# Software Design Specification

## 500 Series Z-Wave Chip NVR Flash Page Contents

| | |
|---|---|
| **Document No.:** | SDS12467 |
| **Version:** | 18 |
| **Description:** | This document describes the 500 Series Z-Wave Chip NVR Flash Page Contents |
| **Written By:** | PSH;MVO;JFR;TRO;ANI;EFH;BBR;CRASMUSSEN;JBU;SSE |
| **Date:** | 2018-03-02 |
| **Reviewed By:** | ANI;PSH;MVO;JFR;JBU;CRASMUSSEN;OPP;LTHOMSEN |
| **Restrictions:** | Public |

**Approved by:**

| Date | CET | Initials | Name | Justification |
|---|---|---|---|---|
| 2018-03-02 | 15:47:19 | JFR | Jørgen Franck | on behalf of NTJ |

## REVISION RECORD

| Doc. Ver. | Date | By | Pages affected | Brief description of changes |
|---|---|---|---|---|
| 1 | 20130220 | PSH | ALL | Initial revision |
| 1 | 20130308 | PSH<br>JFR | ALL | Added NVR layout revision<br>Changed SAW filter according to review comments<br>Added description of CRC16 algorithm<br>Minor typos and CRC16 reference |
| 2 | 20130319 | TRO | Section 3.1 | Add IDVEN (Vendor ID) and IDPROD (Product ID) to the document |
| 2 | 20130429 | PSH | Section 3.1 | Added NVR fields for new sensitivity calibration values |
| 2 | 20130514 | MVO | All | Renamed fields, added CRC16 example, clean-up |
| 2 | 20130515 | ANI | Section 3.1 | Changed not calibrated CCAL value to 0x80 |
| 3 | 20130618 | MVithanage | Section 3.1 | Changed the REV field data format and added the current revision.<br>Added the Z-Wave products to the MTYP field data.<br>Changed the MREV field data format to match REV.<br>Added standard values for the regions in SAWC.<br>Added standard values for the regions in SAWB.<br>Extended the description of NVMP. |
| 3 | 20130618 | ANI | Section 3.1 | Corrected valid range |
| 3 | 20130619 | PSH | Section 3.1 | MTYP field changed to PINS field. MREV changed to NVMCS field.<br>Current revision changed to 0.1 |
| 4 | 20131013 | MVO | Section 3.1 | Now the CRC16 example is calculated over correct data range |
| 5 | 20140224 | MVO | Section 4 | Added mandatory/optional field info to the tables |
| 6 | 20140320 | MVO | Section 4<br>Section 3 | Corrected text<br>Added initial description of the NVR data area |
| 7 | 20140401 | EFH | Section 3.1 | Added NVMT=1, EEPROM with page size 256 |
| 7 | 20140403 | MVO | Appendix A<br>Table 4 | Added CRC16 calculation C source code and sample calculation<br>Corrected SAWC and SAWB fields for ZM5202 |
| 8 | 20140623 | JFR | Section 3.1 | Added limitations with respect to configuration of the NVM chip select pin in NVR (NVMCS field). |
| 9 | 20150809 | TRO | Section 3.1 | Add HW version |
| 10 | 20160707 | JBU | Section 3.2 | Added ECDH keypair to NVR. To be used by slave libraries without external NVM. |
| 11 | 20160801 | JBU | Section 3.2, 3.1,<br>Section 4<br>Section 5 | Elaborated PUK/PRK with generation algorithm. Removed PUK/PRK from section 3.2 since these fields don't exist in rev 0.1.<br>Added PUK/PRK as CM fields if S2 enabled protocol version is used.<br>Added code samples for PUK/PRK generation |
| 12 | 20161123 | JFR | Section 3.1.7 | Updated manufacturer list |
| 13 | 20170201 | JFR | Section 3.2 | PUK and PRK only stored in NVR for all slaves. |
| 14 | 20170206 | JBU | Section 5.2 | Added linux source code example |
| 15 | 20170428 | JBU | Section 5.2 | Better printing of linux source code example |
| 16 | 20170606 | SSE | Section 3.1.7 | Added a new value for the DATAFLASH NVM type from adesto<br>This type of chips have special power down mode |
| 16 | 20170619 | PSH | Section 3.1.7 | Corrected NVM type table wrt. NVMT = 0x03 |
| 17 | 20170814 | JFR | Section 3.1.9 | Truncation of non-volatile memory page size |
| 18 | 20180302 | BBR | All | Added Silicon Labs template |

# Table of Contents

# Table of Tables

# 1 ABBREVIATIONS

| Abbreviation | Explanation |
|---|---|
| CCAL | Crystal CALibration NVR field |
| CRC16 | Cyclic Redundancy Check, 16 bit NVR field |
| EPx | Erase and Programming protection lock bit |
| MREV | Module REVision NVR field |
| MSB | Most Significant Byte |
| PINS | Pin swap |
| NVMCS | NVM chip select |
| NVMP | Non-Volatile-Memory Page size NVR field |
| NVMS | Non-Volatile-Memory Size NVR field |
| NVMT | Non-Volatile-Memory Type NVR field |
| NVR | Non Volatile Registers |
| PID | USB Product IDentifier NVR field |
| RBAP | Read Back and Auto Programming mode lock bit |
| REV | NVR layout REVision NVR field |
| SAWB | SAW Bandwidth NVR field |
| SAWC | SAW Center frequency NVR field |
| TXCAL1 | Frequency Calibration 868.4MHz NVR field |
| TXCAL2 | Frequency Calibration 929.4MHz NVR field |
| UUID | Universally Unique IDentifier NVR field |
| VID | USB Vendor IDentifier NVR field |

# 2 INTRODUCTION

## 2.1 Purpose

This document defines the content of the NVR flash page in the 500 Series chips.

## 2.2 Audience and prerequisites

This document is targeted 500 Series Z-Wave chip programmer designers and Z-Wave SW designers.

# 3 NVR FLASH PAGE IN THE 500 SERIES

The NVR flash page is used to store NVM data throughout a Z-Wave product lifetime. The NVR data can be read from the MCU embedded in the 500 series chip. But it can only be programmed using a programming interface. The Z-Wave protocol and peripheral drivers utilize some of the NVR data for RF calibration data and for data that can be used to identify the peripherals. A part of the NVR data area can be used by the application, e.g. for calibration data, serial numbers etc.

## 3.1 NVR Flash page contents revision v.0.1

The NVR flash page, NVR0, contains the following fields:

| Byte | 0/8 | 1/9 | 2/A | 3/B | 4/C | 5/D | 6/E | 7/F |
|---|---|---|---|---|---|---|---|---|
| **0x00** | EP0 | EP1 | EP2 | EP3 | EP4 | EP5 | EP6 | EP7 |
| **0x08** | RBAP | | | | | | | |
| **0x10** | REV | CCAL | PINS | NVMCS | SAWC | | | SAWB |
| **0x18** | NVMT | NVMS | | NVMP | | UUID | | |
| **0x20** | UUID | | | | | | | |
| **0x28** | UUID | | | | | VID | | PID |
| **0x30** | PID | TXCAL1 | TXCAL2 | - | - | - | - | - |
| **…** | | | | | | | | |
| **0x78** | - | - | - | - | - | - | CRC16 | |
| **0x80** | HW | | | | | | | |
| **..** | | | | | | | | |
| **0xF8** | | | | | | | | |

Addresses from 0x00 to 0x08 are used for lock bits
Addresses from 0x10 to 0x7F are used by Z-Wave protocol and peripheral drivers.
Addresses from 0x80 to 0xFF are reserved for the application

Lock bits are described in the 500 series Z-Wave single chip programming mode document [1].

Note: All multi byte fields should be written with the MSB at the lowest address

### 3.1.1 REV – NVR layout revision (8bit)

This field contains the revision number of the NVR page layout. The layout described in this section corresponds to revision 1.

| REV | Description |
|---|---|
| `0xFF` | Unknown revision |
| `0x00` | v0.0 |
| `0x01` | v0.1 |
| `0x02` | v0.2 |
| . | . |
| . | . |
| . | . |
| `0x10` | v1.0 |
| `0x11` | v1.1 |
| . | . |
| . | . |

Standard usage:

| Current revision | REV |
|---|---|
| v0.1 | `0x01` |

### 3.1.2 CCAL – Crystal Calibration (8bit)

This byte contains the byte used calibrating the RF frequency after the variation in the crystal frequency. The value is a two-compliment number. Refer to [4] for a description of how to generate the calibration value.

| CCAL | Description |
|---|---|
| `0x80` | Not calibrated |
| `0x00-0x7F`<br>`0x81-0xFF` | Valid calibration value |

### 3.1.3 PINS – Pin Swap (8bit)

This byte contains information about pin swapping in the Z-Wave module/chip.

| PINS | Description |
|---|---|
| `0xFF` | Unknown pin swapping or no pin swapping |
| `0x00` | No pin swapping |
| `0x01` | ZM5202 mode. UART0 Tx is moved to P3.5, UART0 Rx is moved to P3.4 and PWM output is moved to P1.0 |
| `0x02-0xFE` | Reserved values |

### 3.1.4 NVMCS – NVM Chip Select (8bit)

This byte contains information about what pin we are using for chip select for the NVM chip. The 4 MSB are the port number and the 4 LSB are the pin number. Notice the limitations with respect to configuration of the NVM chip select pin in NVR.

| NVMCS | Description |
|---|---|
| `0xFF` | Unknown chip select |
| `0xXY` | X = Port Number, Y = Pin number<br>Supported port and pin numbers:<br>`NVMCS = 0x04 - PORT0.4`<br>`NVMCS = 0x14 - PORT1.4`<br>`NVMCS = 0x15 - PORT1.5`<br>`NVMCS = 0x17 - PORT1.7`<br>`NVMCS = 0x25 - PORT2.5`<br>`NVMCS = 0x30 - PORT3.0`<br>`NVMCS = 0x34 - PORT3.4`<br>`NVMCS = 0x36 - PORT3.6`<br>`NVMCS = 0x37 - PORT3.7` |

### 3.1.5 SAWC – SAW Center Frequency (24bit)

This field contains the center frequency mounted on the Z-Wave module. The center frequency is given as an integer in kHz. This field can, in combination with the SAWB field, be used to verify that the SAW filter complies with the frequency contained in the flash code image.

| SAWC | Description |
|---|---|
| `0xFFFFFF` | Unknown center frequency |
| `0x000000` | Direct connection (All frequencies will pass) |
| `0x000001-0xFFFFFD` | SAW filter center frequency |
| `0xFFFFFE` | No SAW filter mounted (RF won't work) |

Standard usage:

| Region | SAWC | $F_c$ |
|---|---|---|
| E | `0x0D3CDE` | 867550kHz |
| U | `0x0DEB7A` | 912250kHz |
| H | `0x0E1578` | 923000kHz |

### 3.1.6 SAWB – SAW Bandwidth (8bit)

This byte contains the bandwidth of the SAW filter mounted on the Z-Wave module. The bandwidth is given as an integer in MHz. This field can, in combination with the SAWC field, be used to verify that the SAW filter complies with the frequency contained in the flash code image.

| SAWB | Description |
|---|---|
| `0xFF` | Unknown bandwidth |
| `0x00-0xFE` | SAW filter bandwidth |

Standard usage:

| Region | SAWB | BW |
|---|---|---|
| E | `0x05` | 5MHz |
| U | `0x08` | 8MHz |
| H | `0x07` | 7MHz |

### 3.1.7 NVMT – Non-Volatile-Memory Type (8bit)

This byte contains the interface type of the NVM chip mounted on the Z-Wave module.

| NVMT | Type | Manufacturer |
|---|---|---|
| 0xFF | - | Unknown NVM chip |
| 0x00 | - | No External NVM chip |
| 0x01 | EEPROM | Atmel Corporation: AT25256B<br>ON Semiconductor: CAT25256 |
| 0x02 | FLASH | Micron Technology: M25PE10, M25PE20, M25PE40 & M25PE80<br>Adestotech (enters only deep sleep): AT45DB021E, AT45DB041E, AT45DB081E, AT45DB161E, AT45DB321E & AT45DB641E |
| 0x03 | EEPROM | ON Semiconductor: CAT25M01 |
| 0x04 | DATAFLASH | Adestotech (enters ultra-deep sleep): AT45DB021E, AT45DB041E, AT45DB081E, AT45DB161E, AT45DB321E & AT45DB641E |

Refer to [5] for details about recommended external NVM components.

### 3.1.8 NVMS – Non-Volatile-Memory Size (16bit)

This field contains the size of the NVM chip mounted on the Z-Wave module. The size is given in kB (kilo bytes)

| NVMS | Description |
|---|---|
| 0xFFFF | Unknown NVM size |
| 0x0000-0xFFFE | NVM size |

### 3.1.9 NVMP – Non-Volatile-Memory Page Size (16bit)

This field contains the page size of the NVM chip mounted on the Z-Wave module. The size is given in bytes.

FLASH: This value represents the smallest erasable size.
EEPROM: Size of the write buffer.

| NVMP | Description |
|---|---|
| 0xFFFF | Unknown NVM page size |
| 0x0000-0x0100 | NVM page size |

Notice: Enter 256 bytes (0x0100) in case page size is bigger than 256 bytes in datasheet.

### 3.1.10 UUID – Universally Unique Identifier (128bit)

This field contains a universally unique identifier that is used for giving the USB interface a unique identifier .This makes it possible for the OS, to which the Z-Wave module is connected, to uniquely identify each Z-Wave device when more than one Z-Wave device is connected via USB.

**NOTICE:** Optional to set UUID in UZB devices.

| UUID | Description |
|---|---|
| `0xFF..FF` | No UUID |
| `0xYY.YY` | Valid UUID |

### 3.1.11 VID – USB Vendor ID (16bit)

This field contains the USB Vendor ID. The Vendor ID is assigned by USB Implementers Forum Inc.

The Silicon Labs Vendor ID, 0x0658, is used if this field is 0xFFFF.

NVR Vendor ID values will not be used if Vendor ID is changed by calling USB API interface.

| VID | Description |
|---|---|
| `0xFFFF` | Uses Silicon Labs Vendor ID 0x0658 |
| `0x0000-0xFFFE` | Vendor specific ID |

### 3.1.12 PID – USB Product ID (16bit)

This field contains the USB Product ID. Product ID is assigned by the end-product manufacturer. Silicon Labs Product ID for USBVCP protocol, 0x0200, is used if this field is 0xFFFF.

NVR Product ID values will not be used if Product ID is changed by calling USB API interface.

Please refer to [3] for Silicon Labs Vendor ID’s.

| PID | Description |
|---|---|
| `0xFFFF` | Uses Silicon Labs Product ID 0x0200 (UZB) |
| `0x0000-0xFFFE` | Product specific ID |

### 3.1.13 TXCAL1 – Frequency Calibration (8bit)

This byte contains a value that, together with TXCAL2, is used for calibrating the RF Tx frequency

| TXCAL1 | Description |
|---|---|
| `0xFF` | Not Calibrated |
| `0x00-0xFE` | Calibration Value |

### 3.1.14 TXCAL2 – Frequency Calibration (8bit)

This byte contains a value that, together with TXCAL1, is used for calibrating the RF Tx frequency

| TXCAL2 | Description |
|---|---|
| `0xFF` | Not Calibrated |
| `0x00-0xFE` | Calibration Value |

### 3.1.15 CRC16 – CRC (16bit)

This word contains a CRC16 value calculated from address 0x10 to address 0x7D. If the CRC16 is not correct all fields in the NVR should be read as 0xFF. The CRC16 is calculated using a CRC-CCITT algorithm with an initialization value of 0x1D0F [2].

| CRC16 | Description |
|---|---|
| `0xYYYY` | CRC16 value |

The CRC16 value is 0xC13D when the value of all the bytes from form address 0x10 to 0x7D is 0xFF. That is, the content of address 7Eh becomes 0xC1 and the content of address 0x7F becomes 0x3D.

### 3.1.16 HW – Hardware version (8 bit), Application area

The Hardware Version field is unique to this particular version of the product. It is reported in command class Version Report. Please read information about hardware version document SDS12652. The Framework is capable to read the hardware version (CommandClassVersion.c).

| HW | Description |
|---|---|
| `0x00-0xFF` | HW version value is version of the entire product. |

## 3.2 NVR Flash page contents revision v.0.2

The NVR flash page, NVR0, contains the following fields:

| Byte | 0/8 | 1/9 | 2/A | 3/B | 4/C | 5/D | 6/E | 7/F |
|---|---|---|---|---|---|---|---|---|
| **0x00** | EP0 | EP1 | EP2 | EP3 | EP4 | EP5 | EP6 | EP7 |
| **0x08** | RBAP | | | | | | | |
| **0x10** | REV | CCAL | PINS | NVMCS | SAWC | | | SAWB |
| **0x18** | NVMT | NVMS | | NVMP | | UUID | | |
| **0x20** | UUID | | | | | | | |
| **0x28** | UUID | | | | | VID | | PID |
| **0x30** | PID | TXCAL1 | TXCAL2 | PUK1 | PUK2 | PUK3 | PUK4 | PUK5 |
| **…** | | | | | | | | |
| **0x50** | PUK30 | PUK31 | PUK32 | PRK1 | PRK2 | PRK3 | PRK4 | PRK5 |
| **…** | | | | | | | | |
| **0x70** | PRK30 | PRK31 | PRK32 | - | - | - | - | - |
| **0x78** | - | - | - | - | - | - | CRC16 | |
| **0x80** | HW | | | | | | | |
| **..** | | | | | | | | |
| **0xF8** | | | | | | | | |

Addresses from 0x00 to 0x08 are used for lock bits
Addresses from 0x10 to 0x7F are used by Z-Wave protocol and peripheral drivers.
Addresses from 0x80 to 0xFF are reserved for the application

Lock bits are described in the 500 series Z-Wave single chip programming mode document [1].

Note: All multi byte fields should be written with the MSB at the lowest address

### 3.2.1 REV – NVR layout revision (8bit)

This field contains the revision number of the NVR page layout. The layout described in this section corresponds to revision 2.

| REV | Description |
|---|---|
| `0xFF` | Unknown revision |
| `0x00` | v0.0 |
| `0x01` | v0.1 |
| `0x02` | v0.2 |
| . | . |
| . | . |
| . | . |
| `0x10` | v1.0 |
| `0x11` | v1.1 |
| . | . |
| . | . |

Standard usage:

| Current revision | REV |
|---|---|
| v0.2 | `0x02` |

### 3.2.2 PUK – Public Key for Security2 (256bit)

The PUK field contains the Public key of the Curve25519 ECDH Keypair used during S2 inclusion.

PRK contains the private key. The PUK and PRK MUST always are stored in the NVR of S2 enabled devices.

Algorithm for creating the PUK and PRK fields

1. Fill PRK with 32 bytes of high-quality random numbers. A high quality can be obtained on linux by reading from /dev/urandom or using the getrandom(2) syscall. On Windows, the System.Security.Cryptography.RNGCryptoServiceProvider or CryptGenRandom() function can be used.
2. Call crypto_scalarmult_curve25519_base(PUK, PRK) to compute PUK. The source code for crypto_scalarmult_curve25519_base can be obtained from keypair_generation_curve25519.zip included with SDK 6.70.
3. Once a module has been programmed with a PUK and PRK, the first 16 bytes of the PUK must be stored and printed on the packaging or casing of that particular module. The PRK MUST be discarded along with all log files listing it after the production run is complete.

Detailed instructions on random number generation and PUK computation can be found in Section 5.

It is RECOMMENDED to repeat this algorithm *before* production starts and create a collection of (PUK, PRK) values sufficient for the entire production run. The PRKs MUST is discarded after production. The PUKs MUST is retained and printed on the module and finished product.

### 3.2.3 PRK – Private Key for Security2 (256bit)

The private part of the Curve25519 key pair used for key exchange in Security2.
See section 3.2.2 for a detailed description.

# 4 NVR FLASH PAGE HANDLING IN PRODUCTION

The fields in the Z-Wave protocol and peripheral driver part (addresses 0x10-0x7F) of the NVR flash page, NVR0, are divided in fields that are mandatory to be set, fields that are conditionally mandatory to be set and in fields that can be set as an option. It is the Z-Wave end-product manufacturer's responsibility to check that the fields have been set correctly.

The fields REV, CCAL, PINS, TXCAL1, TXCAL2 and CRC16 must be set in the NVR Flash page, NVR0, for all 500 Series Z-Wave Chip based products.

The fields NVMT, NVMS and NVMP must be set in the NVR Flash page in all 500 Series Z-Wave Chip based products that has an external NVM.

Additionally, the fields VID and PID must be written to the NVR Flash page in all 500 Series Z-Wave Chip based products where a vendor specific USB interface is utilized.

Optionally, the field UUID can be set in the NVR Flash page during production of a 500 Series Z-Wave Chip based products where the USB interface is utilized. It is also optional to set the SAWC and SAWB fields.

All fields that are not set to a specific value should be set to 0xFF (the contents after erase)

The table below shows a summary of what fields that should be programmed in different Z-Wave module types:

**Table 1, Mandatory and optional NVR fields 1/3**

| | REV | CCAL | PINS | NVMCS | SAWC | SAWB | NVMT | NVMS |
|---|---|---|---|---|---|---|---|---|
| **ZM5304** | M | M | M | M | O | O | M | M |
| **ZM5202** | M | M | M | CM | O | O | CM | CM |
| **ZM5101** | M | M | M | CM | O | O | CM | CM |
| **SD3502** | M | M | M | CM | O | O | CM | CM |
| **SD3503** | M | M | M | CM | O | O | CM | CM |
| M | Mandatory. Must be programmed with a valid value | | | | | | | |
| CM | Conditional Mandatory. Must be programmed with a valid value if an external NVM is connected to the 500 Series module/chip | | | | | | | |
| O | Optional. Can optionally be programmed with a valid value | | | | | | | |

**Table 2, Mandatory and optional NVR fields 2/3**

| | NVMP | UUID | VID | PID | TXCAL1 | TXCAL2 | PUK | PRK |
|---|---|---|---|---|---|---|---|---|
| **ZM5304** | M | O | CM | CM | M | M | CM | CM |
| **ZM5202** | CM | O | CM | CM | M | M | CM | CM |
| **ZM5101** | CM | O | CM | CM | M | M | CM | CM |
| **SD3502** | CM | O | CM | CM | M | M | CM | CM |
| **SD3503** | CM | O | CM | CM | M | M | CM | CM |
| M | Mandatory. Must be programmed with a valid value | | | | | | | |
| CM | Conditional Mandatory. Must be programmed with a valid value if an external NVM is connected to the 500 series module/chip | | | | | | | |
| CM | Conditional Mandatory. Must be programmed with a valid value if a vendor specific USB interface is utilized | | | | | | | |
| CM | Conditional Mandatory. Must be programmed with a valid value if an S2-enabled protocol is used. | | | | | | | |
| O | Optional. Can optionally be programmed with a valid value | | | | | | | |
| D | Default. Must have the default value 0xFF | | | | | | | |

**Table 3, Mandatory and optional NVR fields 3/3**

| | CRC16 | *Other* | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **ZM5304** | M | D | | | | | | |
| **ZM5202** | M | D | | | | | | |
| **ZM5101** | M | D | | | | | | |
| **SD3502** | M | D | | | | | | |
| **SD3503** | M | D | | | | | | |

| | |
|---|---|
| 🟩 M | Mandatory. Must be programmed with a valid value |
| 🟨 CM | Conditional Mandatory. Must be programmed with a valid value if an external NVM is connected to the 500 series module/chip |
| 🟥 CM | Conditional Mandatory. Must be programmed with a valid value if a vendor specific USB interface is utilized |
| 🟦 CM | Conditional Mandatory. Must be programmed with a valid value if an S2-enabled protocol is used. |
| 🟧 O | Optional. Can optionally be programmed with a valid value |
| D | Default. Must have the default value 0xFF |

The 500 Series Z-Wave Modules will have some fields pre-programmed by Silicon Labs. The NVR flash page of the SD3502 and SD3503 chips are not programmed. The following table shows which fields that are pre-programmed in the 500 Series Z-Wave modules (including the Z-Wave development modules):

**Table 4, Pre-programmed NVR fields 1/2**

| | REV | CCAL | PINS | NVMCS | SAWC | SAWB | NVMT | NVMS |
|---|---|---|---|---|---|---|---|---|
| **ZM5304** | P | P | P | P | P | P | P | P |
| **ZM5202** | P | P | P | D | P | P | P | D |
| **ZM5101** | P | P | P | D | D | D | D | D |
| **SD3502** | D | D | D | D | D | D | D | D |
| **SD3503** | D | D | D | D | D | D | D | D |
| **ZDB5202** | P | P | P | P | P | P | P | P |
| **ZDB5304** | P | P | P | P | P | P | P | P |
| **ZDB5101** | P | P | P | P | P | P | P | P |
| **UZB** | P | P | P | P | P | P | P | P |

| | |
|---|---|
| 🟩 P | Pre-programmed with a valid value |
| D | Default value (0xFF) |

**Table 5, Pre-programmed NVR fields 2/2**

| | NVMP | UUID | VID | PID | TXCAL1 | TXCAL2 | CRC16 | *Other* |
|---|---|---|---|---|---|---|---|---|
| **ZM5304** | P | D | D | D | P | P | P | D |
| **ZM5202** | D | D | D | D | P | P | P | D |
| **ZM5101** | D | D | D | D | D | D | P | D |
| **SD3502** | D | D | D | D | D | D | D | D |
| **SD3503** | D | D | D | D | D | D | D | D |
| **ZDB5202** | P | D | D | D | P | P | P | D |
| **ZDB5304** | P | D | D | D | P | P | P | D |
| **ZDB5101** | P | D | D | D | P | P | P | D |
| **UZB** | P | D | P | P | P | P | P | D |

| | |
|---|---|
| 🟩 P | Pre-programmed with a valid value |
| D | Default value (0xFF) |

## 4.1 NVR Flash page handling in programmer

The NVR page is erased when a Flash "Erase Chip" is performed, so the programming software must preserve the NVR contents when programming the chip. All Z-Wave programmers must always read the NVR flash page before running the "Erase chip" programming command, and then optionally modify the NVR fields, and finally write the NVR contents including the modified CRC16 field back to the NVR page.

# 5 PUK AND PRK CREATION

The PUK and PRK is a Curve25519 keypair. The procedure for generating keypairs is slightly different depending on the platform used.

## 5.1 Windows

The following example shows how the PC Programmer creates a keypair:

```
public static void GenerateKeys(byte[] priv, byte[] pub)
{
  RNGCryptoServiceProvider rnd = new RNGCryptoServiceProvider();
  rnd.GetBytes(priv);
  CryptoScalarmultCurve25519Base(pub, priv);
}

public static void GenerateOneKeyPair()
{
  byte[] priv = new byte[32];
  byte[] pub = new byte[32];
  GenerateKeys(priv, pub);
}
```

CryptoScalarmultCurve25519Base is an external call to our s2crypto32.dll (or s2crypto64.dll) method crypto_scalarmult_curve25519_base. DLL binaries are provided as a part of Z-Wave PC based Controller release and is situated in the folder …\Source\Libraries\ZWaveDll\lib\CryptoLibraryS2:

- s2crypto32.dll – Windows 32bit version
- s2crypto64.dll – Windows 64bit version

In C# and Mono projects, CryptoScalarmultCurve25519Base can be implemented like this:

```
[DllImport("s2crypto32", EntryPoint = "crypto_scalarmult_curve25519_base",
CallingConvention = CallingConvention.Cdecl)]

private extern static void CryptoScalarmultCurve25519Base_32([Out,
MarshalAs(UnmanagedType.LPArray, SizeConst = KeySize)] byte[] public_key,[In,
MarshalAs(UnmanagedType.LPArray, SizeConst = KeySize)] byte[] private_key);

[DllImport("s2crypto64", EntryPoint = "crypto_scalarmult_curve25519_base",
CallingConvention = CallingConvention.Cdecl)]

private extern static void CryptoScalarmultCurve25519Base_64([Out,
MarshalAs(UnmanagedType.LPArray, SizeConst = KeySize)] byte[] public_key,[In,
MarshalAs(UnmanagedType.LPArray, SizeConst = KeySize)] byte[] private_key);

private static void CryptoScalarmultCurve25519Base(byte[] public_key, byte[]
private_key)
{
     if (IntPtr.Size == 8) // 64 bit process.
     {
         CryptoScalarmultCurve25519Base_64(public_key, private_key);
     }
     else
     {
         CryptoScalarmultCurve25519Base_32(public_key, private_key);
     }
}
```

## 5.2 Linux

Algorithm for creating the PUK and PRK fields on Linux

1. Fill PRK with 32 bytes of high-quality random numbers. A high quality can be obtained by reading from /dev/urandom or using the getrandom(2) syscall.
2. Call crypto_scalarmult_curve25519_base(PUK, PRK) to compute PUK. The source code for crypto_scalarmult_curve25519_base can be obtained from Tools\KeypairGeneration directory included with SDK 6.71.xx.

The following sample code illustrates how to generate a keypair. It must be linked with the source code mentioned in step 2 above.

```
#define _GNU_SOURCE
#include <unistd.h>
#include <sys/syscall.h>
#include <stdint.h>
#include <stdio.h>
#include <arpa/inet.h>
#define DllExport extern

#include "curve25519.h"

static void print32(uint8_t *p)
{
    for (int i=0; i<32; i++)
    {
        printf("%02x", p[i]);
    }
    printf("\n");
}

static void print_dsk(uint8_t *p)
{
    int i;
    uint16_t *n;
    printf("DSK:         zws2dsk:");
    n = (uint16_t *)p;
    for (i = 0; i < 8; i++)
    {
        printf("%05d", htons(*n));
        n++;
        if(i < 7)
        {
            printf("-");
        }
    }
    printf("\n");
}
```

```
int main()
{
    static uint8_t private_key[32], public_key[32];
    long r = syscall(SYS_getrandom, private_key, 32, 0);
    if (r != 32) {
        printf("Error obtaining random number, aborting.");
        return -1;
    }
    crypto_scalarmult_curve25519_base(public_key, private_key);
    printf("Public key:  ");
    print32(public_key);
    printf("Private key: ");
    print32(private_key);
    print_dsk(public_key);
    return 0;
}
```

# APPENDIX A CRC16 CALCULATION

### Appendix A.1 C source code example

```c
#include <stdio.h>
#include <stdint.h>

/*===========================   Defines   ============================*/
#define NVR_START            0x10
#define NVR_END              0x7F
#define POLY                 0x1021          /* crc-ccitt */
#define REV_ADR              0x10
#define CRC16_MSB_ADR        0x7E
#define CRC16_LSB_ADR        0x7F

/*===========================   ccittCrc16   ============================
 *    CRC-CCITT (0x1D0F) calculation / check
 *
 *    In:  byte string excluding 16bit check field
 *    Out: CRC-16 value
 *  or
 *    In:  byte string including 16bit check field
 *    Out: zero when OK
 *
 *  and
 *    In: The crc input should be set to the initialization
 *        value = 0x1D0F.
 *        It can also be used to carry over crc value between separate
 *        calculations of multiple parts of a frame, e.g. header and body.
 *
 *---------------------------------------------------------------------------
*/
uint16_t
ccittCrc16(
  uint16_t crc,
  uint8_t *pDataAddr,
  uint16_t bDataLen
  )
{
  uint8_t WorkData;
  uint8_t bitMask;
  uint8_t NewBit;

  while(bDataLen--)
  {
    WorkData = *pDataAddr++;

    for (bitMask = 0x80; bitMask != 0; bitMask >>= 1) {
      /* Align test bit with next bit of the */
      /* message byte, starting with msb.    */
      NewBit = ((WorkData & bitMask) != 0) ^ ((crc & 0x8000) != 0);
      crc <<= 1;
      if (NewBit) {
        crc ^= POLY;
      }
    } /* for (bitMask = 0x80; bitMask != 0; bitMask >>= 1) */
  }
  return crc;
}
/*==========================   main   ===============================
 * Run the ccittCrc16 function on a NVR array where all data is 0xFF
 * except for the REV field which is 0x01
 * ---------------------------------------------------------------*/
int main(void)
{
  uint8_t bNVRArray[256];
  uint16_t wCRC16;
  uint16_t i;

    /* Initialize NVR array to 0xFF's */
  for (i=0;i<256;i++)
  {
    bNVRArray[i]=0xFF;
  }
    /* Set Revision field to 0x01 */
  bNVRArray[REV_ADR]=0x01;

    /* Run CRC16 calculation from address NVR_START */
    /* to the data byte before the CRC16 field      */
  wCRC16 = ccittCrc16( 0x1D0F, &bNVRArray[NVR_START], NVR_END-NVR_START+1-2);

    /* store CRC16 value in NVR array */
  bNVRArray[CRC16_MSB_ADR]=(unsigned char)((wCRC16>>8)&0x00FF);
  bNVRArray[CRC16_LSB_ADR]=(unsigned char)(wCRC16&0x00FF);

    /* print out CRC16 value */
  printf("CRC16: %02X%02X\n", bNVRArray[CRC16_MSB_ADR],
bNVRArray[CRC16_LSB_ADR]);


    /* Check CRC16 value by running CRC calculation */
    /* from address NVR_START to the CRC16 field    */
  wCRC16 = ccittCrc16( 0x1D0F, &bNVRArray[NVR_START], NVR_END-NVR_START+1);

    /* Print out result */
  if (wCRC16==0)
    printf("CRC16 verified\n");
  else
    printf("Error: CRC16 mismatch\n");

  return 0;
}
```

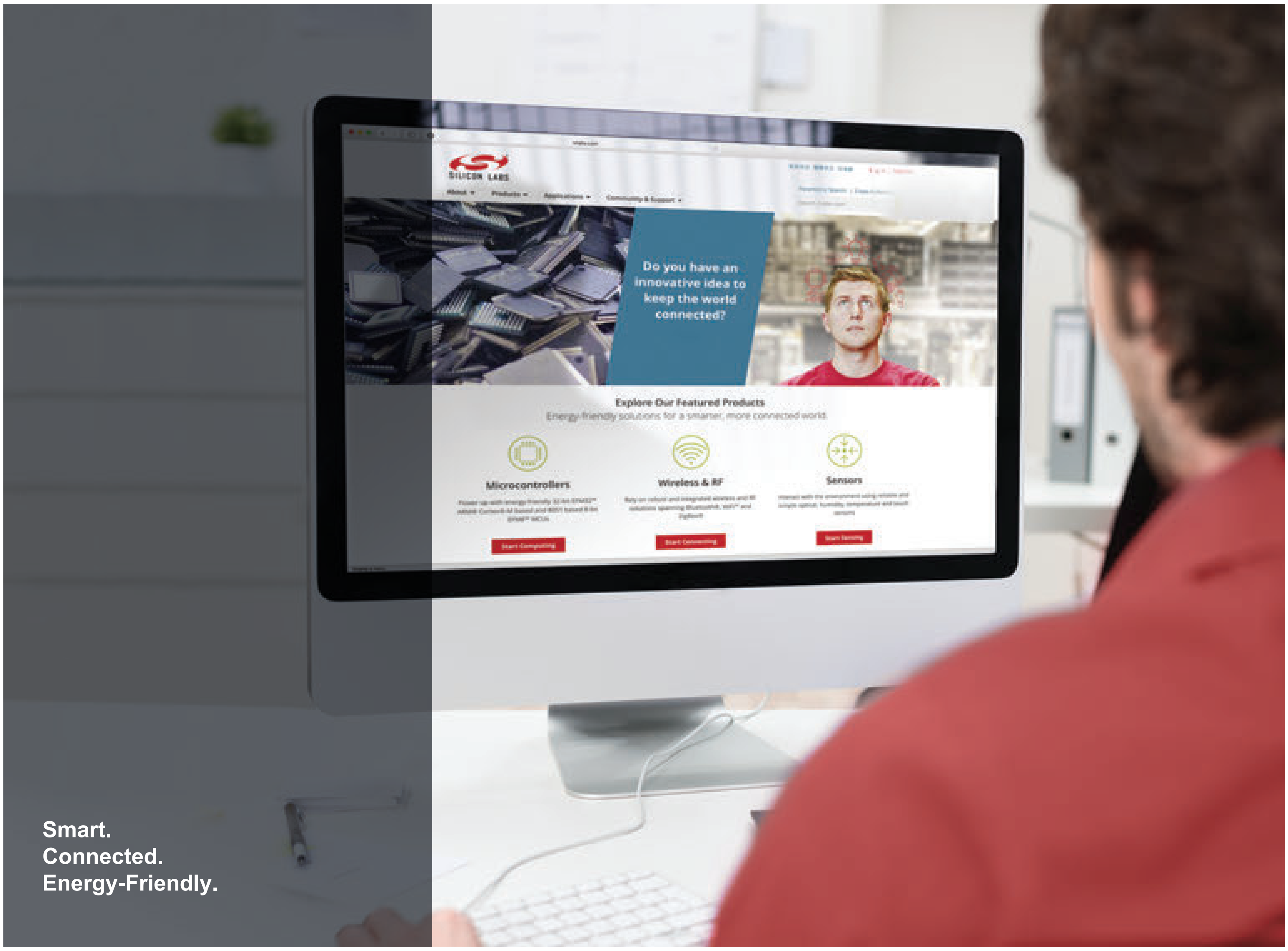
## Appendix A.2 Sample calculation

The CRC16 value is C564 for a NVR where the contents of address 0x10 is 0x01 and all other are 0xFF as seen below.

```
     0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
10: 01 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
20: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
30: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
40: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
50: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
60: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
70: FF FF FF FF FF FF FF FF FF FF FF FF FF FF C5 64
```

## REFERENCES

[1] Silicon Labs, INS11681, Instruction, 500 Series Z-Wave single Chip Programming Mode.
[2] Silicon Labs, SDS11060, Software Design Specification, Z-Wave Command Class Specification.
[3] Silicon Labs, INS12196, Instruction, USB Product Identifiers.
[4] Silicon Labs, INS12524, Instruction, 500 Series Calibration User Guide.
[5] Silicon Labs, INS12213, Instruction, 500 Series Integration Guide.


Smart.
Connected.
Energy-Friendly.
Do you have an innovative idea to keep the world connected?
Explore Our Featured Products
Microcontrollers
Wireless & RF
Sensors

**Products**
*www.silabs.com/products*

**Quality**
*www.silabs.com/quality*

**Support and Community**
*community.silabs.com*

SILICON LABS

**Silicon Laboratories Inc.**
**400 West Cesar Chavez**
**Austin, TX 78701**
**USA**

**http://www.silabs.com**