Application Report
SNLA320–August 2019

Understanding EEPROM Programming for DS160PR410 PCI-Express Gen-4 Redriver

Davor Glisic

ABSTRACT

EEPROM (Electrically Erasable Programmable Read-Only Memory) is frequently used to program a set of customized DS160PR410 PCI-Express Gen-4 Redriver settings that are different from the default values. Using the information provided in this application report makes redriver EEPROM configuration and programming easy to implement and understand. This application report details SMBus-to-EEPROM mapping for the DS160PR410, gives guidance about the Intel hex file format, shows several EEPROM hex file examples for typical DS160PR410 application scenarios, and provides instructions for generating hex files with the SigCon Architect tool. With a complete understanding of how to program and interpret EEPROM hex files for the DS160PR410, and with the aid of SigCon Architect, system designers are better equipped to quickly generate their own customized hex files and increase the efficiency of their designs.

Contents

1 Introduction

EEPROM is a non-volatile memory used in electronic devices to store data that must be saved when power is removed. This non-volatile memory is particularly important when an application requires different start-up configurations than the factory default settings. Upon device power-up, data saved in the EEPROM will load automatically to the device. If EEPROM is not used, interface system designs require external access to the SMBus SDA and SCL lines to set individual registers after each power-up. With EEPROM, designers eliminate the requirement for an external microprocessor or software driver to provide their desired register settings.

Programming EEPROM for the DS160PR410 PCI-Express Gen-4 redriver requires an understanding of how EEPROM relates to the DS160PR410 SMBus registers. When generating EEPROM hex images for the DS160PR410, the following topics must be considered:

- How to configure DS160PR410 for operation in SMBus Master Mode
- SMBus-to-EEPROM register mapping
- How to read EEPROM hex format
- How to calculate the CRC-8 value from a given bit stream of values
- Difference between Number of Slots and Number of Devices
- Difference between programming EEPROM data for a single device verses multiple devices

In this application report, the aforementioned topics are discussed in detail, and several EEPROM hex file examples are provided as a reference.

2 Hardware Configuration

EEPROM programming depends on the number of DS160PR410 redrivers that share the same SMBus interface. It is therefore important to understand how an EEPROM is configured to interface with one or more DS160PR410 redrivers. The following subsections provide insights on how to configure the DS160PR410 to operate in SMBus Master Mode and the EEPROM connections for single and multiple devices.

2.1 4-Level Control Pins

The DS160PR410 has six (GAIN, VOD, EQ1_ADDR1, EQ0_ADDR0, EN_SMB, and RX_DET) 4-level inputs pins that are used to control the configuration of the device. These 4-level inputs use a resistor divider to help set the four valid levels as shown in Table 1.

Table 1. DS160PR410 4-Level Control Pin Settings

PIN LEVEL	PIN SETTING
L0	1 kΩ to GND
L1	13 kΩ to GND
L2	Float
L3	59 kΩ to GND

2.2 DS160PR410 SMBus Master Mode Configuration

Configuration of the DS160PR410 for operation in SMBus Master Mode requires the following steps:

- Strap EN_SMB pin to GND with a 13-kΩ resistor. If multiple DS160PR410 devices are on the same board, it is possible to strap the EN_SMB pins of all devices with a single resistor. The resistor value in this case must be 13 kΩ divided by the number of devices. For example, if strapping the EN_SMB pins of eight devices, the value of the strapping resistor should be 13 kΩ / 8 = 1.62 kΩ.
- Configure the device SMBus slave address. There are 16 unique SMBus slave addresses that can be assigned to the device by placing external resistor straps on the EQ0_ADDR0 and EQ1_ADDR1 pins as shown in Table 2. When multiple DS160PR410 devices are on the same SMBus interface bus, each device must be configured with a unique SMBus slave address.
- Leave RX_DET floating in PCI Express applications. This enables the RX detect state machine that governs the RX detection cycle as defined in the PCI Express specification.
- VOD and GAIN pins may be left floating. The VOD and DC Gain setting are configured during the EEPROM load.
- EEPROM size of 2 kb (256 × 8-bit) such as Microchip AT24C02D is recommended.
- The external EEPROM device address byte must be 0xA0 and capable of 400-kHz operation with a 3.3-V supply.

Table 2. DS160PR410 SMBus Address Map

EQ1_ADDR1 PIN LEVEL	EQ0_ADDR0 PIN LEVEL	7-BIT ADDRESS [HEX]	8-BIT WRITE ADDRESS [HEX]
L0	L0	0x18	0x30
L0	L1	0x19	0x32
L0	L2	0x1A	0x34
L0	L3	0x1B	0x36
L1	L0	0x1C	0x38
L1	L1	0x1D	0x3A
L1	L2	0x1E	0x3C
L1	L3	0x1F	0x3E
L2	L0	0x20	0x40
L2	L1	0x21	0x42
L2	L2	0x22	0x44
L2	L3	0x23	0x46
L3	L0	0x24	0x48
L3	L1	0x25	0x4A
L3	L2	0x26	0x4C
L3	L3	0x27	0x4E

If the DS160PR410 is configured for SMBus master mode, it will remain in the SMBus IDLE state until the READ_EN_N pin is asserted to LOW. Once the READ_EN_N pin is driven LOW, the DS160PR410 becomes an SMBus master and attempts to self-configure by reading device settings stored in an external EEPROM (SMBus 8-bit address 0xA0). When the DS160PR410 has finished reading from the EEPROM successfully, it will drive the ALL_DONE_N pin LOW and change from an SMBus master to an SMBus slave.

2.3 EEPROM Configuration for Single Device

A simplified block diagram of an EEPROM device connected to a single DS160PR410 is shown in Figure 1. If a single DS160PR410 operates in SMBus Master Mode, the EEPROM loads specific SMBus register bits into the device when READ_EN_N pin is asserted low. While data is loading to the device, the device operates as a master over the bus and requests data from the EEPROM. Once the EEPROM contents are successfully read, the ALL_DONE_N pin asserts low. In most redriver EVMs, an LED is attached to the ALL_DONE_N pin to notify that a successful read has occurred. When the ALL_DONE_N pin asserts low, the device releases control of the bus and resumes operation in I2C / SMBus slave mode. At this point, an optional external I2C or SMBus control MCU master may be used for any additional programming or monitoring, though it is not required.

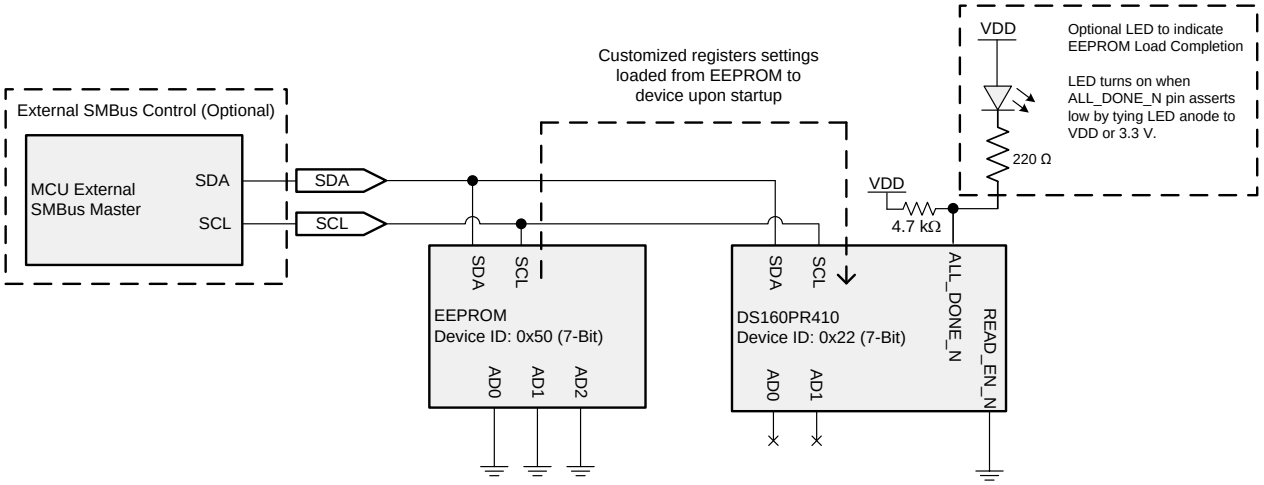

Figure 1. Example of EEPROM Used to Program a Single DS160PR410

2.4 EEPROM Configuration for Multiple Devices

The sequential behavior in which the READ_EN_N and ALL_DONE_N pins function are ideal for systemically programming EEPROM contents to multiple devices that share the same SMBus lines. By asserting the READ_EN_N pin of the first device low, the EEPROM will load the first device's contents into the first device. During this time, no other device should take control of the SMBus lines until this first device finishes and asserts its ALL_DONE_N pin low. Therefore, the ALL_DONE_N pin of the first device can be tied directly to the READ_EN_N pin of the second device in the sequence to indicate when the second device is allowed to take control of the SMBus lines. This daisy-chain process continues until the last device loads its settings from the EEPROM successfully. Daisy chaining is a recommended practice for loading EEPROM settings to multiple devices connected to the same SMBus lines, and this implementation prevents bus contention that can occur when two devices try to read from the EEPROM simultaneously.

A simplified block diagram of an EEPROM device connected to multiple DS160PR410 devices is shown in Figure 2. In this example, there are eight DS160PR410 redrivers. Note how daisy chaining is used to implement sequential EEPROM loading.

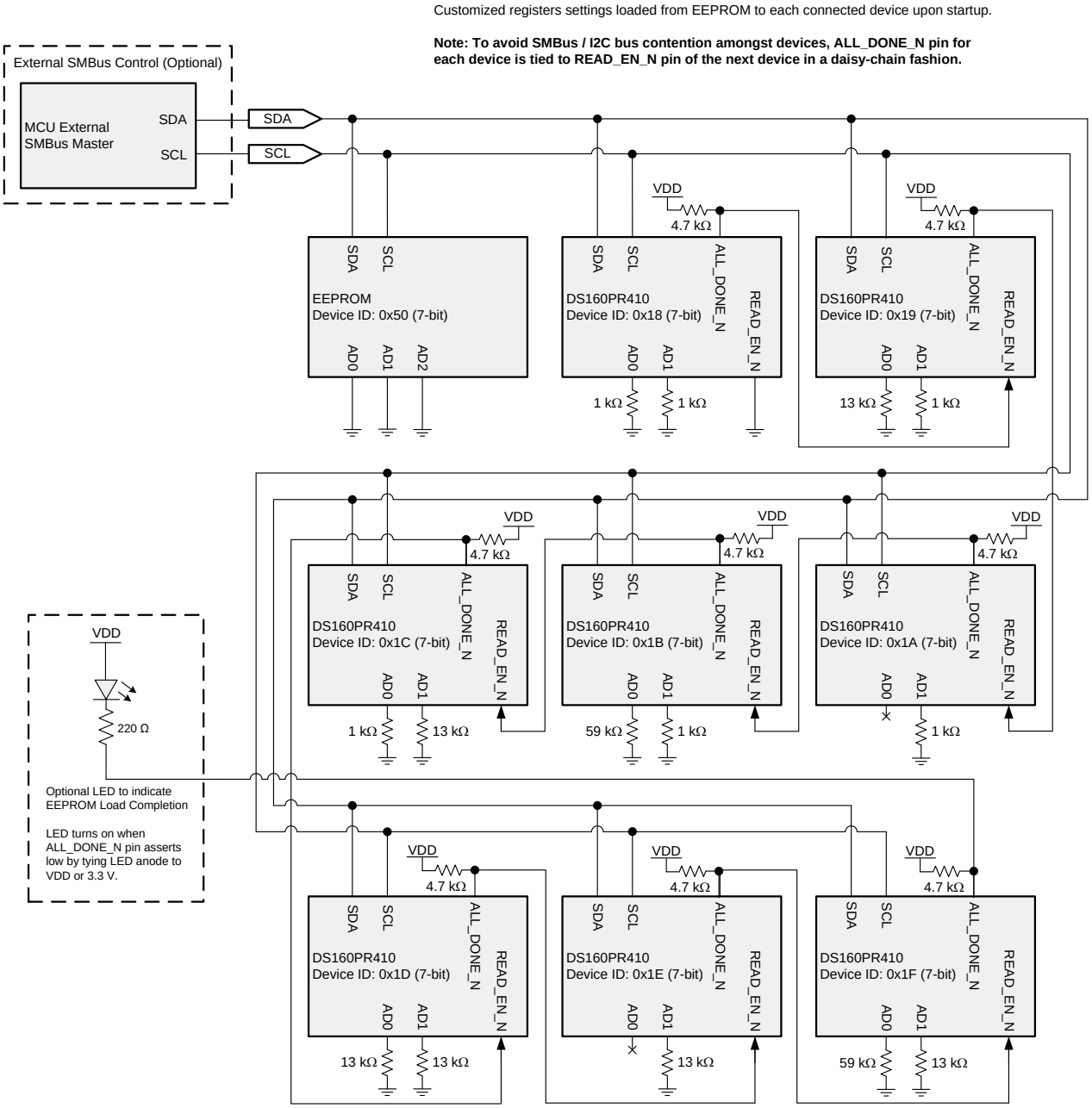

Figure 2. Example of EEPROM Used to Program Eight DS160PR410 Devices

3 SMBus-to-EEPROM Mapping

When populating EEPROM addresses, it is important to understand how the DS160PR410 SMBus Slave registers map to the EEPROM. The EEPROM only takes a subset of the SMBus register bits. SMBus register bits that are not stored in EEPROM cannot be changed from default at device start-up. A table of the DS160PR410 SMBus-to-EEPROM mapping is shown in Table 4.

To read the table, the blue column represents the EEPROM address byte, while columns 2-9 show Bits 7:0 for the corresponding EEPROM address. The matching SMBus register bit for each EEPROM address bit is shown in green, and the respective default value for that bit is shown in the row directly below. For example, EEPROM Address 0x04[6] maps to SMBus Slave Mode Reg 0x04[5], where the default value is 0, while EEPROM Address 0x04[2] maps to SMBus Slave Mode Reg 0x06[7], where the default value is 1.

NOTE: The first three bytes of the EEPROM always contain a base header to control initialization of all devices connected to the same SMBus lines.

3.1 Recommended EEPROM Device Data

Table 3 provides recommended Device Data for each available DS160PR410 CTLE Index with the VOD and DC GAIN parameters at default values.

Table 3. Recommended DS160PR410 Specific EEPROM Data as a Function of CTLE Index

CTLE INDEX	CTLE GAIN AT 4 GHz (dB)	CTLE GAIN AT 8 GHz (dB)	RECOMMENDED DEVICE SPECIFIC EEPROM DATA[1]
0	-0.3	-0.8	0x802E1018
1	0.4	1.3	0x982E1018
2	3.3	5.7	0x81261018
3	3.8	7.1	0x91261018
4	4.9	8.4	0x8A261018
5	5.2	9.1	0x92261018
6	5.4	9.8	0x9A261018
7	6.5	10.7	0x93261018
8	6.7	11.3	0x9B261018
9	7.7	12.6	0x9C261018
10	8.7	13.6	0x9D261018
11	9.1	14.4	0xA5261018
12	9.4	15.0	0xAD261018
13	10.3	15.9	0xAE261018
14	10.6	16.5	0xB6261018
15	11.8	17.8	0xBF261018

(1) The recommended device specific EEPROM data is a 4-byte wide data set. These bytes match with the descriptions of EEPROM Address 0x03-0x06 in Table 4. The eq_bst2[2:0], eq_bst1[2:0], and eq_en_bypass bit are varied to set the desired CTLE gain. All other bits are kept at their default values.

Table 4. EEPROM Address Map from DS160PR410 - Single Device with Default Values

EEPROM ADDRESS BYTE			BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
Description		0 (0x00)	CRC_EN	ADDR MAP ENABLE	EEPROM > 256 Bytes	COMMON CHANNEL	DEVICE COUNT [3]	DEVICE COUNT [2]	DEVICE COUNT [1]	DEVICE COUNT [0]
Default Value	0x00		0	0	0	0	0	0	0	0
Description		1 (0x01)	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
Default Value	0x00		0	0	0	0	0	0	0	0
Description		2 (0x02)	Max EEPROM Burst size [7]	Max EEPROM Burst size [6]	Max EEPROM Burst size [5]	Max EEPROM Burst size [4]	Max EEPROM Burst size [3]	Max EEPROM Burst size [2]	Max EEPROM Burst size [1]	Max EEPROM Burst size [0]
Default Value	0x00		0	0	0	0	0	0	0	0
Description		3 (0x03)	eq_bw[1]	eq_bw[0]	eq_bst2[2]	eq_bst2[1]	eq_bst2[0]	eq_bst1[2]	eq_bst1[1]	eq_bst1[0]
SMBus Register			0x03 [7]	0x03 [6]	0x03 [5]	0x03 [4]	0x03 [3]	0x03 [2]	0x03 [1]	0x03 [0]
Default Value	0x80		1	0	0	0	0	0	0	0
Description		4 (0x04)	eq_term_en	eq_hi_gain	eq_en_dc_off	eq_en	eq_en_bypass	drv_sel_vod[1]	drv_sel_vod[0]	drv_eq_en_override
SMBus Register			0x04 [6]	0x04 [5]	0x04 [4]	0x04 [3]	0x04 [0]	0x06 [7]	0x06 [6]	0x06 [5]
Default Value	0x26		0	0	1	0	0	1	1	0
Description		5 (0x05)	drv_en_pre	drv_en	drv_en_cm_loop	Reserved	Reserved	mr_rx_det_man	en_rx_det_count	sel_rx_det_count
SMBus Register			0x06 [4]	0x06 [3]	0x06 [2]	0x08 [3]	0x08 [2]	0x0D [6]	0x0D [5]	0x0D [4]
Default Value	0x10		0	0	0	1	0	0	0	0
Description		6 (0x06)	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
SMBus Register			0x15 [6]	0x15 [3]	0x15 [2]	0x16 [6]	0x17 [2]	0x17 [1]	0x17 [0]	n/a
Default Value	0x18		0	0	0	1	1	0	0	0

4 EEPROM Hex File Format

Intel Hex is a file format that conveys binary information in ASCII text format. It is widely used for storing and transferring data in ROM, EEPROM, and microcontrollers. As such, it has been adopted as the EEPROM format of choice for TI redrivers and mux buffers. In an Intel hex file, each line consists of hexadecimal data. Below is an example of a 256-byte .hex file.

```
:20000000570010008300830083008300830083008300830083008300830083008300830083004F
:20002000830083008700870087008700870087008700870087008700870087008700870087008700870058
:20004000870087000000000000000000000000000000000000000000000000000000000000000000000092
:20006000000000000000000000000000000000000000000000000000000000000000000000000000000080
:20008000000000812610189126101 8FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC7
:2000A000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF60
:2000C000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF40
:2000E000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF20
:00000001FF
```

Each hex file line conforms to a specific format. The following color scheme differentiates the meaning of the format for each line. Below is an example of how the first line is interpreted:

```
:20000000570010008300830083008300830083008300830083008300830083008300830083004F
```

:	Start Code: Each line always starts with an ASCII colon (:) character.
20	Byte Count: The number of data bytes (0x20) in each hex file.
0000	Load Offset: Starting load offset of data bytes.
00	Record Type: There are six record types. For redriver and mux buffer programming, this value should always be 00: 00: Data Record 01: End-of-File Record 02: Extended Segment Address Record 03: Start Segment Address Record 04: Extended Linear Address Record 05: Start Linear Address Record
57...83	Data: There are n bytes of data per line (n specified by Byte Count parameter).
4F	Checksum: Represents the checksum of the record

to evaluate the checksum of each hex line, every byte in the line is summed together, and the two's complement is taken from the sum. The checksum is the least significant byte of this result. In the hex line example above, the checksum is calculated below:

Ex: 20 + 00 + 00 + 00 + 57 + 00 + 10 + 00 + 83 + 00 + 83 + 00 + 83 + 00 + 83 + 00 + 83 + 00 + 83 + 00 + 83 + 00 + 83 + 00 + 83 + 00 + 83 + 00 +83 + 00 + 83 + 00 +83 +00 + 83 + 00 = 7B1

then do two's complement

$$1'h + not(7B1'h) = 1'h + 84E'h = 84F'h \quad (1)$$

NOTE: Application tools are often used to calculate the checksum automatically. A free online calculator is available here: 8-bit Checksum

Every line with the exception of the last line of the hex file always follows this formatting scheme. The last line of the hex file is always the End-of-File (EOF) record, shown as: 00000001FF.

5 EEPROM Device Data Fundamentals

DS160PR410 EEPROM file contains one base header. Depending on the system design, a CRC and address map header may also be used after the base header. A detailed explanation about the contents of these headers and other key fundamentals are discussed in the subsections below.

5.1 Base Header

The first three bytes define the Base Header. The meaning of the first three bytes is explained in Table 5.

Table 5. Base Header Information

BYTE	BIT NO.	BIT NAME	DESCRIPTION
0	7	CRC_EN	1 = CRC enable. If enabled, each device will have a CRC value specific to the base header (3 bytes), address map header (2 or 3 bytes, if applicable), and data (4 bytes). 0 = CRC disabled. If disabled, the CRC value is not computed, and CRC checking is ignored.
	6	ADDR Map Enable	1 = Address Map Header enable. If enabled, a 2 or 3 byte address map header will be placed after the base header to indicate the start address of each device's EEPROM. 0 = Address Map Header disable. If disabled, the first device's EEPROM information will immediately follow the base header.
	5	EEPROM > 256 Bytes	1 = Required EEPROM size is more than 256 bytes (not recommended for the DS160PR410). 0 = Required EEPROM size is 256 bytes or less (recommended for the DS160PR410).
	4	COMMON CHANNEL	1 = Common Channel Configuration enable. If enabled, the settings for all channels are referenced from one Channel Register's settings. 0 = Common Channel Configuration disable. If disabled, the settings for each channel need to be defined separately.
	3:0	DEVICE COUNT	DEVICE COUNT = (Total number of Devices) - 1 **Note**: This value is not used by the device when the EEPROM loads data, though it is a useful debugging reference.
1	7:0	RES	Reserved. Set bits to 0.
2	7:0	Max EEPROM Burst Size	Maximum number of bytes that are read during a burst read operation. A value of 0x10 is suitable for all EEPROMs using DS160PR410 redriver.

5.1.1 Common Channel Configuration

When Common Channel Configuration is enabled from the Base Header Byte 0, Bit 4, the EEPROM data for one Channel Register page is used as the universal channel settings for all Channel Registers in the device slot. The use of Common Channel Configuration also reduces the overall EEPROM size.

For example, the DS160PR410 contains 4 bytes of EEPROM data per Channel Register page. Without Common Channel Configuration, the total EEPROM size for the device slot per device is 16 bytes (4 bytes × 4 device channels). In contrast, with Common Channel Configuration, the total EEPROM size per device slot becomes only 4 bytes.

5.2 Address Map Header

When multiple devices are used, address map headers are necessary. To assign the correct EEPROM data to the correct device, each device must know the location where it can obtain the correct register settings. Details about where this information exists in the address map header are given in Table 6.

Table 6. Address Map Header Information

BYTE	BIT NO.	BIT NAME	DESCRIPTION
0	7:0	CRC Value	8-Bit CRC value for each device. CRC is computed from the base header (3 bytes), address map header (2 or 3 bytes, if applicable), and EEPROM data specific to the device (4 bytes).
1	7:0	Device EEPROM Start Address	Start address for device-specific EEPROM data. Recall that Address 0x00-0x02 of device EEPROM is stored in the base header.
2	7:3	RES	Reserved. Set bits to 0.
	2:0	Device EEPROM Start Address MSBs	These bits are only set if EEPROM Size > 256 bytes. Up to 3 MSB bits can be appended to the front of the EEPROM start address indicated in Byte 1.

NOTE: Byte 2 is present only if EEPROM > 256 bytes. The DS160PR410 only supports single byte EEPROM start addresses, however, which means the Base Header Address 0x00[5] bit must always be set to 0.

If address map headers are enabled, they occupy 128 bytes (16 devices × 4 device channels × 2 bytes) of EEPROM when EEPROM ≤ 256 bytes (Address 0x00[5] = 0'b) and 192 bytes (16 devices × 4 device channels × 3 bytes) of EEPROM when EEPROM > 256 bytes (Address 0x00[5] = 1'b). For 2-byte address map headers, the device with SMBus write address of 0x30 always occupies EEPROM addresses 0x03 and 0x04 (device channel 0), 0x05 and 0x06 (device channel 1), 0x07 and 0x08 (device channel 2), and 0x09 and 0x0A (device channel 3), a total of 8 bytes. The device with SMBus write address of 0x32 always occupies EEPROM addresses 0x0B through 0x12, and so on, as shown in Table 7. The unused address map header EEPROM addresses should be set to zeros.

With address map headers enabled, the device specific data always starts at the EEPROM address 0x83 when 2-byte address map headers are needed. Similarly, the device-specific data always starts at the EEPROM address 0xC3 when 3-byte address map headers are needed.

Table 7. Address Map Header EEPROM Address as a Function of DS160PR410 SMBus Address

7-BIT ADDRESS [HEX]	8-BIT WRITE ADDRESS [HEX]	ADDRESS MAP HEADER EEPROM ADDRESS (2-BYTE)	ADDRESS MAP HEADER EEPROM ADDRESS (3-BYTE)
0x18	0x30	0x03 - 0x0A	0x03 - 0x0E
0x19	0x32	0x0B - 0x12	0x0F - 0x1A
0x1A	0x34	0x13 - 0x1A	0x1B - 0x26
0x1B	0x36	0x1B - 0x22	0x27 - 0x32
0x1C	0x38	0x23 - 0x2A	0x33 - 0x3E
0x1D	0x3A	0x2B - 0x32	0x3F - 0x4A
0x1E	0x3C	0x33 - 0x3A	0x4B - 0x56
0x1F	0x3E	0x3B - 0x42	0x57 - 0x62
0x20	0x40	0x43 - 0x4A	0x63 - 0x6E
0x21	0x42	0x4B - 0x52	0x6F - 0x7A
0x22	0x44	0x53 - 0x5A	0x7B - 0x86
0x23	0x46	0x5B - 0x62	0x87 - 0x92
0x24	0x48	0x63 - 0x6A	0x93 - 0x9E
0x25	0x4A	0x6B - 0x72	0x9F - 0xAA
0x26	0x4C	0x73 - 0x7A	0xAB - 0xB6
0x27	0x4E	0x7B - 0x82	0xB7 - 0xC2

5.3 Cyclic Redundancy Check (CRC) Calculation

Sometimes, systems require a CRC check to ensure communication integrity between EEPROM and target device. When the CRC is enabled in the Base Header (Address 0x00[7] = 1), each device programmed by the EEPROM will have a specific CRC value in its respective Address Map Header Byte 0. The CRC is calculated through the CRC-8 polynomial, where the input x = [Base Header (3 Bytes) + Address Map header (1 or 2 Bytes) + Device Data (4 Bytes)]. An example is provided below:

Table 8. EEPROM CRC-8 Example

SECTION	VALUE (HEX)
Base Header	0xD70010
Address Map Header	0x83
Device Data	0x81261018
CRC-8 Input	0xD700108381261018
Computed CRC-8 (Address Map Header Byte 0)	0x84

NOTE: Application tools are often used to calculate the CRC automatically. A free online calculator is available here: CRC-8

5.4 Number of Devices versus Number of Slots

There is an important distinction between the number of devices and the number of slots. The number of devices pertains to the total number of physical devices present on the line. A maximum of 16 devices can be programmed from the EEPROM. However, the number of slots pertains to the total number of unique SMBus register settings to load from the EEPROM. Thus, the required size of the EEPROM depends more on the number of unique EEPROM slots that are used compared to the number of devices that will be programmed.

Oftentimes, multiple devices share the same SMBus register settings. If multiple devices share the exact same SMBus register settings, then they can share the same EEPROM slot. In contrast, if different register settings are required for any of the devices connected to the same EEPROM, each different set of SMBus register settings will require its own EEPROM slot.

6 EEPROM Hex File Examples

6.1 Example 1: EEPROM Hex File for One Device, CRC Disabled, Common Channel Configuration Enabled

The simplest case for programming EEPROM is programming for a single DS160PR410 device with all channels requiring the same settings. The following are key factors to consider when programming a single device:

- Address Map Header can typically be disabled because the EEPROM does not need to reference the start address of multiple-device EEPROM data.
- All device channels require the same settings, therefore only one 4-byte slot (device data) is needed.
- The device data should be derived from Table 4. Recommended device settings (device data) are also given in Table 3.
- EEPROM size ≤ 256 bytes is more than adequate for a single DS160PR410.
- Device SMBus slave address must be 0x30 (8-bit)

An example of a hex file for a single DS160PR410 device with all channels requiring the same settings is shown in Figure 3. The data relevant to the DS160PR410 EEPROM address bits is highlighted in green. Note that this example hex file only works if the device SMBus slave address is 0x30 (8-bit).

```
:20000000100010 81261018 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF2E
:20002000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE0
:20004000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC0
:20006000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFA0
:20008000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF80
:2000A000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF60
:2000C000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF40
:2000E000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF20
:00000001FF
```

__ = EEPROM Base Header
__ = Device Data

Figure 3. Example of a Hex File for a Single DS160PR410 Device, Common Channel Configuration Enabled

From the DS160PR410 hex file, the base header bytes are 0x100010. From Table 5, this means the following:

- CRC is disabled (Address 0x00[7] = 0'b).
- No address map header is used (Address 0x00[6] = 0'b).
- EEPROM ≤ 256 bytes (Address 0x00[5] = 0'b).
- Common Channel Configuration is enabled (Address 0x00[4] = 1'b).
- DEVICE COUNT = 1 Device (Address 0x00[3:0] = 0000'b).
- Max EEPROM Burst size = 16 bytes (Address 0x02 = 0x10).

No address map header is used, therefore the remaining 4 bytes following the base header in the green-highlighted section are device-specific data. In the SMBus-to-EEPROM mapping table, these bytes match with the descriptions of the EEPROM Addresses 0x03-0x06. For this example, the 4-byte, device-specific data configures all channels of the device with EQ Index = 2, VOD = 0 dB, and DC GAIN = 0 dB.

6.2 Example 2: EEPROM Hex File for One Device, CRC Disabled, Common Channel Configuration Disabled

When programming EEPROM for a single device with each channel requiring unique settings, the following are considered:

- Address Map Header can typically be disabled because the EEPROM does not need to reference the start address of multiple-device EEPROM data.
- EEPROM size ≤ 256 bytes is more than adequate for a single DS160PR410.
- Each device channel requires unique settings, therefore one 16-byte slot is needed.
- The device data for each channel should be derived from Table 4. Recommended device settings (device data) are also given in Table 3.
- Device SMBus slave address must be 0x30 (8-bit).

An example of a hex file for a single DS160PR410 device with each channel requiring unique settings is shown in Figure 4. Note that this example hex file only works if the device SMBus slave address is 0x30 (8-bit).

```
:20000000000010812610189126101888A26101892261018FFFFFFFFFFFFFFFFFFFFFFFFFF77
:20002000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE0
:20004000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC0
:20006000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFA0
:20008000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF80
:2000A000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF60
:2000C000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF40
:2000E000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF20
:00000001FF
```

= EEPROM Base Header
= Channel 0 Device Data
= Channel 1 Device Data
= Channel 2 Device Data
= Channel 3 Device Data

Figure 4. Example of a Hex File for a Single DS160PR410 Device, Common Channel Configuration Disabled

The base header bytes are 0x000010. From Table 5, the following is derived:

- CRC is disabled (Address 0x00[7] = 0'b).
- No address map header is used (Address 0x00[6] = 0'b).
- EEPROM ≤ 256 bytes (Address 0x00[5] = 0'b).
- Common Channel Configuration is disabled (Address 0x00[4] = 0'b).
- DEVICE COUNT = 1 Device (Address 0x00[3:0] = 0000'b).
- Max EEPROM Burst size = 16 bytes (Address 0x02 = 0x10).

No address map header is used, therefore the remaining 16 bytes following the base header are device-specific data. Channel 0 device data is in the green-highlighted 4-byte section, followed by Channel 1 device data in blue-, Channel 2 device data in yellow-, and Channel 3 device data in magenta-highlighted, 4-byte sections. In the SMBus-to-EEPROM mapping table, these 4-byte sections match with the descriptions of the EEPROM Addresses 0x03-0x06.

In this example, device data configures the Channel 0 with EQ Index = 2, Channel 1 with EQ Index = 3, Channel 2 with EQ Index = 4, and Channel 3 with EQ Index = 5. All channels are configured with VOD = 0 dB and DC GAIN = 0 dB.

6.3 Example 3: EEPROM Hex File for Eight Devices, Two Slots, CRC Disabled, Common Channel Configuration Enabled

When programming EEPROM for multiple devices, the following are considered:

- Address Map Header must be used.
- All device channels require the same settings, therefore only 4-byte slots are needed.
- The device data for each device should be derived from Table 4. Recommended device settings (device data) are also given in Table 3.
- EEPROM size ≤ 256 bytes is more than adequate for two 4-byte slots.

An example of a hex file for eight DS160PR410 devices requiring only two unique EEPROM slots is shown in Figure 5.

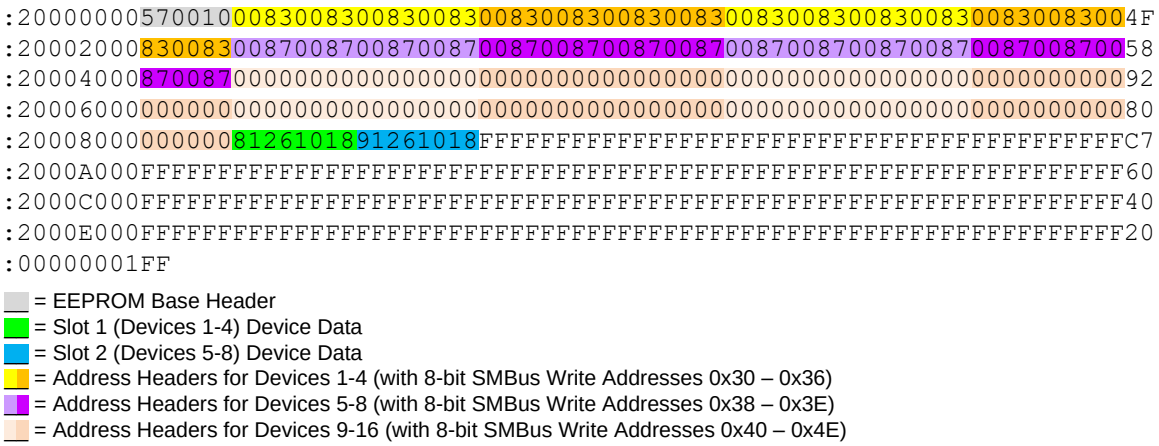

Figure 5. Example of a Hex File for 8 DS160PR410 Devices (2 Unique EEPROM Slots)

The base header bytes are 0x570010. From Table 5, the following is derived:

- CRC is disabled (Address 0x00[7] = 0’b).
- Address map header is enabled (Address 0x00[6] = 1’b).
- EEPROM ≤ 256 bytes (Address 0x00[5] = 0’b).
- Common Channel Configuration is enabled (Address 0x00[4] = 1’b).
- DEVICE COUNT = 8 Devices (Address 0x00[3:0] = 0111’b).
- Max EEPROM Burst size = 16 bytes (Address 0x02 = 0x10).

With the CRC disabled, each Address Map Header Byte 0 is 0x00. In this example, Devices 1-4 share one address map header (0x0083008300830083) while Devices 5-8 share another address header (0x0087008700870087). This occurs if multiple devices are programmed with identical SMBus register settings. A summary of address map headers, device data, and device settings for this hex file example is give in Table 9.

The EEPROM addresses 0x43 through 0x82 are not used and should be set to 0x00 values.

Table 9. Summary of Address Map Headers and Device Data

DEVICE	ADDRESS MAP HEADER	ADDRESS MAP HEADER LOCATION	DEVICE DATA (DEVICE SETTING)	DEVICE DATA LOCATION
1	0x0083008300830083	0x03 - 0x0A	0x81261018 (VOD = 0 dB, GAIN = 0 dB, EQ Index = 2)	0x83 - 0x86
2		0x0B - 0x12		
3		0x13 - 0x1A		
4		0x1B - 0x22		
5	0x0087008700870087	0x23 - 0x2A	0x91261018 (VOD = 0 dB, GAIN = 0 dB, EQ Index = 3)	0x87 - 0x8A
6		0x2B - 0x32		
7		0x33 - 0x3A		
8		0x3B - 0x42		

6.4 Example 4: EEPROM Hex File for Eight Devices, Four Slots, CRC Enabled, Common Channel Configuration Enabled

When programming EEPROM for multiple devices, the following are considered:

- Address Map Header must be used.
- All device channels require the same settings, therefore only 4-byte slots are needed.
- The device data for each device should be derived from Table 4. Recommended device settings (device data) are also given in Table 3.
- EEPROM size ≤ 256 bytes adequate for two 4-byte slots.

An example of a hex file for eight DS160PR410 devices requiring four unique EEPROM slots is shown in Figure 6.

```
:20000000D70010848384838483848336C876C876C876C876C876C876C876C87178B178B17EA
:200020008B178B178B178B178B178B4F8F4F8F4F8F4F8F6C876C876C876C878483848384835
:20004000838483000000000000000000000000000000000000000000000000000000000016
:20006000000000000000000000000000000000000000000000000000000000000000000080
:20008000000000812610189126101888A26101892261018FFFFFFFFFFFFFFFFFFFFFFFFF07
:2000A000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF60
:2000C000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF40
:2000E000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF20
:00000001FF
```

= EEPROM Base Header
= Slot 1 (Devices 1 and 8) Device Data
= Slot 2 (Devices 2, 3 and 7) Device Data
= Slot 3 (Devices 4 and 5) Device Data
= Slot 4 (Device 6) Device Data
= Address Headers for Devices 1-4 (with 8-bit SMBus Write Addresses 0x30 – 0x36)
= Address Headers for Devices 5-8 (with 8-bit SMBus Write Addresses 0x38 – 0x3E)
= Address Headers for Devices 9-16 (with 8-bit SMBus Write Addresses 0x40 – 0x4E)

Figure 6. Example of a Hex File for 8 DS160PR410 Devices (2 Unique EEPROM Slots)

The base header bytes are 0xD70010. From Table 5, the following is derived:

- CRC is enabled (Address 0x00[7] = 1'b).
- Address map header is enabled (Address 0x00[6] = 1'b).
- EEPROM ≤ 256 bytes (Address 0x00[5] = 0'b).
- Common Channel Configuration is enabled (Address 0x00[4] = 1'b).
- DEVICE COUNT = 8 Devices (Address 0x00[3:0] = 0111'b).
- Max EEPROM Burst size = 16 bytes (Address 0x02 = 0x10).

In this example, Devices 1 and 8 share the address map header 0x8483848384838483, as they are programmed with identical device data. Similarly, Devices 2, 3, and 7 share the address map header 0x6C876C876C876C87, Devices 4 and 5 share the address map header 0x178B178B178B178B, and Device 6 has the address map header 0x4F8F4F8F4F8F4F8F. A summary of address map headers, device data, and device settings for this hex file example is give in Table 10.

The EEPROM addresses 0x43 through 0x82 are not used and should be set to 0x00 values.

Table 10. Summary of Address Map Headers and Device Data

DEVICE	ADDRESS MAP HEADER	ADDRESS MAP HEADER LOCATION	DEVICE DATA (DEVICE SETTING)	DEVICE DATA LOCATION
1	0x8483848384838483	0x03 - 0x0A	0x81261018 (VOD = 0 dB, GAIN = 0 dB, EQ Index = 2)	0x83 - 0x86
2	0x6C876C876C876C87	0x0B - 0x12	0x91261018 (VOD = 0 dB, GAIN = 0 dB, EQ Index = 3)	0x87 - 0x8A
3		0x13 - 0x1A		
4	0x178B178B178B178B	0x1B - 0x22	0x8A261018 (VOD = 0 dB, GAIN = 0 dB, EQ Index = 4)	0x8B - 0x8E
5		0x23 - 0x2A		
6	0x4F8F4F8F4F8F4F8F	0x2B - 0x32	0x92261018 (VOD = 0 dB, GAIN = 0 dB, EQ Index = 5)	0x8F - 0x92
7	0x6C876C876C876C87	0x33 - 0x3A	0x91261018 (VOD = 0 dB, GAIN = 0 dB, EQ Index = 3)	0x87 - 0x8A
8	0x8483848384838483	0x3B - 0x42	0x81261018 (VOD = 0 dB, GAIN = 0 dB, EQ Index = 2)	0x83 - 0x86

7 Using SigCon Architect Tool for Generating EEPROM Hex Files

The DS160PR410 profile for the SigCon Architect can help with generating custom EEPROM hex files. This section outlines the necessary steps to generate the Example 4 hex file.

1. Install the SigCon Architect Version 3.0.0.8 application and the compatible SigCon Architect profiles containing the DS160PR410 profile.
2. Start the SigCon Architect application.
3. Click "Continue in Demo Mode" if an actual evaluation module (EVM) is not connected to a PC through a USB2ANY interface.
4. Select the DS160PR410 Configuration Page.
5. Click "Apply" in the USB2ANY Details section of the Configuration Page.
6. Select the DS160PR410 EEPROM Page.
7. In the Base Header Details section of the EEPROM Page, complete the following:
    1. Set No. of Devices to 8.
    2. Select 256 Bytes for EEPROM Size.
    3. Check Common Channel, Address Map, and CRC boxes.
8. In the Slot Update Details section of the EEPROM Page, complete the following:
    1. Set No. of Slots to 4.
    2. Select Slot Number 1 and update Slot data to 81 26 10 18 (use Table 3 as a reference). Check if Major Channel Settings for the slot are as desired.
    3. Select Slot Number 2 and update Slot data to 91 26 10 18. Check if Major Channel Settings for the slot are as desired.
    4. Select Slot Number 3 and update Slot data to 8A 26 10 18. Check if Major Channel Settings for the slot are as desired.

5. Select Slot Number 4 and update Slot data to 92 26 10 18. Check if Major Channel Settings for the slot are as desired.
6. Assign Slot# to each Device Address in the Address/Slot list Selection:
    1. Set Slot#1 to Devices Addresses 0x30 and 0x3E.
    2. Set Slot#2 to Devices Addresses 0x32, 0x34, and 0x3C.
    3. Set Slot#3 to Devices Addresses 0x36 and 0x38.
    4. Set Slot#4 to Devices Address 0x3A.
9. Review EEPROM Data Table.
10. Click on "Write to EEPROM Hex" to save / generate a hex file.

A complete DS160PR410 EEPROM Page is shown in Figure 7.

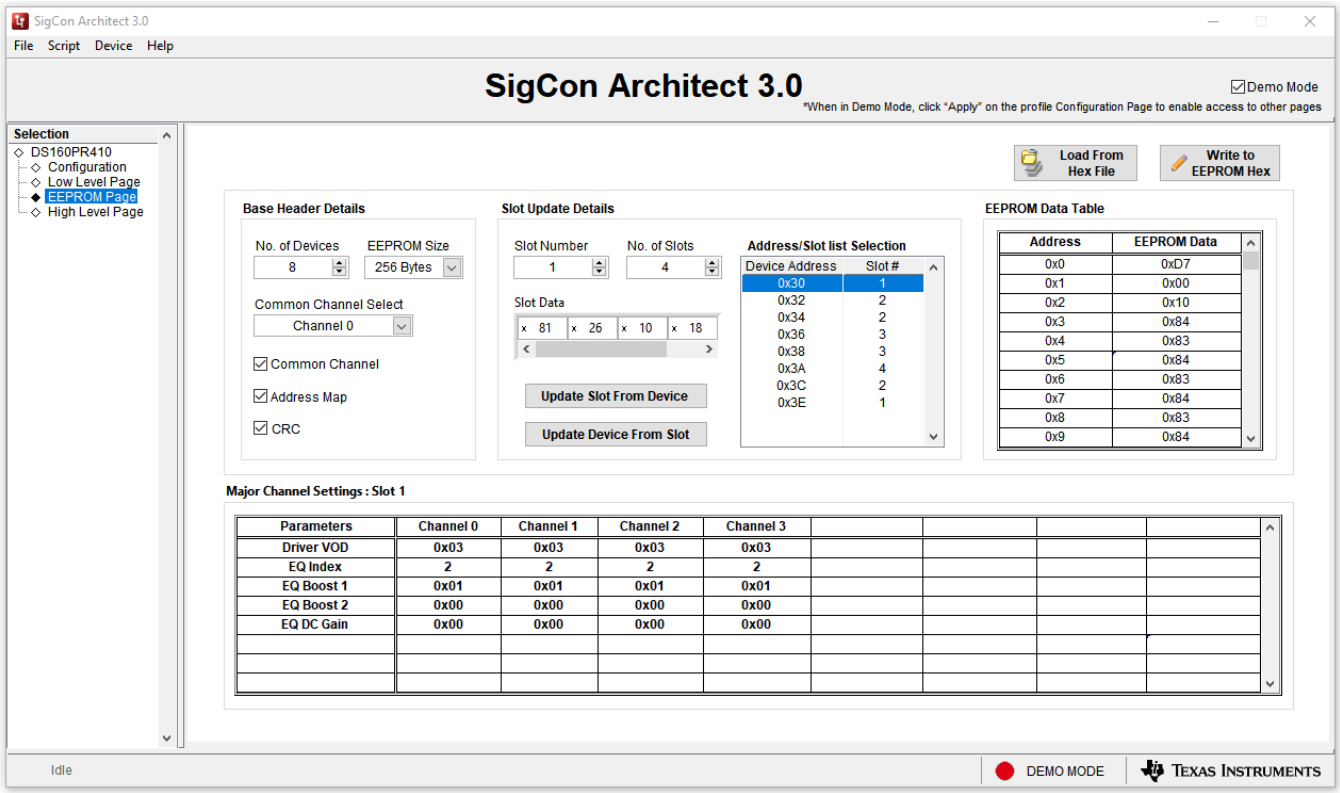

Figure 7. SigCon Architect DS160PR410 EEPROM Page

8 Conclusion

In this application note, the benefits of EEPROM are explored as they relate to DS160PR410. Device-specific EEPROM concepts such as the Base Header, Address Map Header, CRC, and EEPROM data slot are explained in detail. In addition, the requirements of the Intel hex format are revealed to help users differentiate between EEPROM sections relevant to formatting and EEPROM sections relevant to the device settings. Step-by-step instruction for generating hex files using SigCon Architect as provided as well. With a complete understanding of how to program and interpret these EEPROM hex files and the aid of SigCon Architect, system designers are better equipped to quickly generate their own customized hex files and increase the efficiency of their DS160PR410 designs.

9 References

1. Texas Instruments, *DS160PR410 4-Channel PCI-Express Gen-4 Linear Redriver* data sheet (SNLS645)
2. Texas Instruments, *DS160PR410 Programming Guide* (SNLU255)
3. Texas Instruments, *Understanding EEPROM Programming for High Speed Redrivers and Mux Buffers* application report (SNLA228)

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265