Programming EEPROM for the ZL3026x and ZL4025x Family of Devices

1. Introduction

This application note explains how to program the EEPROM of the ZL30261/3/5/7 and ZL40251/3 devices using the SPI interface.

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

2. Background

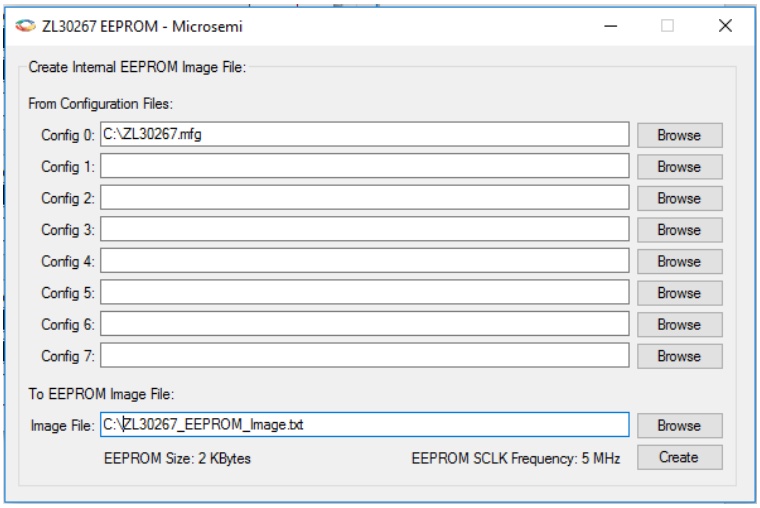
Configuration Files

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 an 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:



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.

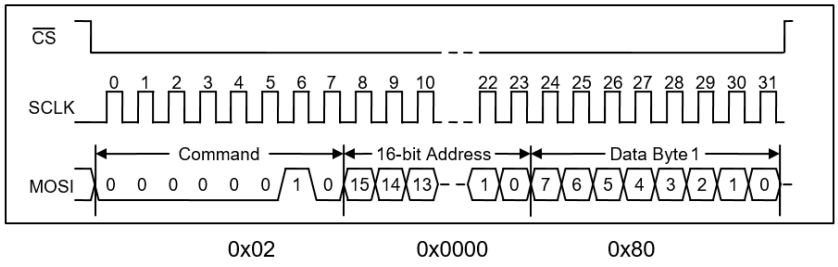
3. Procedure for loading EEPROM file

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 block and one final 20-byte block.

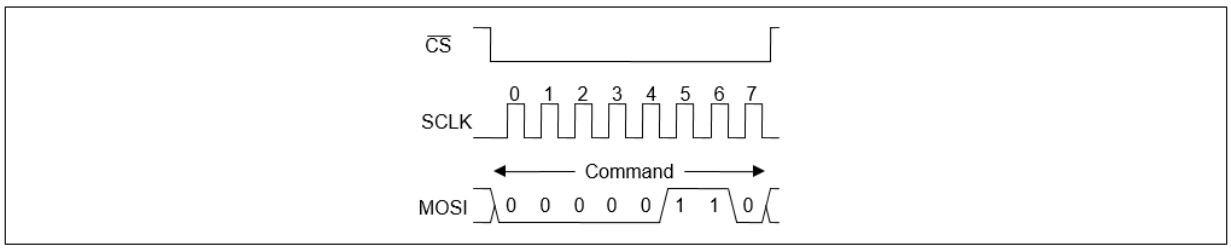
Step 2: Make the EEPROM accessible to the SPI interface

The EEPROM is mapped into the same space as the device registers. Access to the EEPROM is gained by setting register 0x0000 to 0x80. (The EEPROM can also be made available using control pins at device reset time. See section 5.2 in the ZL3026x data sheet or section 5.2 in the ZL4025x data sheet for more information). The following SPI transaction will map the EEPROM to address 0x000 of the device:



Step 3: Enable the EEPROM for Writing

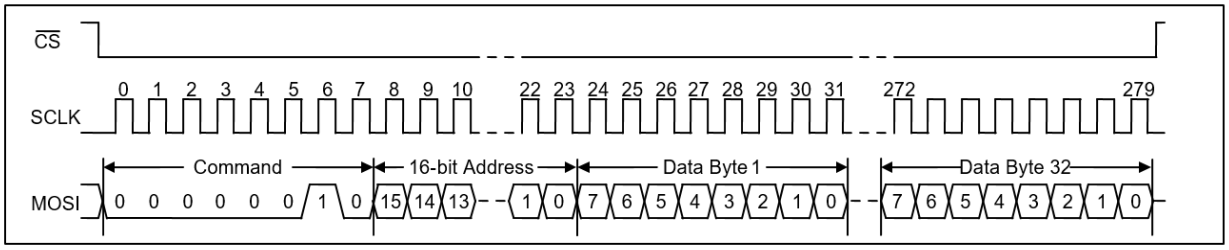
The EEPROM must be enabled for writing before *each* write transaction. This procedure uses a 32-byte burst for writing to the EEPROM. Therefore, a write enable command must be sent before each 32-byte burst. The following sequence enables the EEPROM for writing:



0x06

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.



Step 5: Delay

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

Step 6: Repeat steps 3, 4, and 5

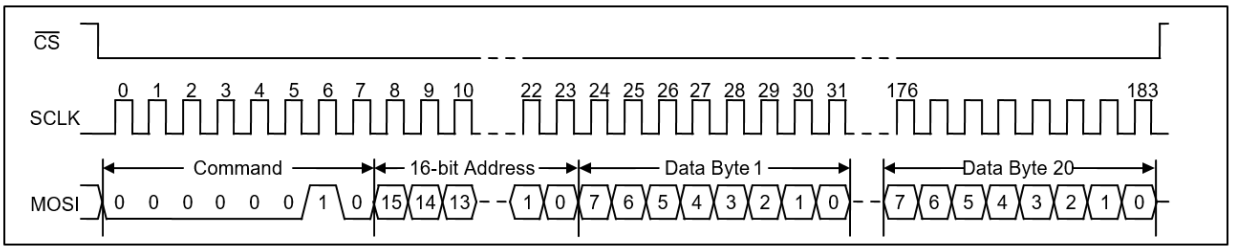
Steps 3, 4, and 5 should be repeated for each of the 32-byte blocks parsed from the EEPROM image file. On each iteration, the starting address should be incremented by 32. For example, on the second iteration, the address is 0x0020. On the third iteration, the address is 0x0040, and so on.

Step 7: Send the last block

The final block of the Image file contains only 20 bytes. This final block should be loaded the same way as the other blocks, but with a smaller burst.

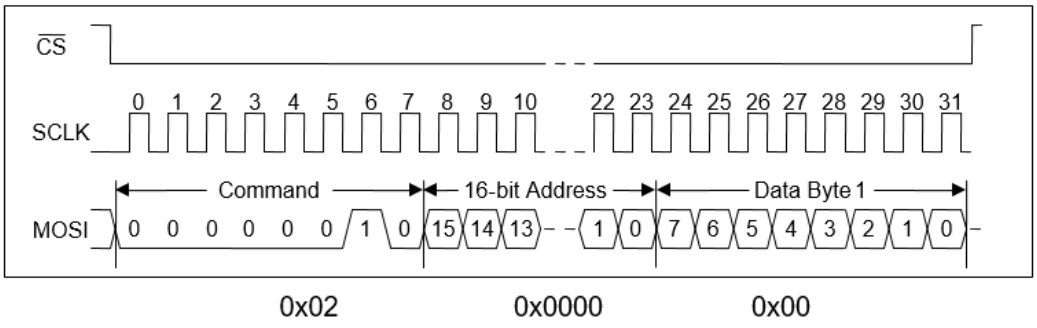
First enable the EEPROM for writing as in step 3

Then issue the final burst of 20 byte as follows. The address for this final transaction is 0x07E0.



Step 8: Unmap the EEPROM from address 0x0000

Restore access to the register map rather than the EEPROM by writing 0x0000 to 0x00, as shown below, or by resetting or power-cycling the device.



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 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**.