Programming EEPROM for the ZL3026x and ZL4025x Family of Devices Using the I^{2}C Interface

1. Introduction

This application note explains how to program the EEPROM of the ZL30261/3/5/7 and ZL40251/3 devices using the I^{2}C interface. This is a subset of the “EESEL Register Write Method” described in ZLAN-659. Please refer to ZLAN-675 for a description of using the SPI interface to program the EEPROM using the “EESEL Register Write Method.”

The goal of this application note is to guide the user in generating the correct I^{2}C sequence to program the EEPROM of a device so that when it is reset or powered on, it will start up with a pre-programmed configuration.

2. Background

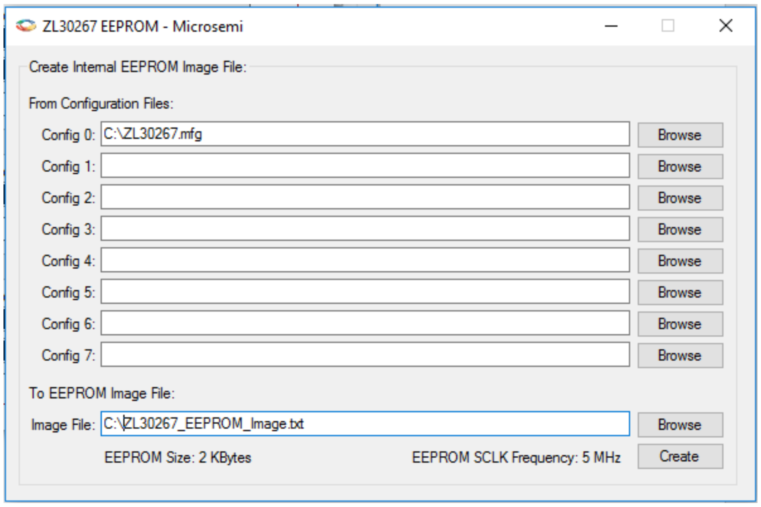
Configuration Files

Configuration files are generated by the GUI. After the device has been configured via the GUI to the desired state, the user selects **File→Save** Configuration, and the configuration file is generated. The configuration file contains the register write sequence including any delays that are required to configure the device. It configures things such as input frequencies, output frequencies, output channels used, signal formats used, etc. A configuration file is also known as a .mfg file, after the file name extension.

EEPROM Images

The ZL30261/3/5/7 and ZL40251/3 devices contain an internal EEPROM that can be programmed to store multiple configurations. Up to eight different device configurations can be saved in the EEPROM. At reset time, if the EEPROM is programmed, the device will be configured based on one of the stored configurations. The configuration that is loaded is determined by the settings on the AC0/AC1/AC2 pins of the device when RSTN goes high.

EEPROM Image files are generated by the GUI. When the User selects **EEPROM→Create EEPROM Image File** in the GUI, a dialog box like the one below appears:



Create Internal EEPROM Image File:

From Configuration Files:

Config 0: C:\ZL30267.mfg

Config 1:

Config 2:

Config 3:

Config 4:

Config 5:

Config 6:

Config 7:

To EEPROM Image File:

Image File: C:\ZL30267_EEPROM_Image.txt

EEPROM Size: 2 KBytes

EEPROM SCLK Frequency: 5 MHz

The user can enter up eight configuration (.mfg) files through this interface. When the user clicks **Create**, an image file with all eight configurations is created. If a particular configuration is left blank, it will be saved in the EEPROM image as blank. If a blank configuration is chosen by the AC0/AC1/AC2 pins, the device will start up in the default reset state. Several formats are available for file creation. The procedure below assumes the file is saved in the default format, as an ASCII text file with the .txt file extension. The .txt format contains comments at the beginning of the file, then one byte in hex format per line, beginning with the byte corresponding to EEPROM address 0x0000.

3. Procedure for loading EEPROM file

Definitions:

The following definitions apply to all I^{2}C bus transaction diagrams in this application note:

- START condition = I^{2}C Master transitions SDA from high to low while SCL is held high
- STOP condition = I^{2}C Master transitions SDA from low to high while SCL is held high
- A = Acknowledge = I^{2}C Slave (ZL3026x or ZL4025x) pulls SDA low
- Write Command = I^{2}C Master transmits 0x02 on SDA
- Write Enable Command = I^{2}C Master transmits 0x06 on SDA
- Read Command = I^{2}C Master transmits 0x03 on SDA

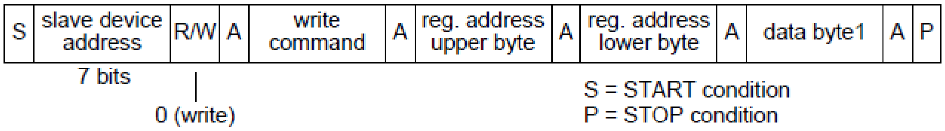
Note that all register address and all data bytes are transmitted most significant bit first on the SDA signal.

Step 1: Parse the EEPROM Image file into 32-byte blocks

The EEPROM write buffer in the device is 32 bytes long. This is the buffer used to transfer data from the SPI interface to the physical EEPROM. To write the whole EEPROM, the buffer must be filled multiple times using data from the EEPROM image file generated by the GUI, and the contents transferred to the physical EEPROM. The EEPROM image file should be stripped of comments, and then organized into 32-byte blocks. All lines that begin with a semicolon character (“;”) are comments. The procedure outlined below will load the EEPROM with a series of 32-byte bursts. The EEPROM image has 2036 bytes, so it should be split into 63 32-byte blocks and one final 20-byte block.

Step 2: Make the EEPROM accessible to the I^{2}C interface

The EEPROM is mapped into the same space as the device registers. Access to the EEPROM is gained by setting register 0x0000 to 0x80. To write 0x80 to register 0x0000 using the I^{2}C interface, the I^{2}C Write Transaction must be used as shown below.



S = START condition
P = STOP condition
A = acknowledge (SDA low)
NA = not acknowledge (SDA high)

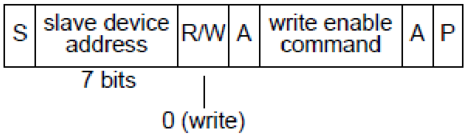
Where:

- Register Address Upper Byte = 0x00
- Register Lower Byte = 0x00
- Data Byte 1 = 0x80

Note Data Byte 1 is the only data byte written; therefore, it is immediately followed by the STOP condition.

Step 3: Enable the EEPROM for Writing

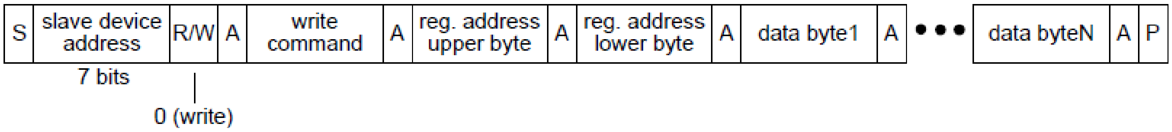
The EEPROM must be enabled for writing before *each* write transaction. This procedure uses a 32-byte burst for writing to the EEPROM. Therefore, a write enable command must be sent before each 32-byte burst. The following sequence enables the EEPROM for writing:



S = START condition
P = STOP condition
A = acknowledge (SDA low)
NA = not acknowledge (SDA high)

Step 4: Send a 32-byte burst of data

Send the first 32-byte block of data to the EEPROM, as shown in the transaction below. The starting address is 0x0000.



Where:

- Data Byte 1 corresponds to EEPROM address 0x0000

- Data Byte N = Data Byte 32, which corresponds to EEPROM address 31 (0x001F hex)

Step 5: Delay

Delay for a minimum of 5 ms to ensure the EEPROM write cycle completed.

Step 6: Repeat steps 3, 4, and 5

Steps 3, 4, and 5 should be repeated for each of the 32-byte blocks parsed from the EEPROM image file. On each iteration, the starting address and corresponding data bytes should be incremented by 32. For example, on the second iteration, the address is 0x0020. On the third iteration, the address is 0x0040, and so on.

Step 7: Send the last block

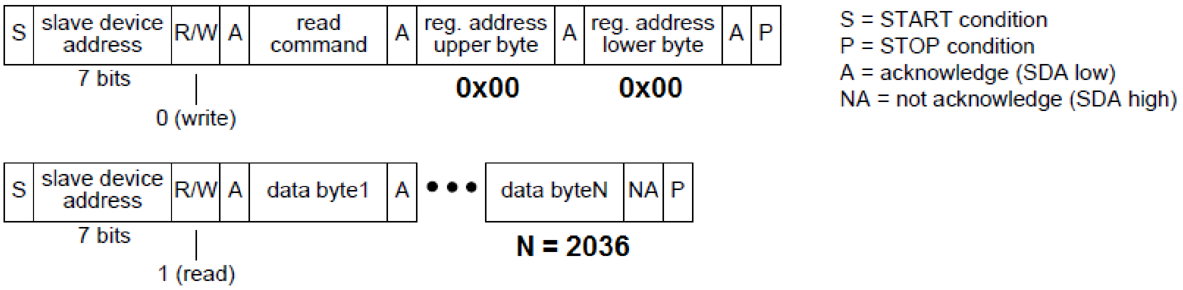
The final block of the Image file contains only 20 bytes. This final block should be written to the EEPROM the same way as the other blocks, but with a smaller 20-byte burst. In other words,

- The address for this final transaction is 0x07E0, and
- Data Byte N of Step 4 = Data Byte 20

Note that address range 0x07F4 to 0x07FF is reserved for factory traceability and test information. The user has the ability to overwrite these bytes if desired, but Microsemi recommends against this. Therefore, to help prevent an accidental overwrite of these registers by the user, address range 0x07F4 to 0x07FF is not contained in the EEPROM image file.

Step 8: Read back and verify the contents of the EEPROM

In order to verify that the EEPROM has been written correctly, the contents of the EEPROM should be read back and compared with the EEPROM image file. To read back the EEPROM contents beginning with address 0x0000, the bus master first does an I^{2}C write to the device. In this transaction three bytes are written: The Read command (0x03), the upper byte of the register address (0x00), and the lower byte of the register address (0x00). The bus master then does an I^{2}C read. During each acknowledge (A) bit the device fetches data from the read address and then increments the read address. The device then transmits the data to the bus master during the next 8 SCL cycles. The bus master terminates the read with a not-acknowledge (NA) followed by a STOP condition (P). After the I^{2}C write there can be unlimited idle time on the bus before the I^{2}C read, but the device cannot tolerate other I^{2}C bus traffic between the I^{2}C write and the I^{2}C read. Care must be taken to ensure that the I^{2}C read is the first command on the bus after the I^{2}C write to ensure the two-part read transaction happens correctly, as shown in the sequence below:



Step 9: Unmap the EEPROM from address 0x0000

Restore access to the register map rather than the EEPROM by writing 0x0000 to address 0x00. This procedure is identical to Step 2 above, with the exception that Data Byte 1 = 0x00. This unmapping can also be achieved by resetting or power-cycling the device.

Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high–performance and radiation–hardened analog mixed–signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power–over–Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif., and has approximately 4,800 employees globally. Learn more at **www.microsemi.com**.

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA

Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

E-mail: sales.support@microsemi.com

Microsemi makes no warranty, representation, or guarantee regarding the information contained herein or the suitability of its products and services for any particular purpose, nor does Microsemi assume any liability whatsoever arising out of the application or use of any product or circuit. The products sold hereunder and any other products sold by Microsemi have been subject to limited testing and should not be used in conjunction with mission-critical equipment or applications. Any performance specifications are believed to be reliable but are not verified, and Buyer must conduct and complete all performance and other testing of the products, alone and together with, or installed in, any end-products. Buyer shall not rely on any data and performance specifications or parameters provided by Microsemi. It is the Buyer's responsibility to independently determine suitability of any products and to test and verify the same. The information provided by Microsemi hereunder is provided "as is, where is" and with all faults, and the entire risk associated with such information is entirely with the Buyer. Microsemi does not grant, explicitly or implicitly, to any party any patent rights, licenses, or any other IP rights, whether with regard to such information itself or anything described by such information. Information provided in this document is proprietary to Microsemi, and Microsemi reserves the right to make any changes to the information in this document or to any products and services at any time without notice.