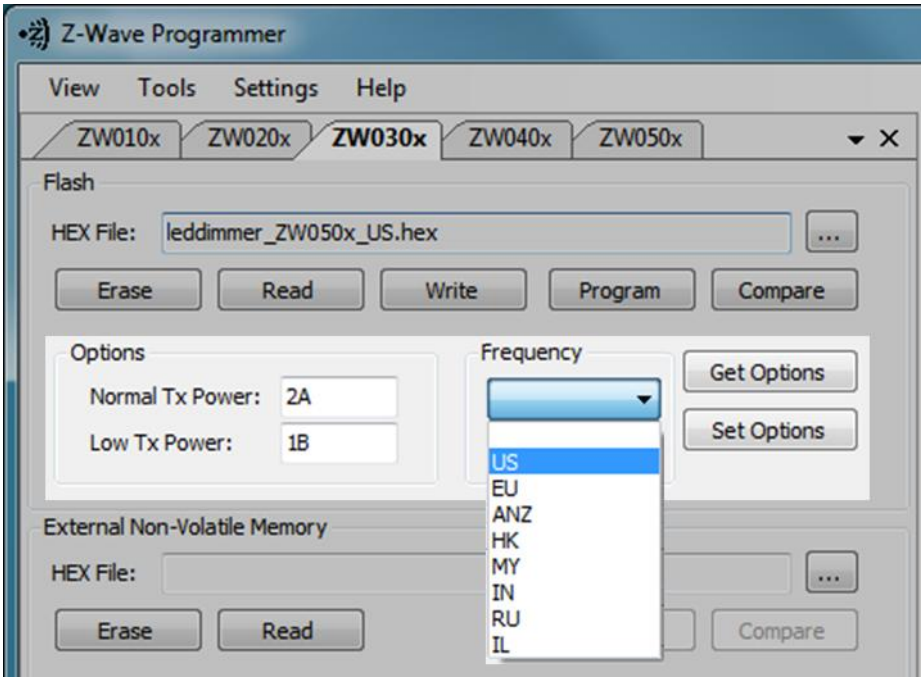


Figure 46. ZW0201/ZW0301 RF Parameters

To read back the RF parameters from the Z-Wave chip, click on the '**Get Options**' button.
To adjust the RF parameters, change the desired value then click on the '**Set Options'** button.

The "**Normal Tx Power**" entry controls the transmission power during normal operation.

The "**Low Tx Power**" entry controls the transmission power during inclusion of a new network node and during network repair to ensure a sufficient signal quality margin of discovered inter-node links.

The values entered into the text fields are control values which control the Power Amplifier (PA) gain. The resulting transmission power depends on the Z-Wave Module antenna conditions.

Table 6 shows the **only** valid control values. The table outlines the radiated power from the PA to the antenna, assuming no filters or other attenuation between the antenna and the PA.

Table 6. ZW0201/ZW0301 Tx Power Settings

Radiated Power [dBm]	PA Control value
-3	0x28
-1	0x29
0.5	0x2A
-19	0x14
-17	0x15
-15	0x16
-13	0x17
-11	0x18
-9	0x19
-7	0x1A
-5	0x1B

The control registers "**Normal Tx Power**" and "**Low Tx Power**" MUST be entered in hexadecimal form but without "0x", e.g. A0.
A precise mapping between radiated RF power from the antenna and the PA control value requires measurements with the specific product.

5.4.6.2 ZW040x RF parameters

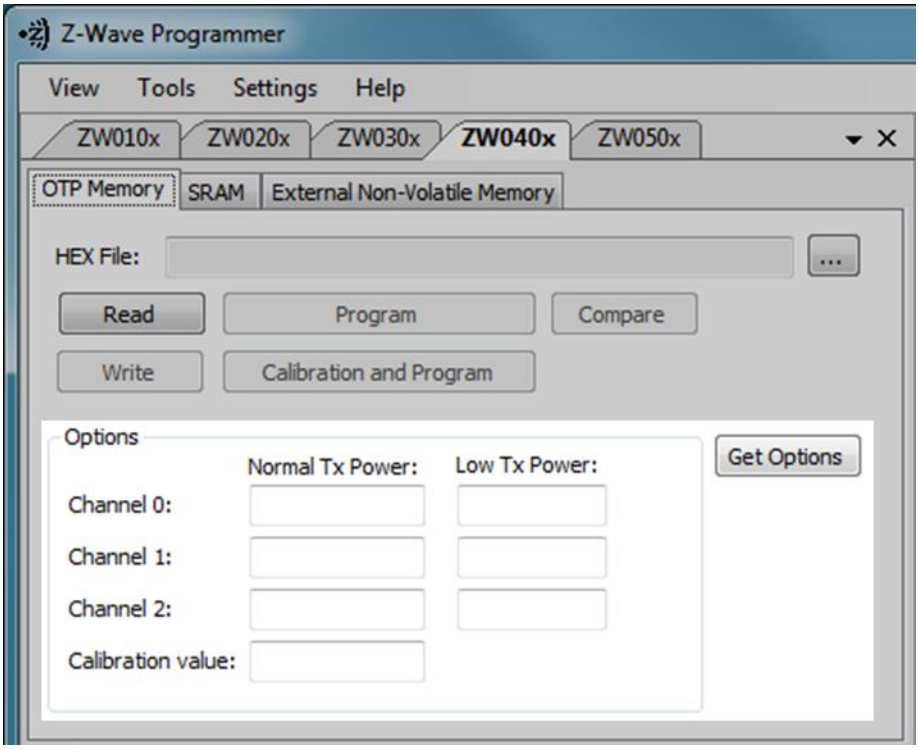


Figure 47. ZW0401 RF Parameters

To read back the RF parameters from the Z-Wave chip, click on the '**Get Options**' button. Being an OTP technology, the ZW040x chip family does not allow the PC Programmer to adjust RF parameters.

Table 7 shows the **only** valid control values. The table outlines the radiated power from the PA to the antenna, assuming no filters or other attenuation between the antenna and the PA:

Table 7. ZW040x Tx Power Settings

Radiated Power	PA Control value
Normal	0x20
Normal -2dB	0x18
Normal -4dB	0x10
Normal -6dB	0x0C
Normal -8dB	0x0A
Normal -10dB	0x08
Normal -12dB	0x06
Normal -14dB	0x05
Normal -16dB	0x04
Normal -18dB	0x03
Low: Normal -20dB	0x02
Normal -22dB	0x01

The control registers "**Normal Tx Power**" and "**Low Tx Power**" MUST be entered in hexadecimal form but without "0x", e.g. 20.
A precise mapping between radiated RF power from the antenna and the PA control value requires measurements with the specific product.

For more information on typical power values, see [5].

5.4.6.3 ZW050x RF parameters

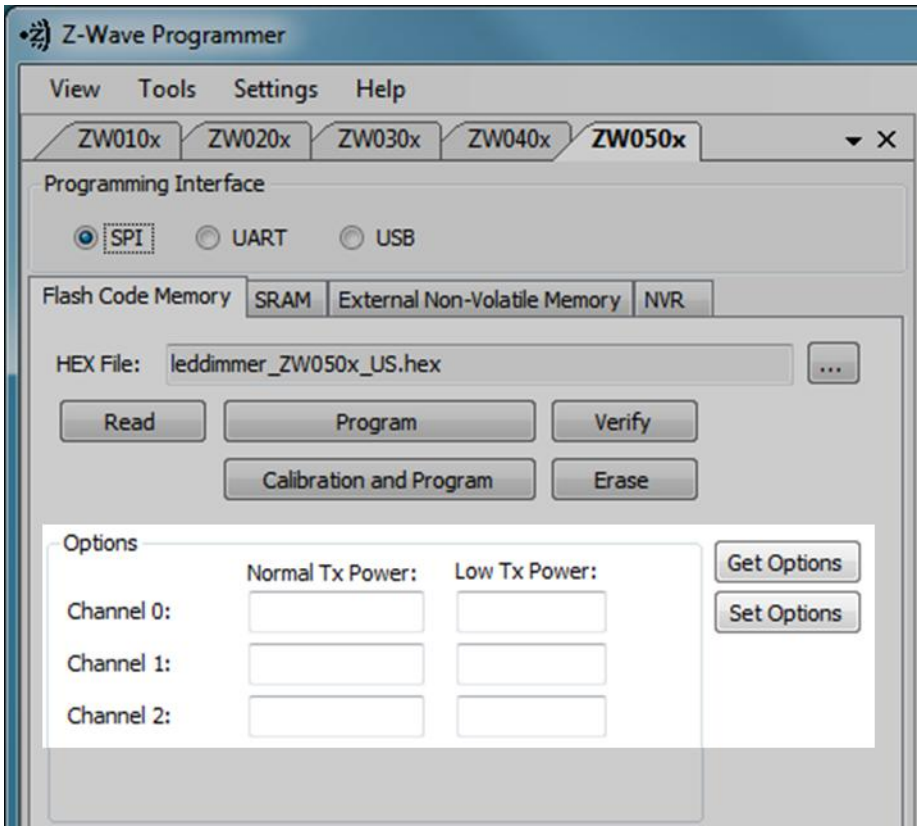


Figure 48. ZW0501 RF Parameters

To read back the RF parameters from the Z-Wave chip, click on the '**Get Options**' button.
To adjust the RF parameters, change the desired value then click on the '**Set Options'** button.

The "**Normal Tx Power**" entry controls the transmission power during normal operation.

The "**Low Tx Power**" entry controls the transmission power during inclusion of a new network node and during network repair to ensure a sufficient signal quality margin of discovered inter-node links.

The values entered into the text fields are control values which control the Power Amplifier (PA) gain. The resulting transmission power depends on the Z-Wave Module antenna conditions.

The table below shows the **only** valid control values. The table outlines the radiated power from the PA to the antenna, assuming no filters or other attenuation between the antenna and the PA:

Table 8. ZW050x Tx Power Settings

Radiated Power [dBm]	PA Control value
Normal	0x3F
Normal -2dB	0x24
Normal -4dB	0x1E
Normal -6dB	0x16
Normal -8dB	0x11
Normal -10dB	0x0E
Normal -12dB	0x0B
Normal -14dB	0x09
Normal -16dB	0x07
Normal -18dB	0x05
Low: Normal -20dB	0x04
Normal -22dB	0x03

The control registers "**Normal Tx Power**" and "**Low Tx Power**" MUST be entered in hexadecimal form but without "0x", e.g. 3F.
A precise mapping between radiated RF power from the antenna and the PA control value requires measurements with the specific product.

NOTE: For ZM5304US, the Normal power is 0x1F. This only applies for the ZM5304US.

For more information on output power versus PA Control values, refer to the relevant 500 Series datasheets.

5.5 **SRAM** (ZW040x)

This section applies only to ZW040x Chips[4].

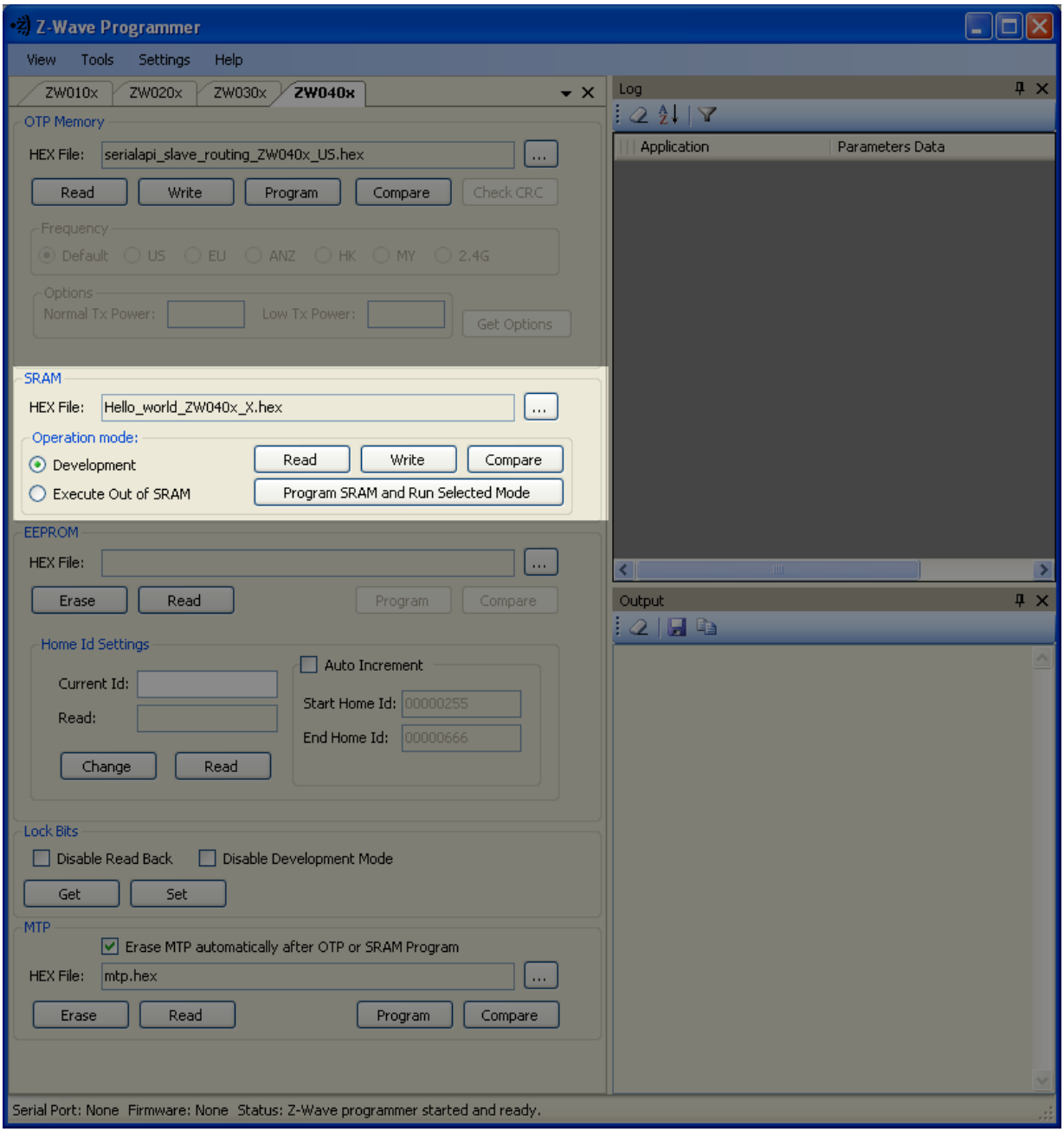


Figure 49. SRAM

The 400 series Z-Wave Chip has two SRAM blocks that can be accessed from the SPI interface.

[4] The Z-Wave PC Programmer v.2.65 and potentially newer versions include an SRAM tab for ZW050x modules. This tab SHOULD NOT be used.

5.5.1 SRAM operation modes

SRAM can be operated in two modes:

- Development;
- Execute out of SRAM

A 4 kbyte SRAM block may be used as code memory in the “Execute out of SRAM” mode. A 12 kbyte SRAM is used as code memory in development mode.

In Development mode, the 12 kbyte SRAM overlays the upper 12 kbytes of the OTP memory. This enables a way of running application software in RAM during development, thus not having to program the OTP.

5.5.2 SRAM operation functions

5.5.2.1 Read

The content of SRAM can be read by activating the **Read** button. The content of the SRAM is displayed in the **Output** view.

5.5.2.2 Write

Activate the **Write** button to write a HEX file to SRAM.

5.5.2.3 Compare

Activate the **Compare** button to compare a HEX file to the SRAM content will be compared with the selected file, a procedure alike Flash/OTP comparison.

5.5.2.4 Program SRAM and Run Selected Mode

Select a HEX file to be programmed to SRAM in the file location bar, and activate the **Program SRAM and Run Selected Mode** button. The file will be written to the SRAM with verification, and the selected Operation Mode will be run.

5.6 External NVM

The Z-Wave PC Programmer can manipulate the external NVM of Z-Wave modules. However, Z-Wave PC Programmer supports only EEPROM's having sizes equal to 16KB, 64KB and 128KB as external NVM. Finally the external NVM can only be accessed via SPI.

To select the wanted EEPROM size go to '**Settings'** in the menu bar and select the '**Other**' TAB.

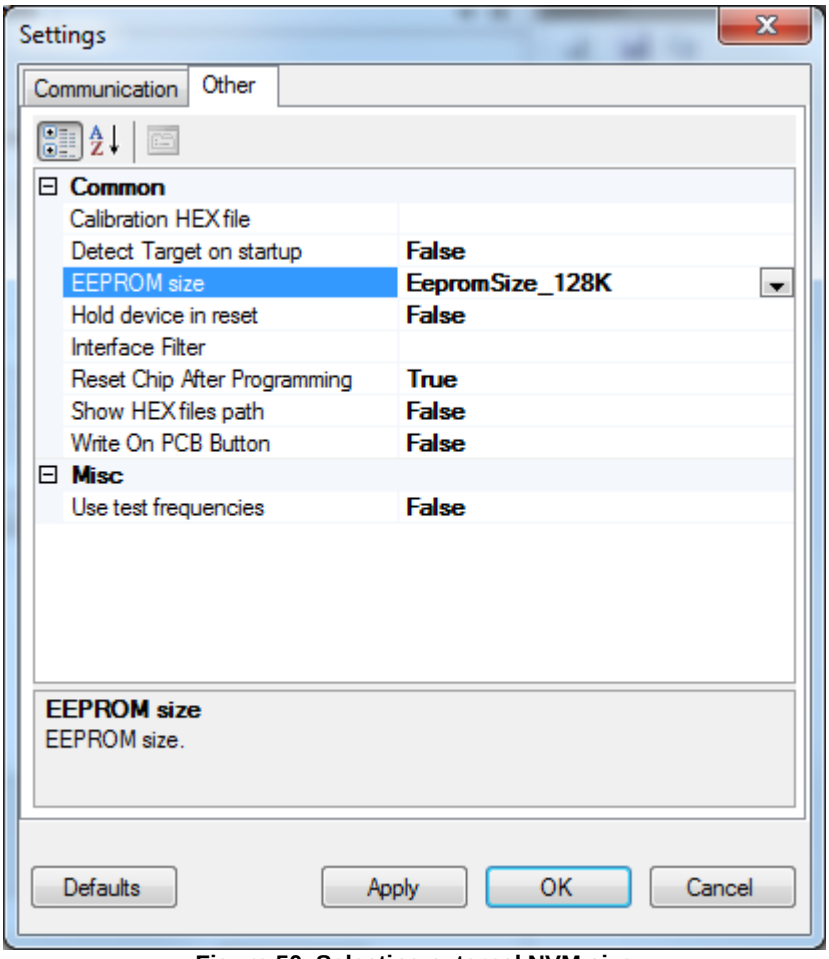


Figure 50. Selecting external NVM size

The external NVM can be accessed via the '**External Non-Volatile Memory**' TAB for a given Z-Wave chip.

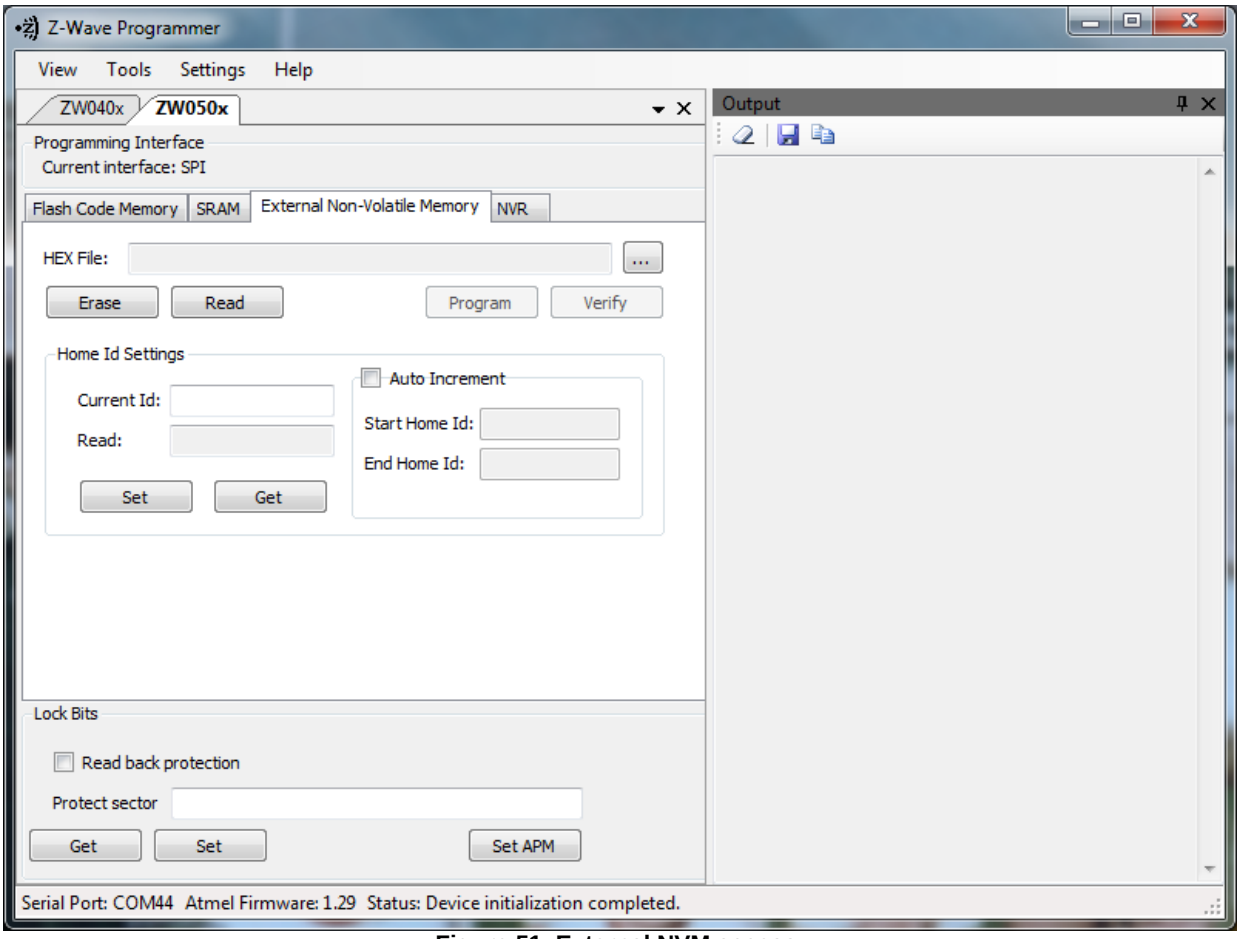


Figure 51. External NVM access

5.6.1 Writing an External NVM file to the External NVM

Select a HEX file to be written to External NVM and activate the '**Program**' button. The file will be written to the External NVM. This process will take a few moments to complete.

If any error occurs during programming, a failure message will appear in a pop-up window.

5.6.2 Clear the External NVM content

To clear the entire External NVM including the Home ID, click on the '**Erase'** button. The process will take a few moments to complete.

5.6.3 Reading the External NVM content

The Z-Wave PC Programmer can read the content of External NVM mounted on a Z-Wave module and save it to a file. To read the content of the External NVM, do the following:

Click on the '**Read**' button, and the content of the External NVM will be read. This process will take a few moments. When finished, the content is displayed in the **Output** view.

To save the external NVM content to a HEX file, click on the '**Save**' icon in the **Output** view.

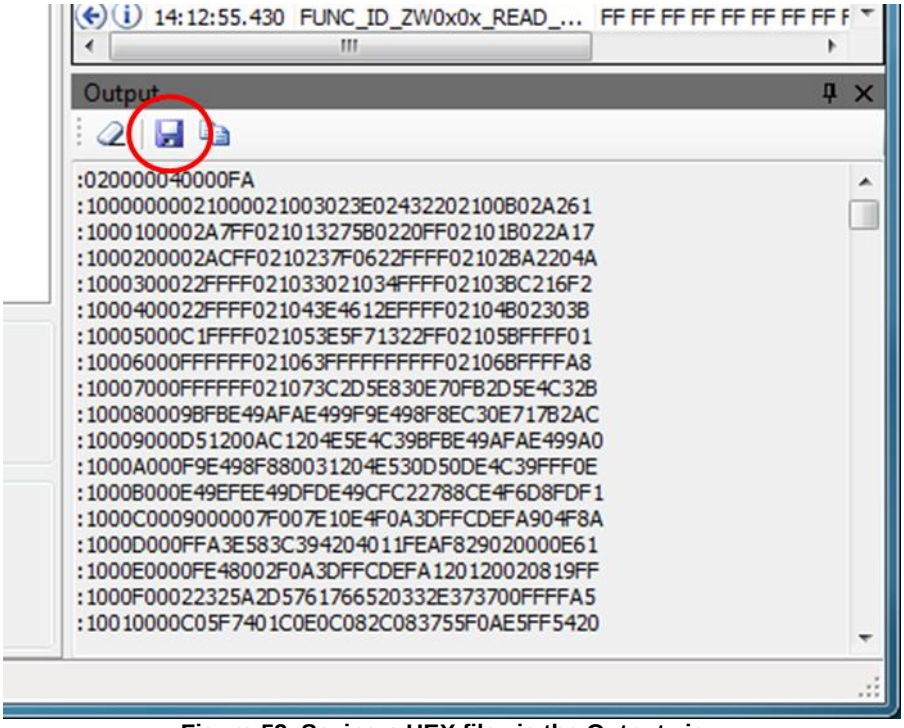


Figure 52. Saving a HEX file via the Output view

5.6.4 Comparing the External NVM content

Z-Wave PC Programmer can be used to compare the content of an External NVM to a hex file as follow:

Click on the open HEX file button and browse to the HEX file in question. After the HEX file was selected, click on the 'Compare' button. A status pop-up window shows the progress.

5.6.5 Home ID manipulation

The Home ID stored in the External NVM can be read by the Z-Wave Programmer. Simply click on the '**Read**' button. The Home ID is shown in the '**Read**' textbox.

The Z-Wave PC Programmer can be used to change or read the Home ID of a Z-Wave module. To change the Home ID, write the desired Home ID in the '**Current ID**' text box then click on the '**Change**' button. The new Home ID will be written to the External NVM. The Home ID is read back from the External NVM and displayed in the '**Read'** textbox.

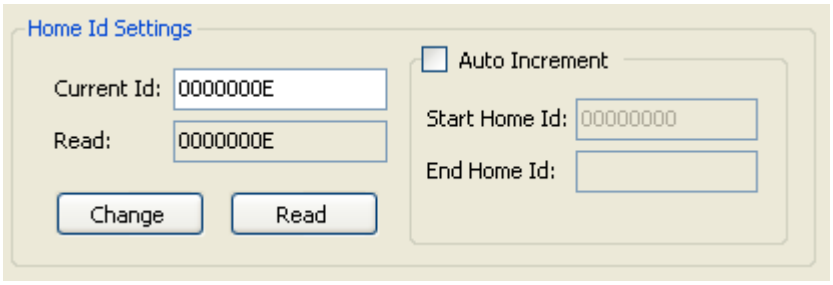


Figure 53. The Home ID GUI elements

The Z-Wave PC Programmer can also be used to setup a range of Home IDs for automated Home ID management. Each time the '**Change'** button is clicked, the next Home ID from the range is used. To use this feature, check the '**Auto increment**' checkbox and set a start/end Home ID in the '**Start Home ID**' and '**End Home ID**' text boxes respectively.

Click on the '**Change**' button, and the next available Home ID in the range will be shown in the '**Current ID**' textbox and written to the External NVM as shown in Figure 54.

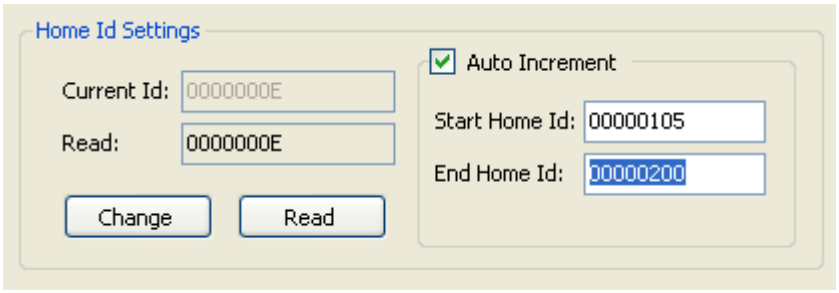


Figure 54. Using the PC Programmer Home ID Auto increment feature

5.7 Tools

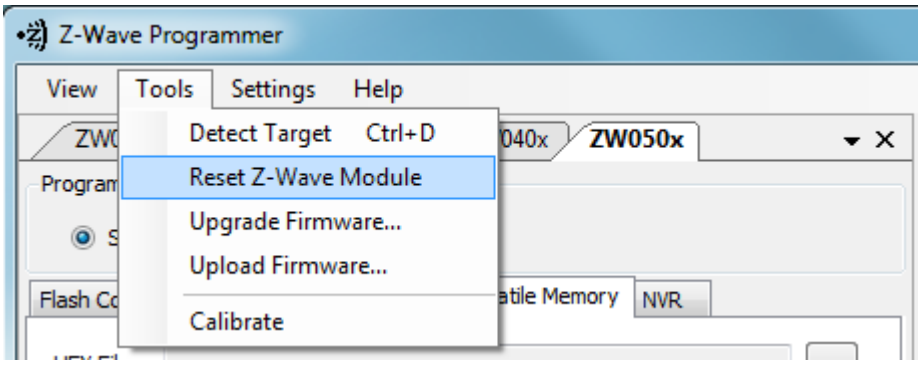


Figure 55. Resetting a Z-Wave module from Z-Wave PC Programmer

This function sends a reset signal to the module. Note that this feature only is operational when the Z-Wave module is connected to the ZDP03A via the ISP interface (the ZDP03A module connector).

5.8 Miscellaneous settings

A number of configuration parameters may be maintained via the Settings->Other menu item.

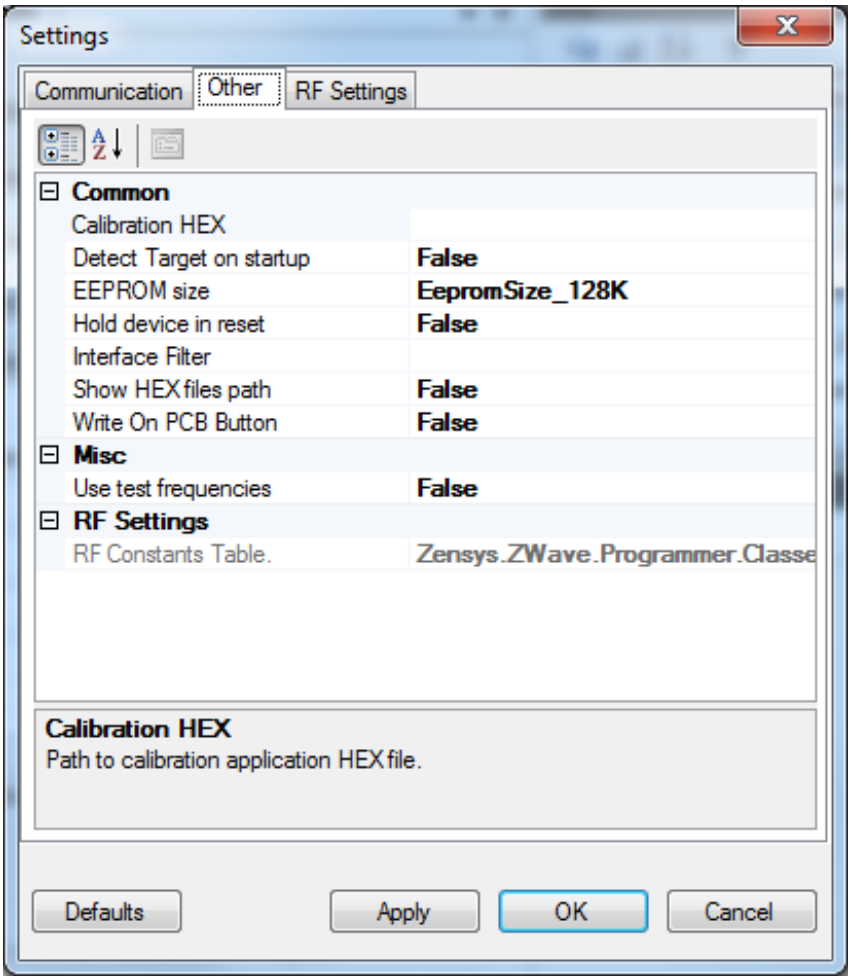


Figure 56. Other settings

5.8.1 Calibration HEX

Main menu > Settings > Other > Calibration Hex

This option defines which hex file the Z-Wave PC Programmer must use for performing the crystal calibration.

The Sigma Designs RBK-ZWAVECALIBOX implements the required functionality for crystal calibration. A dedicated calibration firmware image must be written to the chip to carry out the crystal calibration. The result is recorded by the Z-Wave PC Programmer application and subsequently programmed into the proper NVR entries.

The HEX file specified by this option contains the dedicated calibration firmware image. The HEX file must be selected to match the actual module type and the connections to the RBK-ZWAVECALIBOX. Refer to [7]. The Z-Wave PC Programmer automatically writes this HEX file to the Z-Wave chip during calibration. If “Calibrate and Program” is selected in the Z-Wave PC Programmer, the requested HEX file containing an application firmware image is written to the chip after calibration is completed.

5.8.2 Detect Target on startup

Main menu > Settings > Other

Switch ‘Detect Target on startup’ to **True.** The Programmer will detect the target on each application startup.

5.8.3 EEPROM size

Main menu > Settings > Other

Specify the size of external NVM memory.

5.8.4 Hold device in reset

Main menu > Settings > Other

If set to **True**, a reset signal will be sent to the Z-Wave module after programming until the ZDP03A is power cycled.

5.8.5 Interface Filter

This option serves to narrow the search of the required PC interface port for the ZDP03A Z-Wave Programming hardware. Any keyword, e.g. ‘UART’ can be typed in the text field.

5.8.6 Reset Chip after programming

Enable this option to make the chip restart after programming has been completed.

5.8.7 Show HEX file path

When this option is enabled, the Z-Wave PC Programmer will show the full path of HEX files in the PC Programmer main window.

When the option is not enabled, the Z-Wave PC Programmer will only show the HEX files names.

5.8.8 Write on PCB Button

When this option is enabled, programming of the flash memory may be initiated by pressing the physical S1 pushbutton on ZDP03A as an alternative to pressing the “Program” button in the PC Programmer.

5.8.9 Use test frequencies

When this option is enabled, the ZW020x/ZW030x frequency dropdown list of the presents a range of test frequencies to be used for internal testing.

5.9 Keyboard shortcuts

The GUI is extended with a number of shortcut keys to provide an easier method of navigating the Z-Wave PC Programmer. The table below lists the available shortcut keys.

Table 9. Z-Wave PC Programmer shortcut keys

Z-Wave PC Programmer features	Shortcut Keys
Detect target	Ctrl + D
Select ZW010x target	F1
Select ZW020x target	F2
Select ZW030x target	F3
Select ZW040x target	F4
Select ZW050x target	F5
Open a flash HEX file	Ctrl + O
Program flash (write then verify)	Ctrl + P
Write flash (Write only)	Ctrl + W
Erase flash	Ctrl + E
Read flash	Ctrl + R
Compare flash content with a HEX file	Ctrl + M
Write RF settings (except ZW020x/ZW030x)	SHIFT + W
Read RF settings (except ZW020x/ZW030x)	SHIFT + R
Open an External NVM HEX file	Alt + O
Program External NVM	Alt + P
Clear External NVM	Alt + E
Read External NVM	Alt + R
Compare External NVM content with a HEX file	Alt + M
Change home ID in External NVM	Alt + F1
Read home ID from External NVM	Alt + F2
Read SRAM (ZW040x only)	Ctrl + Alt + R
Write SRAM (ZW040x only)	Ctrl + Alt + W
Compare SRAM (ZW040x only)	Ctrl + Alt + C
Program SRAM and run selected mode (ZW040x only)	Ctrl + Alt + P
Exit	Alt+X

The shortcut keys can also be obtained directly from the GUI. Simply hold the mouse over a GUI button to see a tool tip for this particular button.

5.10 Z-Wave Programmer console interface

The Z-Wave PC Programmer provides a console interface. The console interface allows the Z-Wave PC Programmer to be used from a command prompt and to be integrated in automation scripts. Scripts may benefit from using the application status code returned by the Z-Wave PC Programmer. Refer to 5.10.3.

The following text may also be found in the help function of the console interface. The help text can be displayed by entering the command `ZWaveProgrammer /?`.

5.10.1 Console interface syntax

The Console interface has the following syntax:

ZWaveProgrammer.exe -c <comport> | -s <comport> | -u <comport> [-t <chipType>] [operands...]

where

-c <comport>	Enable SPI programming interface using Atmel on ZDP03A. <comport> must have the format "COMx" where x is number of the USB port that is connected to the Atmel on the ZPD03A (through the Silicon labs bridge).
-s <comport>	Enable UART programming interface. <comPort> must have the format "COMx" where x is number of the UART port that is connected to UART0 on the Z-Wave chip.
-u <comport>	Enable USB programming interface programming. <comPort> must have the format "COMx" where x is number of the USB port that is connected to the USB interface of the Z-Wave chip.
-t <comport>	Check the detected chip type against <chiptype>. <chiptype> can be either ZW010x, ZW020x, ZW030x, ZW040x or ZW050x
operands	See the following sections for relevant chip type. The Z-Wave Programmer GUI will open, if no operands are applied.

5.10.1.1 100, 200 and 300 Series Z-Wave SoC/Module specific operands

-p <filename>.hex	Erase Flash code space and lock bits, program Flash code space with the Intel hex file specified, and verify.
-f <filename>.hex	Erase Flash code space and lock bits, and program chip without verification.
-r <filename>.hex	Read the Flash code space contents and write it to the Intel hex file specified.
-v <filename>.hex	Verify Flash code space contents against the Intel hex file specified.
-e	Erase Flash code space and lock bits.
-pf <frequency>	Set the frequency where <frequency> is either EU, US, ANZ, HK, MY and IN depending on the SDK used. Must be used in conjunction with the -p or –f operands.
-ro	Read RF options bytes from Flash code space. Refer to the user guide for a description of these bytes.
-so <nc0> <lc0>	Set Flash code space RF options. Where: nc0 is the normal power level for RF channel 0 (hex byte) lc0 is the low power level for RF channel 0 (hex byte) See section 5.4.6 for a description of legal power values depending on SoC used.

5.10.1.2 400 Series Z-Wave SoC/Module specific operands

-p <filename>.hex	Run Xtal calibration, program OTP code space with the Intel hex file specified and run CRC check on code space contents.
-r <filename>.hex	Read the OTP code space contents and write it to the Intel hex file specified.
-v <filename>.hex	Verify OTP code space contents against the Intel hex file specified.
-so <nc0> <lc0> <nc1> <lc1> <nc2> <lc2>	Set OTP code space RF options. Not valid in EooS and Development modes. Where: nc0 is the normal power level for RF channel 0 (hex byte) lc0 is the low power level for RF channel 0 (hex byte) nc1 is the normal power level for RF channel 1 (hex byte) lc1 is the low power level for RF channel 1 (hex byte) nc2 is the normal power level for RF channel 2 (hex byte) lc2 is the low power level for RF channel 2 (hex byte) See section 5.4.6 for a description of legal power values depending on SoC used.
-ls <lockbitsbyte>	Set lockbits. Where <lockbitsbyte> is the lock bits byte (hex byte). See also section 4.5.4 regarding lock bits description.
-lg	Get lock bits. Returns Lock Bits byte (hex byte) See also section 4.5.4 regarding lock bits description.
-sr <filename>.hex	Read the SRAM contents and write it to Intel hex file specified.
-sw <filename>.hex	Write the contents of Intel hex file specified to the XRAM.
-sc <filename>.hex	Compare the XRAM contents with Intel hex file specified.
-swrd <filename>.hex	Read the Intel hex file specified, write the contents to the XRAM and enable Development mode.
-sweo <filename>.hex	Read the Intel hex file specified, write the contents to the XRAM and enable EooS mode.
-sro	Read RF options from XRAM. Only valid in Development mode.
-sso <nc0> <lc0> <nc1> <lc1> <nc2> <lc2>	Set XRAM RF options. Only valid in Development mode. Where: nc0 is the normal power level for RF channel 0 (hex byte) lc0 is the low power level for RF channel 0 (hex byte) nc1 is the normal power level for RF channel 1 (hex byte) lc1 is the low power level for RF channel 1 (hex byte) nc2 is the normal power level for RF channel 2 (hex byte) lc2 is the low power level for RF channel 2 (hex byte) See section 5.4.6 for a description of legal power values depending on SoC

	used.
-mp <filename>.hex	Read the Intel hex file specified and write the contents to the MTP memory. Zero-fill empty areas.
-mp 0	Erase entire MTP memory.
-mr <filename>.hex	Read the Intel hex file specified and write the contents to the MTP memory.

Notice: In Normal working mode of the ZW040x chip, any operation with MTP memory will leave ZW040x chip in the reset state. You need to cycle the power of the chip to switch its working mode to Normal and start execution of the embedded application from OTP.

5.10.1.3 500 Series Z-Wave SoC/Module specific operands

-p <filename>.hex	Read NVR, erase Flash code space, lock bits and NVR, Run Xtal and TX calibration, re-write updated NVR contents to NVR, program Flash code space with the Intel hex file specified and run CRC check on code space contents.
-r <filename>.hex	Read the Flash code space contents and write it to the Intel hex file specified.
-v <filename>.hex	Verify chip against the Intel hex file specified.
-so <nc0> <lc0> <nc1> <lc1> <nc2> <lc2>	Set Flash code space RF options. Not valid in EooS mode. Where: nc0 is the normal power level for RF channel 0 (hex byte) lc0 is the low power level for RF channel 0 (hex byte) nc1 is the normal power level for RF channel 1 (hex byte) lc1 is the low power level for RF channel 1 (hex byte) nc2 is the normal power level for RF channel 2 (hex byte) lc2 is the low power level for RF channel 2 (hex byte) See section 5.4.6 for a description of legal power values depending on SoC used.
-ls <lbb0> <lbb1> <lbb2> <lbb3> <lbb4> <lbb5> <lbb6> <lbb7> <lbb8	Set lock bits bytes (hex). “lbbx” values should be used in hex format e.g. FA FF AA FF FF B0 FF FF FF See also section 4.4.6 regarding lock bits description. Notice that it is possible to change Auto Programming bits via command interface.
-lg	Get the 9 lock bit bytes (hex bytes, e.g. F8 FF FF FF FF FF FF FF FD). See also section 4.5.4 regarding lock bits description.
-sr <filename>.hex	Read the XRAM contents and write it to the Intel hex file specified. NB: Not possible using the USB interface.
-sw <filename>.hex	Write the contents of Intel hex file specified to the XRAM.
-sc <filename>.hex	Compare the XRAM contents with the Intel hex file specified.
-sweo <filename>.hex	Read the Intel hex file specified, write the contents to the XRAM and enable EooS mode.
-nv <filename>.hex	Read the Intel hex file specified and write the contents to the NVR memory.
-nV <filename>.hex	Read NVR and display contents as a Intel hex file on the console and write it to Intel hex file specified.

5.10.1.4 External Non-Volatile Memory (NVM) operands:

-pe <filename>.hex	Programs the external NVM with the file specified. Zero-fill empty areas.
-pe 0	Erase entire external NVM.
-pr <filename>.hex	Read contents of external NVM and write it to the Intel hex file specified.
-ph <HomeID>	Where <HomeID> (hex string, e.g. D234ADFF) is the Z-Wave Home ID to be written to the external NVM.

Note: the NVM operands are valid for all Z-Wave chips/modules, but are only valid when using the SPI programming interface.

5.10.2 Examples

Program the flash and set the frequency to US using SPI:

```
ZWaveProgrammer –c com1 –pf US –p target.hex
```

Erase the flash using UART:

```
ZWaveProgrammer –s com4 –e
```

Read the ASIC flash content to the file read.hex using USB:

```
ZWaveProgrammer –u com18 -r read.hex
```

Compare the content of ASIC flash to the content of the hex file c.hex using USB via CP210 to module UART:

```
ZWaveProgrammer –u com12 –v cp.hex
```

Write the content of the hex file eep.hex to the External NVM using SPI:

```
ZWaveProgrammer –c com1 –pe eep.hex
```

Reads the contents of the External NVM and write it to the file out.hex using UART:

```
ZWaveProgrammer –s com1 –pr out.hex
```

Change the home ID in the External NVM to 12345678 using USB:

```
ZWaveProgrammer –u com1 –ph 12345678
```

5.10.3 Console interface status codes

The ZWaveProgrammer returns an application return code when used in console mode. This allows an automation script to evaluate the result of a requested Z-Wave PC Programmer operation.

The return code may be displayed in a Windows Console by typing the command: “echo %errorlevel%”.

If the requested operation has been completed successfully, the return code is zero. The full list of return codes is shown below:

Table 10. PC Programmer console mode return codes

Return code	Explanation
0	Success
1	Could not compare External NVM content with the HEX file
2	Could not compare Flash content with the HEX file
3	Could not compare SRAM content with the HEX file
4	Could not detect the Z-Wave device
5	Could not erase the External NVM content
6	Could not erase the Flash content
7	Could not get the firmware version
8	Could not initialize External NVM
9	Could not read the Lock Bits
10	Could not set the Lock Bits. If you are trying to write lock bits not for the first time - erase device first. If the device is not erasable - replace it by a new one.
11	Could not read the application RF settings from Flash
12	Could not read the External NVM content
13	Could not read the External NVM options
14	Could not read the Flash content
15	Could not read the Flash options
16	Could not read the general options from Flash
17	Could not read the Home ID
18	Could not read the SRAM content
19	Cannot reset connected Z-Wave Module
20	Could not switch the device to BootLoader mode.
21	Could not switch the chip operation mode. Required operation mode may be disabled by a lock bit.
22	Could not switch the chip to Programming mode
23	Could not upgrade the firmware of the device
24	Could not write the application RF settings to Flash
25	Could not write content of the HEX file to External NVM

Return code	Explanation
26	Could not write content of the HEX file to Flash
27	Could not write the application RF settings to Flash
28	Could not write the general options to Flash
29	Could not write the Home ID
30	Could not write content of the HEX file to SRAM
31	Comparison of External NVM content with the HEX file failed
32	Comparison of SRAM content with the HEX file failed
33	The HEX file for External NVM is not specified or specified file does not exist.
34	The HEX file for Flash is not specified or specified file does not exist.
35	Hex file not valid. Address out of range.
36	Hex file not valid
37	End value for Home ID must be greater than Start value
38	RF frequency was not selected
39	'Start Home Id' or 'End Home Id' was not specified
40	Undefined general option was encountered
41	Could not initialize MTP
42	Could not read the MTP content
43	Could not erase the MTP content
44	Could not program MTP
45	Could not compare MTP content with the HEX file
46	Programming of MTP failed
47	Comparison of MTP content with the HEX file failed
48	HEX file for MTP not specified
49	Reading the HEX file failed
50	Read/Write operation timeout.
251	Other errors

5.11 Updating the Z-Wave programming hardware

5.11.1 Manual firmware upload

The ZDP03A firmware can be updated to a newer version via the Z-Wave PC Programmer GUI.

To open the firmware update dialog select **Tools > Upload Firmware...**

Select the Z-Wave Programmer firmware hex file and click ‘Open’.

If the upgrade was successful, the appropriate message will be displayed.

You can read the firmware version in the Status Bar at the bottom of the Z-Wave PC Programmer window.

5.11.2 Automatic firmware version check and update

Z-Wave Programmer checks PCB firmware version during startup. If the firmware version in the Programmer board is older than the version known to Z-Wave Programmer application, the following message will pop up:

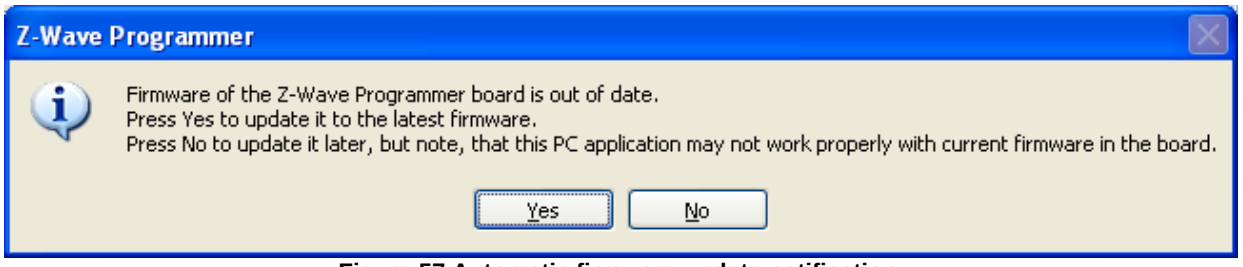


Figure 57.Automatic firmware update notification.

The user may choose to upgrade to the latest firmware version or to leave the current version running. However, the latter is not recommended.

REFERENCES

[1] Sigma Designs, INS10247, Instruction, Z-Wave ZW0102/ZW0201/ZW0301 Application Programming Guide v5.02 Patch2
[2] Sigma Designs, INS10795, Instruction, 400 Series Z-Wave Single Chip Programming Mode
[3] Sigma Designs, INS10579, Instruction, Programming the ZW0102 Flash and Lock Bits
[4] Sigma Designs, INS11681, Instruction, 500 Series Z-Wave Single Chip Programming Mode
[5] Sigma Designs, DSH11055, Datasheet, ZM4101 Z-Wave Module
[6] Sigma Designs, INS11681, Instruction, 500 Series Z-Wave Single Chip Programming Mode
[7] Sigma Designs, INS12524, Instruction, 500 Series Calibration User Guide