Application Report
SNLA228–October 2014

Understanding EEPROM Programming for High Speed Repeaters and Mux Buffers

Michael Lu, Prescott Siao

ABSTRACT

System designers often use EEPROM (Electrically Erasable Programmable Read-Only Memory) to program a set of customized high speed repeater and mux buffer start-up settings that are different from the default. Using the information here will make repeater EEPROM configuration and programming easy to implement and understand. This application note addresses SMBus-to-EEPROM mapping for 2-channel repeaters, 8-channel repeaters (8-channel uni-directional and 4-lane bi-directional), and 2:1/1:2 mux buffers. In addition, this application note provides guidance and several examples regarding how to read the Intel hex file format as it relates to each programmed TI device. With a complete understanding of how to program and interpret EEPROM hex files for TI's 2-channel repeaters, 8-channel repeaters, and 2:1/1:2 mux buffers, system designers are better equipped to generate their own customized hex files and increase the efficiency of their final designs. The information in this Application Report applies to the DS80PCIxxx, DS100BRxxx, and DS125BRxxx drivers as well as the DS100MB203 and DS125MB203 mux buffers.

Contents

1 Introduction

EEPROM is 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 in order 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 TI's high speed repeaters requires an understanding of how EEPROM relates to the high speed repeater Slave Mode SMBus registers. When generating EEPROM hex images for one of TI's high speed repeaters, the following must be considered:

- Users must map EEPROM address bits correctly to the matching device SMBus register bits. Note that only a subset of SMBus register settings are mapped to EEPROM.
- The contents of an EEPROM are typically stored in Intel hex-file format. The format at times can appear cryptic, especially when multiple devices are programmed to the EEPROM.
- Programming multiple device slots requires either two or three additional address map header bytes per device to denote the CRC and starting address of each device slot.

To address these design challenges, this application note explains how to map SMBus registers to EEPROM addresses, how to interpret the Intel hex file format, and how to program EEPROM data for multiple devices.

2 EEPROM Physical Configuration

EEPROM programming depends on the number of repeaters that share the same SMBus interface. It is therefore important to understand how an EEPROM is configured to interface with TI's repeaters and mux buffers. The following subsections provide insight about EEPROM connections for single and multiple devices.

2.1 EEPROM Configuration for Single Device

A simplified block diagram of EEPROM connected to a single device is shown in Figure 1. If a single device operates in SMBus Master Mode, the EEPROM loads specific SMBus register bits into the device when $\overline{\text{READ\_EN}}$ 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 $\overline{\text{ALL\_DONE}}$ pin asserts low. In most repeater and mux buffer EVMs, an LED is attached to the $\overline{\text{ALL\_DONE}}$ pin to notify that a successful read has occurred. Once the $\overline{\text{ALL\_DONE}}$ pin asserts low, the device releases control of the bus and resumes operation in SMBus slave mode. At this point, an optional external SMBus control MCU master may be used for any additional programming or monitoring, though it is not required.

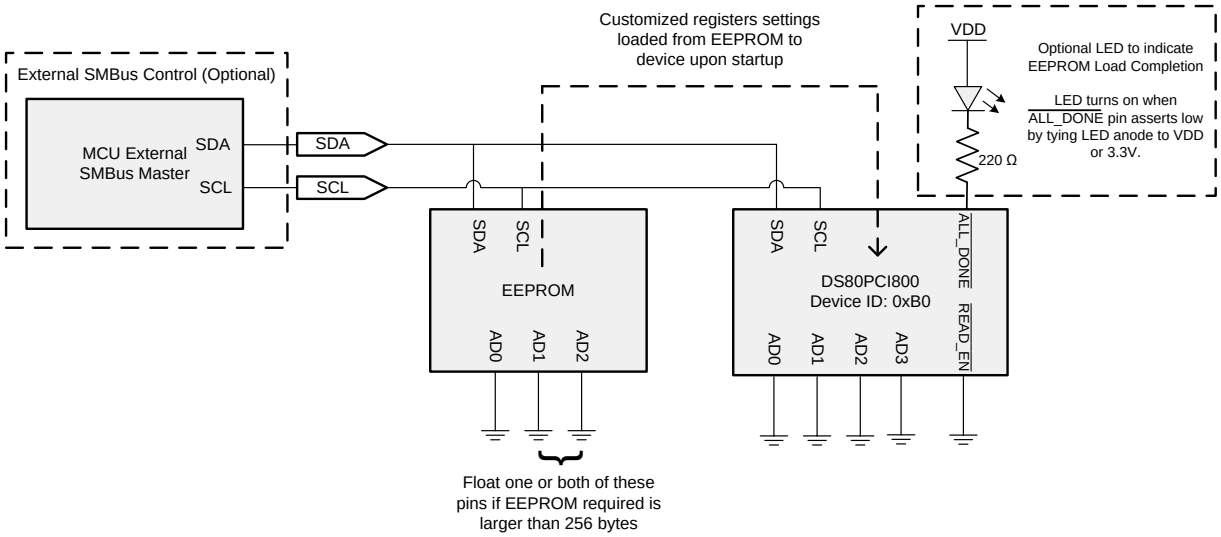

Figure 1. Example of EEPROM Used to Program a Single DS80PCI800

2.2 EEPROM Configuration for Multiple Devices

The sequential behavior in which the $\overline{\text{READ\_EN}}$ and $\overline{\text{ALL\_DONE}}$ pins function are ideal for systemically programming EEPROM contents to multiple devices that share the same SMBus lines. By asserting the $\overline{\text{READ\_EN}}$ 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 can take control of the SMBus lines until this first device finishes and asserts its $\overline{\text{ALL\_DONE}}$ pin low. Therefore, the $\overline{\text{ALL\_DONE}}$ pin of the first device can be tied directly to the $\overline{\text{READ\_EN}}$ pin of the second device in the sequence to indicate when the second device can 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 EEPROM connected to multiple devices is shown in Figure 2. In this example, there are 5 x DS80PCI800 repeaters. Note how daisy chaining is used to implement sequential EEPROM loading.

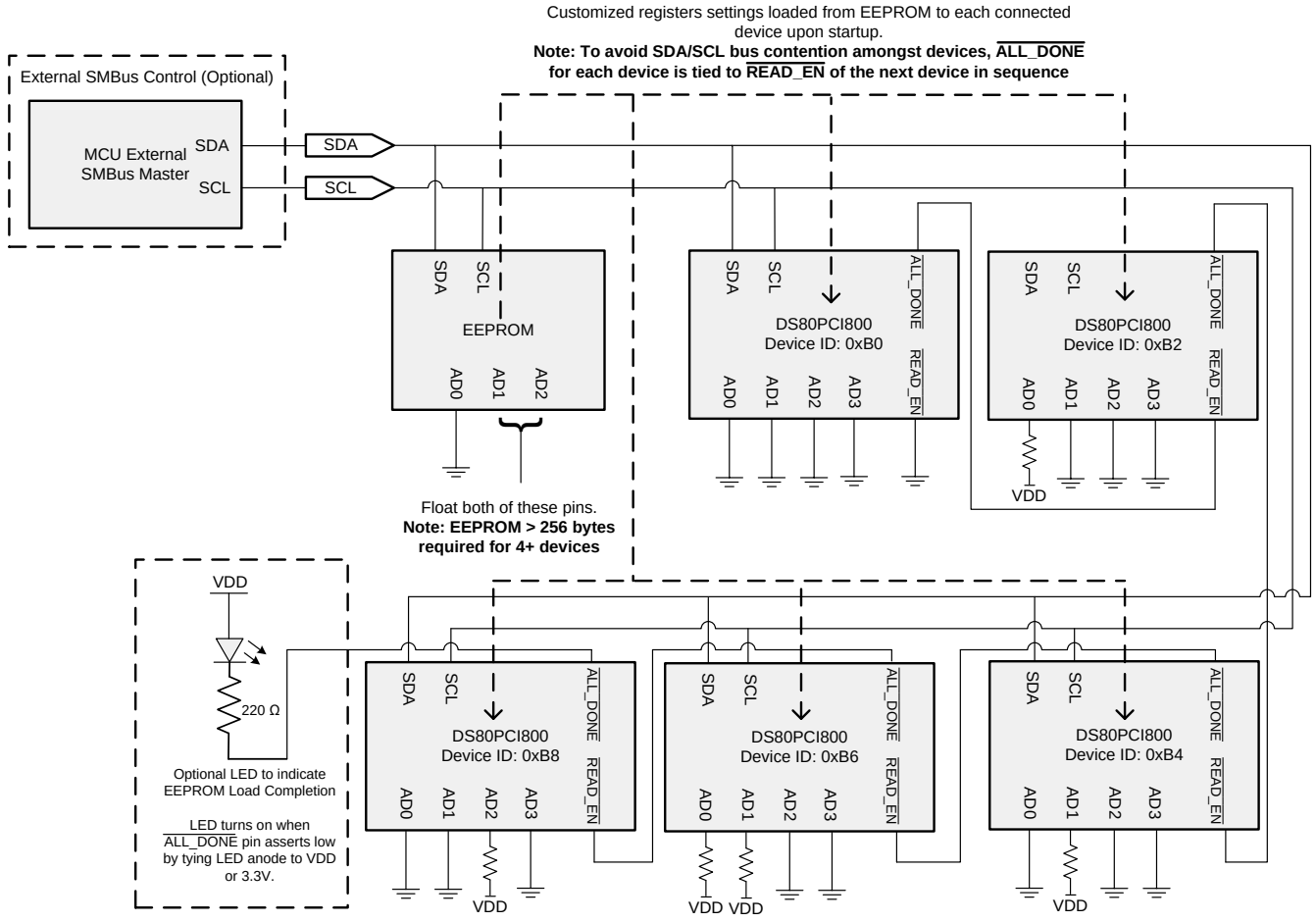

Figure 2. Example of EEPROM Used to Program Five DS80PCI800 Devices

3 SMBus-to-EEPROM Mapping

When populating EEPROM addresses, it is important to understand how the device 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 startup. A table of the DS125BR401A SMBus-to-EEPROM mapping is shown in Table 1.

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 0x05[4] maps to SMBus Slave Mode Reg 0x04[1], where the default value is 0, while EEPROM Address 0x06[2] maps to SMBus Slave Mode Reg 0x0B[6], where the default value is 1.

Though TI's 2-channel repeaters, 8-channel repeaters, and 2:1/1:2 mux buffers differ from one another regarding the function description of each specific SMBus register bit, they all share the same SMBus-to-EEPROM register bit-to-bit mapping.

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.

Table 1. EEPROM Address Map from DS125BR401A - Single Device with Default Value

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 Present	EEPROM > 256 Bytes	Reserved	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	Max EEPROM	Max EEPROM	Max EEPROM	Max EEPROM	Max EEPROM	Max EEPROM	Max EEPROM
			Burst size [7]	Burst size [6]	Burst size [5]	Burst size [4]	Burst size [3]	Burst size [2]	Burst size [1]	Burst size [0]
Default Value	0x00		0	0	0	0	0	0	0	0
Description		3 (0x03)	PWDN_CH7	PWDN_CH6	PWDN_CH5	PWDN_CH4	PWDN_CH3`	PWDN_CH2	PWDN_CH1	PWDN_CH0
SMBus Register			0x01 [7]	0x01 [6]	0x01 [5]	0x01 [4]	0x01 [3]	0x01 [2]	0x01 [1]	0x01 [0]
Default Value	0x00		0	0	0	0	0	0	0	0
Description		4 (0x04)	Reserved	Reserved	Reserved	Reserved	OVRD_PWDN	CH7_EQ_LIM	CH6_EQ_LIM	CH5_EQ_LIM
SMBus Register			0x02 [5]	0x02 [4]	0x02 [3]	0x02 [2]	0x02 [0]	0x04 [7]	0x04 [6]	0x04 [5]
Default Value	0x00		0	0	0	0	0	0	0	0
Description		5 (0x05)	CH4_EQ_LIM	CH3_EQ_LIM	CH2_EQ_LIM	CH1_EQ_LIM	CH0_EQ_LIM	Reserved	OVRD_SD_TH	Reserved
SMBus Register			0x04 [4]	0x04 [3]	0x04 [2]	0x04 [1]	0x04 [0]	0x06 [4]	0x08 [6]	0x08 [5]
Default Value	0x04		0	0	0	0	0	1	0	0
Description		6 (0x06)	OVRD_IDLE	OVRD_RX_DET	OVRD_MODE_B	Reserved	Reserved	Reserved	Reserved	Reserved
SMBus Register			0x08 [4]	0x08 [3]	0x08 [2]	0x08 [1]	0x08 [0]	0x0B [6]	0x0B [5]	0x0B [4]
Default Value	0x07		0	0	0	0	0	1	1	1
Description		7 (0x07)	Reserved	Reserved	Reserved	Reserved	CH0_IDLE_AUTO	CH0_IDLE_SEL	CH0_RXDET_1	CH0_RXDET_0
SMBus Register			0x0B [3]	0x0B [2]	0x0B [1]	0x0B [0]	0x0E [5]	0x0E [4]	0x0E [3]	0x0E [2]
Default Value	0x00		0	0	0	0	0	0	0	0
Description		8 (0x08)	CH0_EQ_7	CH0_EQ_6	CH0_EQ_5	CH0_EQ_4	CH0_EQ_3	CH0_EQ_2	CH0_EQ_1	CH0_EQ_0
SMBus Register			0x0F [7]	0x0F [6]	0x0F [5]	0x0F [4]	0x0F [3]	0x0F [2]	0x0F [1]	0x0F [0]
Default Value	0x2F		0	0	1	0	1	1	1	1
Description		9 (0x09)	CH0_SCP	CH0_MODE	Reserved	Reserved	Reserved	CH0_VOD_2	CH0_VOD_1	CH0_VOD_0
SMBus Register			0x10 [7]	0x10 [6]	0x10 [5]	0x10 [4]	0x10 [3]	0x10 [2]	0x10 [1]	0x10 [0]
Default Value	0xAD		1	0	1	0	1	1	0	1
Description		10 (0x0A)	CH0_DEM_2	CH0_DEM_1	CH0_DEM_0	Reserved	CH0_IDLE_THA_1	CH0_IDLE_THA_0	CH0_IDLE_THD_1	CH0_IDLE_THD_0
SMBus Register			0x11 [2]	0x11 [1]	0x11 [0]	0x12 [7]	0x12 [3]	0x12 [2]	0x12 [1]	0x12 [0]
Default Value	0x40		0	1	0	0	0	0	0	0
Description		11 (0x0B)	CH1_IDLE_AUTO	CH1_IDLE_SEL	CH1_RXDET_1	CH1_RXDET_0	CH1_EQ_7	CH1_EQ_6	CH1_EQ_5	CH1_EQ_4
SMBus Register			0x15 [5]	0x15 [4]	0x15 [3]	0x15 [2]	0x16 [7]	0x16 [6]	0x16 [5]	0x16 [4]
Default Value	0x02		0	0	0	0	0	0	1	0
Description		12 (0x0C)	CH1_EQ_3	CH1_EQ_2	CH1_EQ_1	CH1_EQ_0	CH1_SCP	CH1_MODE	Reserved	Reserved
SMBus Register			0x16 [3]	0x16 [2]	0x16 [1]	0x16 [0]	0x17 [7]	0x17 [6]	0x17[5]	0x17[4]
Default Value	0xFA		1	1	1	1	1	0	1	0
Description		13 (0x0D)	Reserved	CH1_VOD_2	CH1_VOD_1	CH1_VOD_0	CH1_DEM_2	CH1_DEM_1	CH1_DEM_0	Reserved
SMBus Register			0x17 [3]	0x17 [2]	0x17 [1]	0x17 [0]	0x18 [2]	0x18 [1]	0x18 [0]	0x19 [7]
Default Value	0xD4		1	1	0	1	0	1	0	0

Table 1. EEPROM Address Map from DS125BR401A - Single Device with Default Value (continued)

EEPROM Address Byte			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Description		14 (0x0E)	CH1_IDLE_THA_1	CH1_IDLE_THA_0	CH1_IDLE_THD_1	CH1_IDLE_THD_0	CH2_IDLE_AUTO	CH2_IDLE_SEL	Reserved	Reserved
SMBus Register			0x19 [3]	0x19 [2]	0x19 [1]	0x19 [0]	0x1C [5]	0x1C [4]	0x1C [3]	0x1C [2]
Default Value	0x00		0	0	0	0	0	0	0	0
Description		15 (0x0F)	CH2_EQ_7	CH2_EQ_6	CH2_EQ_5	CH2_EQ_4	CH2_EQ_3	CH2_EQ_2	CH2_EQ_1	CH2_EQ_0
SMBus Register			0x1D [7]	0x1D [6]	0x1D [5]	0x1D [4]	0x1D [3]	0x1D [2]	0x1D [1]	0x1D [0]
Default Value	0x2F		0	0	1	0	1	1	1	1
Description		16 (0x10)	CH2_SCP	CH2_MODE	Reserved	Reserved	Reserved	CH2_VOD_2	CH2_VOD_1	CH2_VOD_0
SMBus Register			0x1E [7]	0x1E [6]	0x1E [5]	0x1E [4]	0x1E [3]	0x1E [2]	0x1E [1]	0x1E [0]
Default Value	0xAD		1	0	1	0	1	1	0	1
Description		17 (0x11)	CH2_DEM_2	CH2_DEM_1	CH2_DEM_0	Reserved	CH2_IDLE_THA_1	CH2_IDLE_THA_0	CH2_IDLE_THD_1	CH2_IDLE_THD_0
SMBus Register			0x1F [2]	0x1F [1]	0x1F [0]	0x20 [7]	0x20 [3]	0x20 [2]	0x20 [1]	0x20 [0]
Default Value	0x40		0	1	0	0	0	0	0	0
Description		18 (0x12)	CH3_IDLE_AUTO	CH3_IDLE_SEL	CH3_RXDET_1	CH3_RXDET_0	CH3_EQ_7	CH3_EQ_6	CH3_EQ_5	CH3_EQ_4
SMBus Register			0x23 [5]	0x23 [4]	0x23 [3]	0x23 [2]	0x24 [7]	0x24 [6]	0x24 [5]	0x24 [4]
Default Value	0x02		0	0	0	0	0	0	1	0
Description		19 (0x13)	CH3_EQ_3	CH3_EQ_2	CH3_EQ_1	CH3_EQ_0	CH3_SCP	CH3_MODE	Reserved	Reserved
SMBus Register			0x24 [3]	0x24 [2]	0x24 [1]	0x24 [0]	0x25 [7]	0x25 [6]	0x25 [5]	0x25 [4]
Default Value	0xFA		1	1	1	1	1	0	1	0
Description		20 (0x14)	Reserved	CH3_VOD_2	CH3_VOD_1	CH3_VOD_0	CH3_DEM_2	CH3_DEM_1	CH3_DEM_0	Reserved
SMBus Register			0x25 [3]	0x25 [2]	0x25 [1]	0x25 [0]	0x26 [2]	0x26 [1]	0x26 [0]	0x27 [7]
Default Value	0xD4		1	1	0	1	0	1	0	0
Description		21 (0x15)	CH3_IDLE_THA_1	CH3_IDLE_THA_0	CH3_IDLE_THD_1	CH3_IDLE_THD_0	OVRD_FAST_IDLE	HI_IDLE_TH_CH0-3	HI_IDLE_TH_CH4-7	FAST_IDLE_CH0-3
SMBus Register			0x27 [3]	0x27 [2]	0x27 [1]	0x27 [0]	0x28 [6]	0x28 [5]	0x28 [4]	0x28 [3]
Default Value	0x09		0	0	0	0	1	0	0	1
Description		22 (0x16)	FAST_IDLE_CH4-7	LO_GAIN_CH0-3	LO_GAIN_CH4-7	CH4_IDLE_AUTO	CH4_IDLE_SEL	CH4_RXDET_1	CH4_RXDET_0	Reserved
SMBus Register			0x28 [2]	0x28 [1]	0x28 [0]	0x2B [5]	0x2B [4]	0x2B [3]	0x2B [2]	0x2C [7]
Default Value	0x80		1	0	0	0	0	0	0	0
Description		23 (0x17)	Reserved	Reserved	Reserved	Reserved	Reserved	CH4_EQ_1	CH4_EQ_0	CH4_SCP
SMBus Register			0x2C [6]	0x2C [5]	0x2C [4]	0x2C [3]	0x2C [2]	0x2C [1]	0x2C [0]	0x2D [7]
Default Value	0x5F		0	1	0	1	1	1	1	1
Description		24 (0x18)	Reserved	Reserved	Reserved	Reserved	CH4_VOD_2	CH4_VOD_1	CH4_VOD_0	CH4_DEM_2
SMBus Register			0x2D [6]	0x2D [5]	0x2D [4]	0x2D [3]	0x2D [2]	0x2D [1]	0x2D [0]	0x2E [2]
Default Value	0x5A		0	1	0	1	1	0	1	0
Description		25 (0x19)	CH4_DEM_1	CH4_DEM_0	Reserved	CH4_IDLE_THA_1	CH4_IDLE_THA_0	CH4_IDLE_THD_1	CH4_IDLE_THD_0	CH5_IDLE_AUTO
SMBus Register			0x2E [1]	0x2E [0]	0x2F [7]	0x2F [3]	0x2F [2]	0x2F [1]	0x2F [0]	0x32 [5]
Default Value	0x80		1	0	0	0	0	0	0	0
Description		26 (0x1A)	CH5_IDLE_SEL	CH5_RXDET_1	CH5_RXDET_0	Reserved	Reserved	Reserved	Reserved	Reserved
SMBus Register			0x32 [4]	0x32 [3]	0x32 [2]	0x33 [7]	0x33 [6]	0x33 [5]	0x33 [4]	0x33 [3]
Default Value	0x05		0	0	0	0	0	1	0	1

Table 1. EEPROM Address Map from DS125BR401A - Single Device with Default Value (continued)

EEPROM Address Byte			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Description		27 (0x1B)	Reserved	CH5_EQ_1	CH5_EQ_0	CH5_SCP	Reserved	Reserved	Reserved	Reserved
SMBus Register			0x33 [2]	0x33 [1]	0x33 [0]	0x34 [7]	0x34 [6]	0x34 [5]	0x34 [4]	0x34 [3]
Default Value	0xF5		1	1	1	1	0	1	0	1
Description		28 (0x1C)	CH5_VOD_2	CH5_VOD_1	CH5_VOD_0	CH5_DEM_2	CH5_DEM_1	CH5_DEM_0	Reserved	CH5_IDLE_THA_1
SMBus Register			0x34 [2]	0x34 [1]	0x34 [0]	0x35 [2]	0x35 [1]	0x35 [0]	0x36 [7]	0x36 [3]
Default Value	0xA8		1	0	1	0	1	0	0	0
Description		29 (0x1D)	CH5_IDLE_THA_0	CH5_IDLE_THD_1	CH5_IDLE_THD_0	CH6_IDLE_AUTO	CH6_IDLE_SEL	CH6_RXDET_1	CH6_RXDET_0	Reserved
SMBus Register			0x36 [2]	0x36 [1]	0x36 [0]	0x39 [5]	0x39 [4]	0x39 [3]	0x39 [2]	0x3A [7]
Default Value	0x00		0	0	0	0	0	0	0	0
Description		30 (0x1E)	Reserved	Reserved	Reserved	Reserved	Reserved	CH6_EQ_1	CH6_EQ_0	CH6_SCP
SMBus Register			0x3A [6]	0x3A [5]	0x3A [4]	0x3A [3]	0x3A [2]	0x3A [1]	0x3A [0]	0x3B [7]
Default Value	0x5F		0	1	0	1	1	1	1	1
Description		31 (0x1F)	Reserved	Reserved	Reserved	Reserved	CH6_VOD_2	CH6_VOD_1	CH6_VOD_0	CH6_DEM_2
SMBus Register			0x3B [6]	0x3B [5]	0x3B [4]	0x3B [3]	0x3B [2]	0x3B [1]	0x3B [0]	0x3C [2]
Default Value	0x5A		0	1	0	1	1	0	1	0
Description		32 (0x20)	CH6_DEM_1	CH6_DEM_0	Reserved	CH6_IDLE_THA_1	CH6_IDLE_THA_0	CH6_IDLE_THD_1	CH6_IDLE_THD_0	CH7_IDLE_AUTO
SMBus Register			0x3C [1]	0x3C [0]	0x3D [7]	0x3D [3]	0x3D [2]	0x3D [1]	0x3D [0]	0x40 [5]
Default Value	0x80		1	0	0	0	0	0	0	0
Description		33 (0x21)	CH7_IDLE_SEL	CH7_RXDET_1	CH7_RXDET_0	Reserved	Reserved	Reserved	Reserved	Reserved
SMBus Register			0x40 [4]	0x40 [3]	0x40 [2]	0x41 [7]	0x41 [6]	0x41 [5]	0x41 [4]	0x41 [3]
Default Value	0x05		0	0	0	0	0	1	0	1
Description		34 (0x22)	Reserved	CH7_EQ_1	CH7_EQ_0	CH7_SCP	Reserved	Reserved	Reserved	Reserved
SMBus Register			0x41 [2]	0x41 [1]	0x41 [0]	0x42 [7]	0x42 [6]	0x42 [5]	0x42 [4]	0x42 [3]
Default Value	0xF5		1	1	1	1	0	1	0	1
Description		35 (0x23)	CH7_VOD_2	CH7_VOD_1	CH7_VOD_0	CH7_DEM_2	CH7_DEM_1	CH7_DEM_0	Reserved	CH7_IDLE_THA_1
SMBus Register			0x42 [2]	0x42 [1]	0x42 [0]	0x43 [2]	0x43 [1]	0x43 [0]	0x44 [7]	0x44 [3]
Default Value	0xA8		1	0	1	0	1	0	0	0
Description		36 (0x24)	CH7_IDLE_THA_0	CH7_IDLE_THD_1	CH7_IDLE_THD_0	Reserved	Reserved	Reserved	Reserved	Reserved
SMBus Register			0x44 [2]	0x44 [1]	0x44 [0]	0x47 [3]	0x47 [2]	0x47 [1]	0x47 [0]	0x48 [7]
Default Value	0x00		0	0	0	0	0	0	0	0
Description		37 (0x25)	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
SMBus Register			0x48 [6]	0x4C [7]	0x4F [6]	0x4C [5]	0x4C [4]	0x4C [3]	0x4C [0]	0x59 [0]
Default Value	0x00		0	0	0	0	0	0	0	0
Description		38 (0x26)	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
SMBus Register			0x5A [7]	0x5A [6]	0x5A [5]	0x5A [4]	0x5A [3]	0x5A [2]	0x5A [1]	0x5A [0]
Default Value	0x54		0	1	0	1	0	1	0	0
Description		39 (0x27)	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
SMBus Register			0x5B [7]	0x5B [6]	0x5B [5]	0x5B [4]	0x5B [3]	0x5B [2]	0x5B [1]	0x5B [0]
Default Value	0x54		0	1	0	1	0	1	0	0

4 EEPROM Hex File Format

Hex files are an ASCII text file with the extension .hex, and the EEPROM format of choice for TI repeaters and mux buffers is the Intel hex file format. Intel hex file format is widely used for storing and transferring data in ROM, EEPROM, and microcontrollers. In an Intel hex file, each line consists of hexadecimal data. Below is an example of the first few lines of a .hex file for the DS80PCI800.

```
:2000000060001000330000580000 7D0000A20000C70000EC0000110100360 1005B010080EE
:200020000100A50100CA0100000000000000000000000000004270003ED20003ED2002FD4
:20004000AD4002FA1409E05F428005F5A8005F5A8005F5A800005454000004270003ED2039
:20006000003ED2002FAD4002FA1409E05F428005F5A8005F5A8005F5A80000545400000411
:20008000270003ED20003ED2002FAD4002FA1409E05F428005F5A8005F5A8005F5A8000066
```

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:

:2000000060001000330000580000 7D0000A20000C70000EC0000110100360 1005B010080EE

:	Start Code: Each line will always start with an ASCII colon (:) character
20	Byte Count: The number of data bytes (0x20) in each hex line.
0000	Load Offset: Starting load offset of data bytes.
00	Record type: There are six record types. For repeater and mux buffer programming, this value will 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.
60…80	Data: There are *n* bytes of data per line (*n* specified by Byte Count parameter).
EE	Checksum: Represents the checksum of the record.

In order 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 + 60 + 00 + 10 + 00 + 33 + 00 + 00 + 58 + 00 + 00 + 7D + 00 + 00 + A2 + 00 + 00 + C7 + 00 + 00 + EC + 00 + 00 + 11 + 01 +00 + 36 + 01 + 00 +5B +01 + 00 + 80 = 512

then do two's complement

$$1'h + not(512'h) = 1'h + 2ED'h = 2EE'h \quad (1)$$

NOTE: Application tools are often used to calculate this checksum automatically.

Every line with the exception of the last line of hex will follow this format scheme. The last line of hex will always be the End-of-File (EOF) record, shown as: 00000001FF.

5 EEPROM Device Data Fundamentals

Each repeater and mux buffer 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 2.

Table 2. 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 (37 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. This is necessary if there are more than 4 EEPROM slots. 0 = Required EEPROM size is 256 bytes or less. This value indicates that up to 4 EEPROM slots can be programmed.
	4	RES	Reserved. Set bit to 0.
	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 TI's high speed repeaters and mux buffers.

5.2 Address Map Header

When multiple devices are used, address map headers are necessary. In order 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 3.

Table 3. 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 (37 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	7:0	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, as indicated by asserting Base Header Address 0x00[5] = 1. For example, if the EEPROM start address is located at Address 0x1F4, 9 bits are required. Thus, Address Map Header Byte 1 = 0xF4, and Address Map Header Byte 2 = 0x01. If EEPROM ≤ 256 bytes, then the address map header will be 2 bytes, not 3 bytes.

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 via the CRC-8 polynomial, where the input x = [Base Header (3 Bytes) + Address Map header (1 or 2 Bytes) + Device Data (37 Bytes)]. An example is provided below:

Table 4. EEPROM CRC-8 Example

Section	Value (Hex)
Base Header	0xC00010
Address Map Header	0x23
Device Data	0x000004070000AA80000AA80000AA80000AA8 0080015500001550000155000015500000 5454
CRC-8 Input	0xC000102300000407000 0AA80000AA80000AA8000 0AA8008001550000155000015500001550000 05454
Computed CRC-8 (Address Map Header Byte 0)	0xB6

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 Example 1: EEPROM Hex File for 1 Device, CRC Disabled

The simplest case for programming EEPROM is programming for a single device. The following are key factors to consider when programming a single device:

- Address Map Header can typically be disabled, since the EEPROM does not need to reference the start address of multiple-device EEPROM data.
- Single-device configurations will not require more than 256 bytes of EEPROM.
- Only one slot is needed.

Below is an example of a hex file for a single DS80PCI800 device, shown in Figure 3. The data relevant to the DS80PCI800 EEPROM address bits is highlighted in green.

```
:20000000000010000000407002FAD4002FAD4002FAD4002FAD400805F5A8005F5A8005F5AD9
:200020008005F5A8000054540000000000000000000000000000000000000000000000F6
:2000600000000000000000000000000000000000000000000000000000000000000000080
:2000800000000000000000000000000000000000000000000000000000000000000000060
:2000A0000000000000000000000000000000000000000000000000000000000000000040
:2000C0000000000000000000000000000000000000000000000000000000000000000020
:2000E0000000000000000000000000000000000000000000000000000000000000000000
:00000001FF
```

= EEPROM Base Header

= Device Address Map Header + Data

Figure 3. Example of a Hex File for a Single DS80PCI800 Device

From the DS80PCI800 hex file, the base header bytes are 0x000010. From Table 2, 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).
- DEVICE COUNT = 1 Device (Address 0x00[3:0] = 0000'b).
- Max EEPROM Burst size = 16 bytes (Address 0x02 = 0x10).

Since no address map header is used, the remaining 37 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 EEPROM Address 0x03-0x27.

7 Example 2: EEPROM Hex File for 4 EEPROM slots, CRC Enabled

When programming EEPROM with up to 4 EEPROM slots, the following are considered:

- Address Map Header must be used.
- If 1-4 EEPROM slots are needed, then EEPROM size ≤ 256 bytes is adequate.
- Address Map Header will be 2 bytes.

Below is an example of a hex file for 4 x DS80PCI800 devices (4 unique EEPROM slots) that are CRC enabled, shown in Figure 4.

```
:20000000C30010B6235548 6D6D DF92 00000000000000000000000000000000000000000000004C
:20002000000000 00000407 0000 AA80000AA80000AA80000AA8008001550000155000001556C
:20004000 00001550000005454 00000407 0000 AA80000AA80000AA80000AA8008001550000FA
:20006000155000015500001550000054540000040700 00AA40000AA40000AA40000AA4007D
:20008000 8001548000154800015480001548000054540000040700 00AA40000AA40000AA87
:2000A000 40000AA400800154800015480001548000154800005454 00000000000000000C6
:2000C000000000000000000000000000000000000000000000000000000000000000000020
:2000E000000000000000000000000000000000000000000000000000000000000000000000
:00000001FF
```

= EEPROM Base Header

= Device 0 Address Map Header + Data

= Device 1 Address Map Header + Data

= Device 2 Address Map Header + Data

= Device 3 Address Map Header + Data

Figure 4. Example of a Hex File for 4 x DS80PCI800 Devices (4 Unique EEPROM Slots) that are CRC enabled

The base header bytes are 0xC30010. From Table 2, 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).
- DEVICE COUNT = 4 Devices (Address 0x00[3:0] = 0011'b).
- Max EEPROM Burst size = 16 bytes (Address 0x02 = 0x10).

There are four address map headers specific for each device. Note that only two bytes are used for the address map header, since the EEPROM ≤ 256 bytes. From Table 3, the following is derived:

- Device 0 [CRC, Start Address] = [0xB6, 0x23]
- Device 1 [CRC, Start Address] = [0x55, 0x48]
- Device 2 [CRC, Start Address] = [0x6D, 0x6D]
- Device 3 [CRC, Start Address] = [0xDF, 0x92]

By searching for the start address relevant to each device, the remaining 37 bytes for that device's register settings can be found. For example, in Device 3, the 37 bytes of device data begin at EEPROM Address 0x92.

8 Example 3: EEPROM Hex File for 12 Devices, CRC Disabled

When programming EEPROM for more than 4 EEPROM slots, the following are considered:

- Address Map Header must be used.
- If more than 4 EEPROM slots are required, then EEPROM size > 256 bytes is required. This means that the Address Map Header is 3 bytes.

Below is an example of a hex file for 12 x DS100BR111 devices (11 unique EEPROM slots), shown in Figure 5.

```
:200000006B0010003300005800007D0000A20000C70000EC0000110100360100 5B010080E3
:2000200001 00A50100A501 00000000000000000000000000000004270003ED20003ED2002FD4
:20004000AD4002FA1409E05F428005F5A8005F5A8005F5A800005454000004270003ED2039
:20006000003ED20003AD4002FA1409E05F428005F5A8005F5A8005F5A800005454000004 3D
:20008000270003ED20003AD2002FAD4002FA1409E05F428005F5A8005F5A8005F5A80000 6A
:2000A0005454000004270003ED20003ED2001FAD4002FA1409E05F428005F5A8005F5A804C
:2000C00005F5A800005454000004270003ED20003ED2002FAD40007A1409E05F428005F5DD
:2000E000A8005F5A8005F5A800005454000004270003ED20003AD2002FAD4000FA1409E07B
:200100005F428005F5A8005F5A8005F5A800005454000004270003ED20003ED2002FED40F2
:2001200002FA1409E05F428005F5A8005F5A8005F5A800005454000004270003ED20003E07
:20014000D20001AD4002FA1409E05F428005F5A8005F5A8005F5A800005454000004270075
:2001600003ED20003ED2002FAD40000A1409E05F428005F5A8005F5A8005F5A800005454F6
:20018000000004270003ED20003ED2002FED4003FA1409E05F428005F5A8005F5A8005F5C8
:2001A000A800005454000004270003ED20003ED200AAAD4002FA1409E05F428005F5A80051
:2001C0005F5A8005F5A8000054540000000000000000000000000000000000000000009C
:2001E000000000000000000000000000000000000000000000000000000000000000FF
:00000001FF
```

= EEPROM Base Header

= Device 0 Address Map Header + Data

= Device 1 Address Map Header + Data

= Device 2 Address Map Header + Data

= Device 3 Address Map Header + Data

= Device 4 Address Map Header + Data

= Device 5 Address Map Header + Data

= Device 6 Address Map Header + Data

= Device 7 Address Map Header + Data

= Device 8 Address Map Header + Data

= Device 9 Address Map Header + Data

= Device 10 Address Map Header + Data

= Device 11 Address Map Header (Shares Device 10 Data Slot)

Figure 5. Example of a Hex File for 12 x DS100BR111 Devices (11 Unique EEPROM Slots)

The base header bytes are 0x6B0010. From Table 2, 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] = 1'b).
- DEVICE COUNT = 12 Devices (Address 0x00[3:0] = 1011'b).
- Max EEPROM Burst size = 16 bytes (Address 0x02 = 0x10).

Since CRC is disabled, each Address Map Header Byte 0 is 0x00. In this example, Device 10 and Device 11 share the same address map header. This occurs if multiple devices are programmed with identical SMBus register settings.

There are 10 unique address map headers for 11 devices. Unlike the previous examples, the address map header here has 3 bytes, since the EEPROM > 256 bytes. Recall that when the address map header is 3 bytes, the 3 LSBs of the Address Map Header Byte 2 become the 3 MSBs of the EEPROM start address. Thus, from Table 3, the following is derived:

- Device 0 [Start Address] = 0x33
- Device 1 [Start Address] = 0x58
- Device 2 [Start Address] = 0x7D
- Device 3 [Start Address] = 0xA2
- Device 4 [Start Address] = 0xC7
- Device 5 [Start Address] = 0xEC
- Device 6 [Start Address] = 0x111
- Device 7 [Start Address] = 0x136
- Device 8 [Start Address] = 0x15B
- Device 9 [Start Address] = 0x180
- Device 10 [Start Address] = 0x1A5
- Device 11 [Start Address] = 0x1A5

By searching for the start address relevant to each device the remaining 37 bytes that are used to program the device can be found. For example, in Device 6, the 37 bytes of device data begin at EEPROM Address 0x111.

9 Summary

In this application note, the benefits of EEPROM are explored as they relate to TI's high speed 2-channel repeaters, 8-channel repeaters, and 2:1/1:2 mux buffers. 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 Intel hex format are revealed to help users differentiate between EEPROM sections relevant to formatting and EEPROM sections relevant to the device settings. With a complete understanding of how to program and interpret these EEPROM hex files, system designers are better equipped to generate their own customized hex files and increase the efficiency of their final designs.

10 References

1. "Intel Hexadecimal Object File Format Specification", Revision A, 1/6/88.
2. DS80PCI800 Datasheet (SNLS334)
3. DS125BR401A Datasheet (SNLS466)

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

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