



Intel® 41210 Serial to Parallel PCI Bridge Initialization by the SM Bus Using The PIC16F876A Microcontroller

White Paper

May 2004



INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. Intel products are not intended for use in medical, life saving, life sustaining applications.

Intel may make changes to specifications and product descriptions at any time, without notice.

Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an ordering number and are referenced in this document, or other Intel literature may be obtained by calling 1-800-548-4725 or by visiting Intel's website at <http://www.intel.com>.

Intel is a trademark or registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2004, Intel Corporation

Contents

1.0	Introduction	5
1.1	Workaround Definition	5
1.2	Affected Errata.....	5
2.0	Workaround Implementation	6
2.1	Errata 19	6
2.2	Errata 20	8
2.3	Errata 25.....	9
A	How Workarounds Are Implemented	14
B	EEPROM MAP	16

Figures

1	Initialization Overview.....	5
2	Customer Reference Code for Errata 19.....	7
3	Errata 19 Implementation by SM Bus Protocol.....	7
4	Errata 20 Implementation by the Customer Reference Code.....	8
5	Errata 20 Implementation by SM Bus Protocol.....	9
6	Errata 25 Implementation by the Customer Reference Code.....	11
7	Errata 25 Implementation by SM Bus Protocol (BCTRL: Offset 0x03E).....	12
8	Errata 25 Implementation by SM Bus Protocol (EXP_CTL: Offset 0x04C)	12
9	Errata 25 Implementation by SM Bus Protocol (PCIEXERRRUNC_MSK: Offset 0x130)	13
10	Errata 25 Implementation by SM Bus Protocol (PCIEXERRRUNC_SEV: Offset 0x134)	13
11	Updating bit 15 - 9 of the BCNF Register (0x40) for both functions.	15

Tables

1	Byte 0 Definition.....	14
2	Byte 1 Definition.....	15
3	Internal EEPROM Memory Map	17
4	EEPROM Data by Register	19

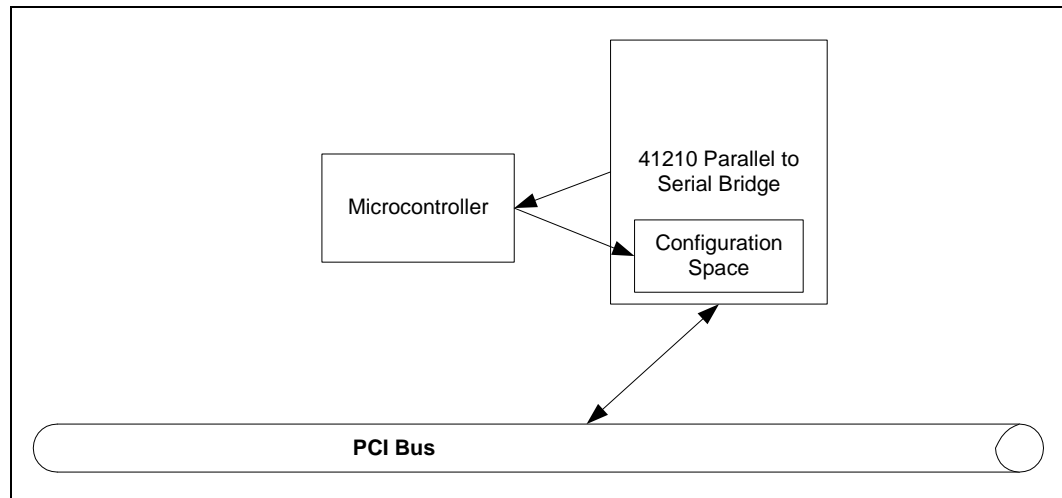
1.0 Introduction

In the 41210 Bridge, some of the current errata (19, 20, and 25) that exist require you to load space values into registers before you can perform workarounds for those errata. This paper describes the methodology of loading these space registers for each of these errata using the PIC16F876A microcontroller attached to the 41210 Bridge and its I2C bus.

The configuration register changes must usually be done prior to PCI enumeration. This means you need to set the CFGRETRY pin to high during power on.

When you set the CFGRETRY pin to high, the signal is interrupted and the 41210 Serial to Parallel Bridge will not respond to any configuration commands coming from the PCI-Express bus. This allows the PIC16F876A microcontroller to update the configuration registers on the device by the SM Bus. Once you have updated the desired configuration registers, the condition is cleared (or released) by clearing the Configuration Cycle Retry bit (Bit 3) in the Bridge Initialization Register (BINIT: 0xFC) for both functions. This allows PCI enumeration to proceed.

Figure 1. Initialization Overview¹



1.1 Workaround Definition

The workarounds for a number of errata observed in the Intel® 41210 Serial to Parallel PCI Bridge require configuration space registers be loaded with specific values. Intel recommends that you use the PIC16F876A microcontroller attached to the 41210 Bridge SM Bus prior to releasing the CFGRETRY signal.

1.2 Affected Errata

The following current 41210 Bridge erratum are affected by this white paper:

- Errata 19 in [Section 2.1](#)

1. *This information can only be used for the 41210 Serial to Parallel Bridge and may not be used on any other devices, and may not be used for any other purposes without Intel's written permission.*

- Errata 20 in [Section 2.2](#)
- Errata 25 in [Section 2.3](#)

Refer to [Section 2.0](#) for a complete description of these errata and how to implement workarounds once you have set register values.

2.0 Workaround Implementation

The workaround implementations for 41210 Bridge Erratum 19, 20, 24, and XX are described in this section. For a more general description of how these workarounds are implemented, refer to [Appendix A, “How Workarounds Are Implemented”](#).

Workaround Implementations are discussed in two categories for each errata:

- By Pseudo Code Implementation
- By Customer Reference Code

Note: All of these implementations must be performed prior to PCI enumeration.

2.1 Errata 19

Errata Description: The PCIExpress link is unreliable if the Intel® 41210 Serial to Parallel PCI Bridge enters L0s state

When L0s is enabled, the system may hang or behave in an unstable manner.

Errata Workaround

The workaround for this is to insure that entry into the L0s state is prevented using control bits in the EXP_LCTL control register Bits 0 and 1 of the PCI Express link control register at Configuration offset 54h must be set to 00b to disable L0s entry.

2.1.1 Errata 19 Workaround Implementation

2.1.1.1 Pseudo Code Implementation

Perform the following step:

1. Write 0x00 to 1st Byte EXP_LCTL - PCI Express Link Control Register (0x54).

2.1.1.2 Customer Reference Code Implementation

[Figure 2](#) shows the Customer Reference Code Implementation for Errata 19.

Figure 2. Errata 19 Customer Reference Code

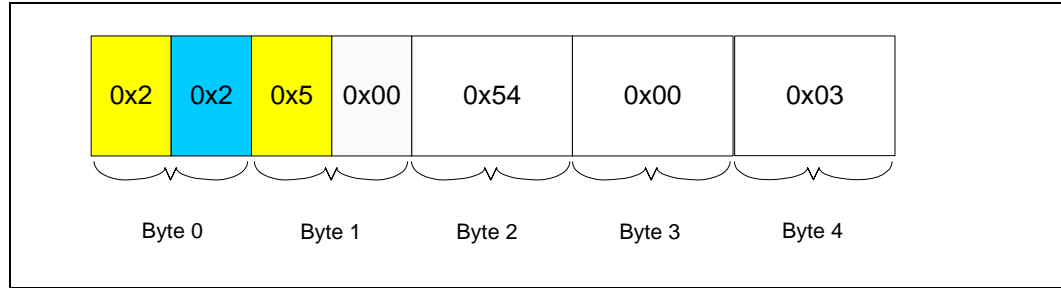
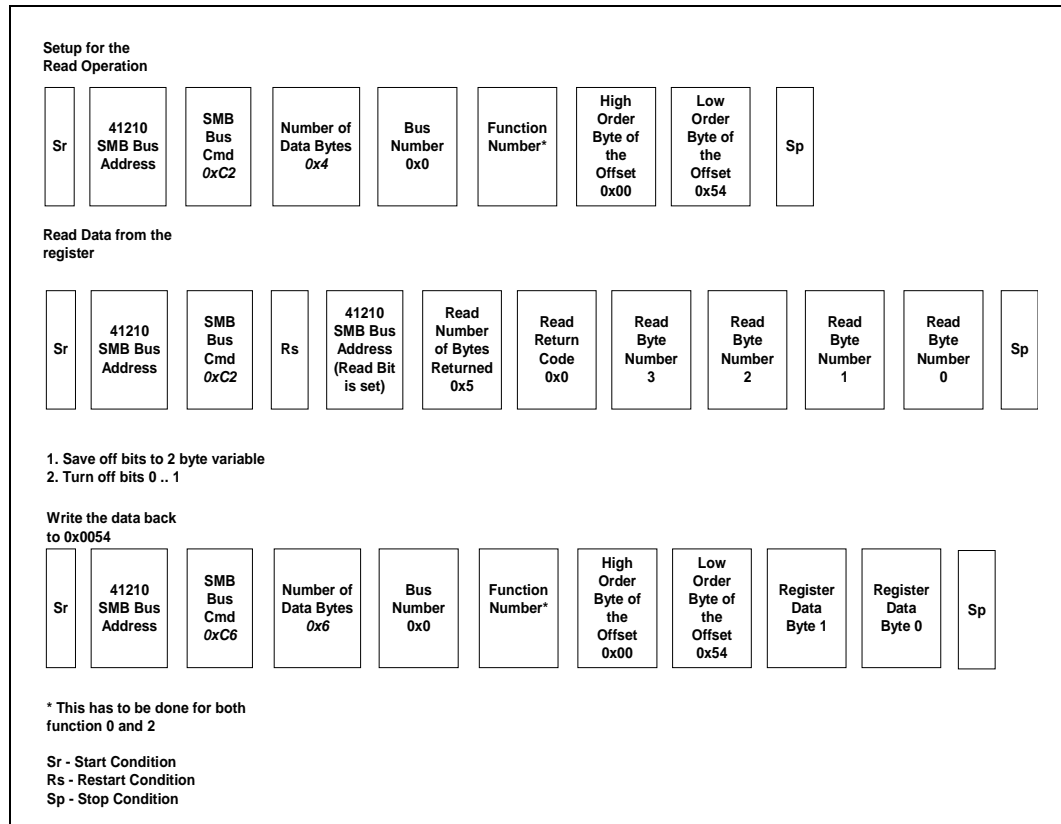


Figure 3. Errata 19 Implementation by SM Bus Protocol



2.2 Errata 20

Errata Description: Slow edge rates are observed when the Intel® 41210 Serial to Parallel PCI Bridge is driving the PCI-X bus at specific temperatures

The internal circuit used to set PCI/PCIX drive levels at power up does not always function correctly, possibly causing indeterminate drive levels on the PCI/PCI-X bus. This can be prevented by setting Bits 29:17 of the Configuration register at offset 224 to 1 for both PCI functions (0 and 2).

Errata Workaround

The workaround for this is to set bits of the compensation register, 0x224, starting at offset 0x226 of both PCI functions with 03FE h

2.2.1 Errata 20 Workaround Implementation

2.2.1.1 Pseudo Code Implementation

Perform the following steps:

1. Read 2 bytes from Offset 0x224, Compensation Register, into a variable that is 2 bytes long.
2. Set bits 29:17 on in the variable, 0x3FFE.
3. Write 2 bytes stored in the variable back to 0x224.

2.2.1.2 Customer Reference Code Implementation

Figure 4. Errata 20 Implementation by the Customer Reference Code

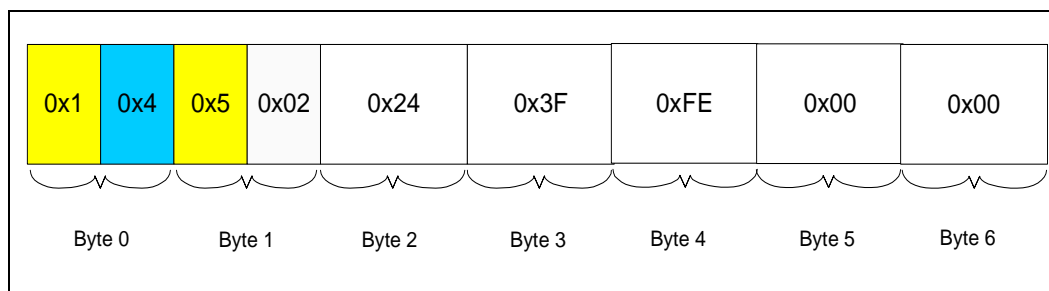
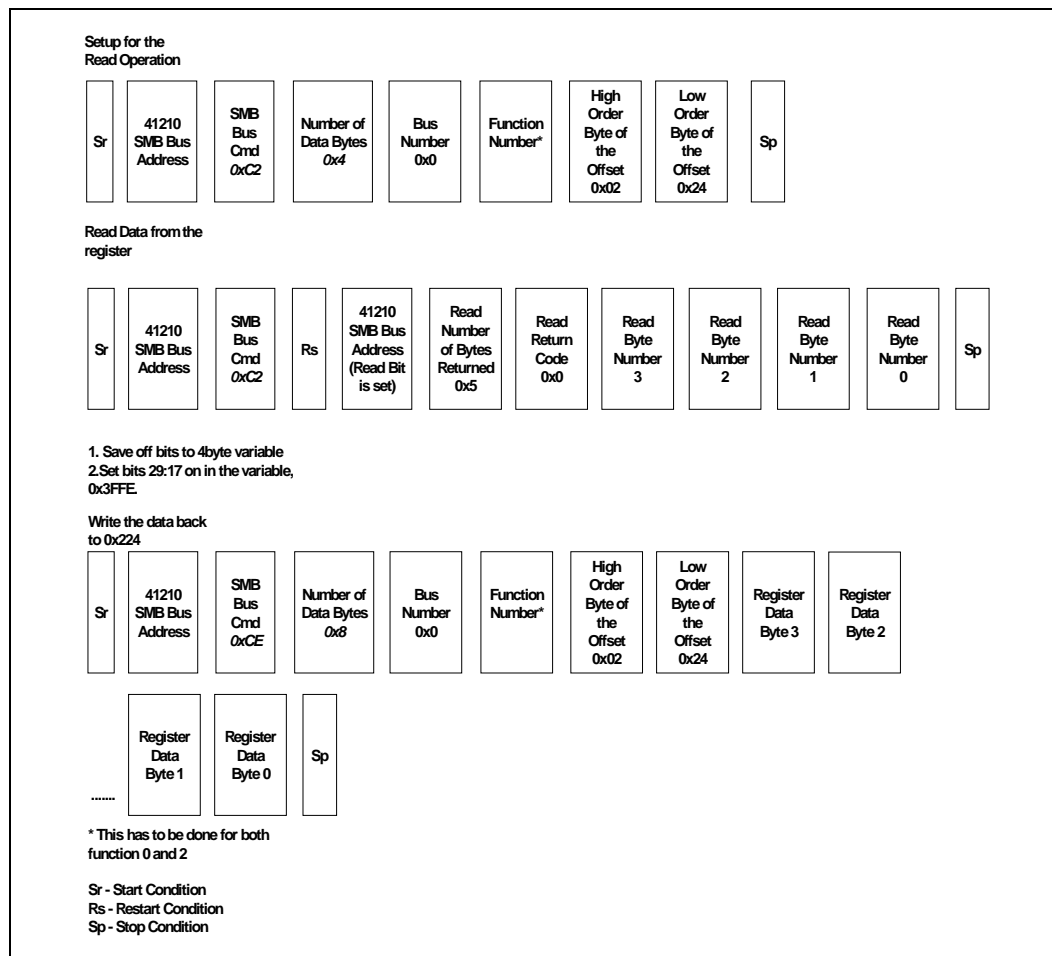


Figure 5. Errata 20 Implementation by SM Bus Protocol



2.3 Errata 25

Errata Description: PCI Parity error without ep field being set

In PCI and PCI-X mode using 32-bit data transfers, when a read request gets disconnected at an even dword boundary with data parity error, such that the subsequent request (for partial data) gets retried, the completion for this request is issued over PCI-Express to MCH (root complex) without the poisoned data 'ep' field set in PCI-Express TLP header. This could result in silent data corruption if escalation of uncorrectable errors is not enabled by BIOS.

Note: There would be a potential race condition between the completion being returned to the CPU and the error escalation. Read completion may complete normally at the CPU followed by error escalation (Interrupt/SERR). Under normal circumstances corrupted completions would complete at the CPU and would trigger error handlers.

Errata Workaround

To prevent such a parity error from going undetected, the following bits must be set in the 41210 device, and error escalation must be properly set in the PCIE root complex.

Offset 3E: BCTRL – Bridge Control

Set bit 01 to ‘1’

SERR# Enable (SE): Controls the forwarding of secondary interface SERR# assertions on the primary interface. Enables Non-Fatal/Fatal Error reporting and will also set the signaled SERR# bit when an uncorrectable error message is sent.

Offset 4C: EXP_DCTL – EXP Device Control Register

Set bit 02 to ‘1’

Report Fatal Errors: When this bit is set, 41210 is enabled to generate ERR_FATAL message on EXP.

Offset 130: PCIXERRUNC_MSK – Uncorrectable PCI/X Error Mask Register

Set bit 07 to ‘0’

PCI/X Uncorrectable Data Parity Error Detected Mask

Offset 134: PCIXERRUNC_SEV – Uncorrectable PCI/X Error Severity Register

Set bit 07 to ‘1’

PCI/X Uncorrectable Data Parity Error Detected Severity

2.3.1 Errata 25 Workaround Implementation

2.3.1.1 Pseudo Code Implementation

1. Perform the following steps for **Offset 3E: BCTRL – Bridge Control**:
 - a. Read 2 bytes from Offset 0x03E, BCTRL, into a variable that is 2 bytes long.
 - b. Set bit 1 on of the variable, 0x2
 - c. Write 2 bytes stored in the variable back to 0x03E.
2. Perform the following steps for **Offset 4C: EXP_DCTL – EXP Device Control Register**:
 - a. Read 2 bytes from Offset 0x04C, EXP_DCTL – EXP, into a variable that is 2 bytes long.
 - b. Set bit 2 on of the variable, 0x4
 - c. Write 2 bytes stored in the variable back to 0x04C
3. Perform the following steps for **Offset 130: PCIXERRUNC_MSK – Uncorrectable PCI/X Error Mask Register**:
 - a. Read 2 bytes from Offset 0x130, PCIXERRUNC_MSK, into a variable that is 2 bytes long.
 - b. Turn Off bit 7 on of the variable, 0x8

- c. Write 2 bytes stored in the variable back to 0x130
- 4. Perform the following steps for **Offset 134: PCIXERRUNC_SEV – Uncorrectable PCI/X Error Severity Register**:
 - a. Read 2 bytes from Offset 0x134, PCIXERRUNC_MSK, into a variable that is 2 bytes long.
 - b. Set bit 7 on of the variable, 0x8
 - c. Write 2 bytes stored in the variable back to 0x134

2.3.1.2 Customer Reference Code Implementation

Figure 6. Errata 25 Implementation by the Customer Reference Code

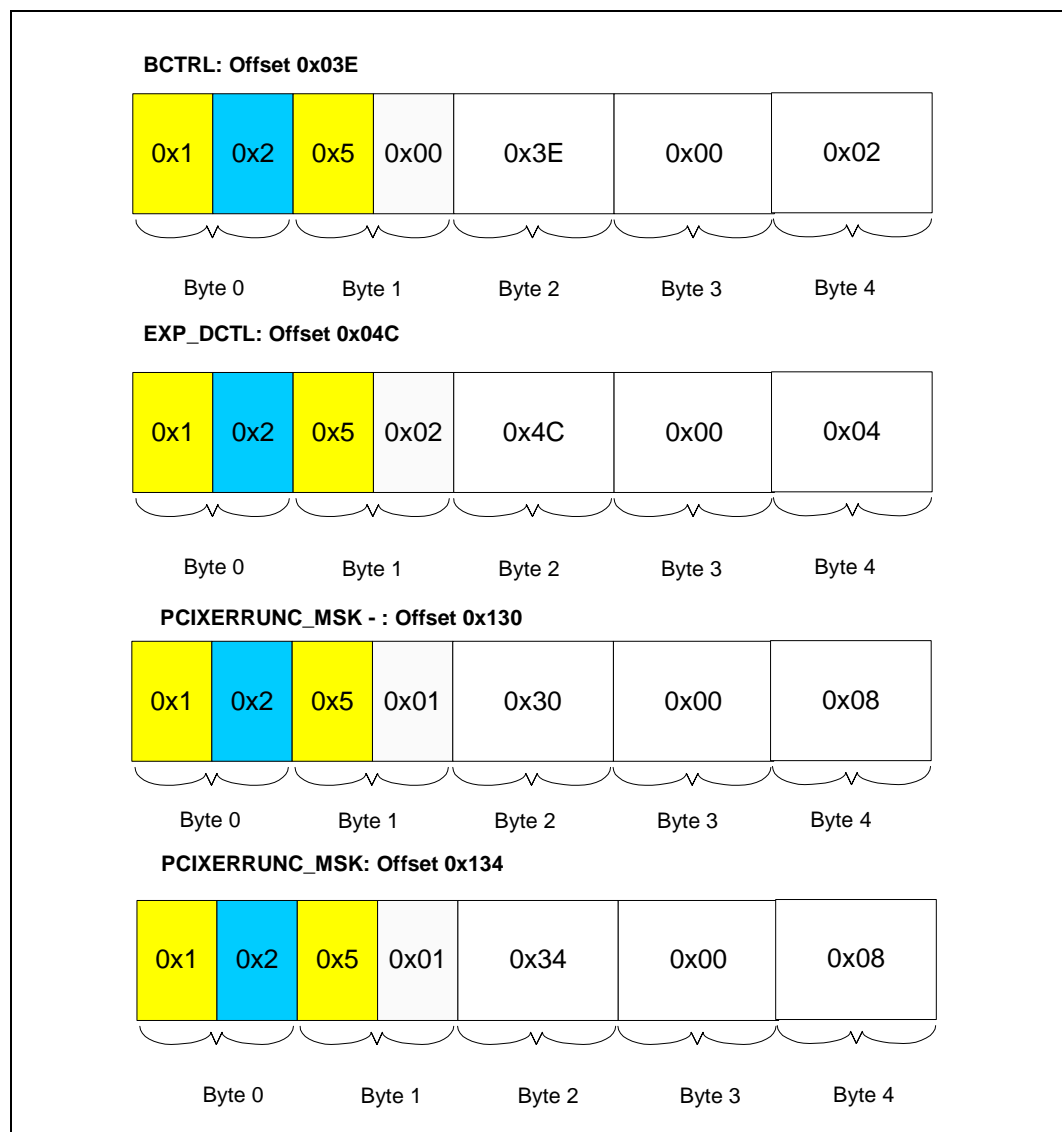


Figure 7. Errata 25 Implementation by SM Bus Protocol (BCTRL: Offset 0x03E)

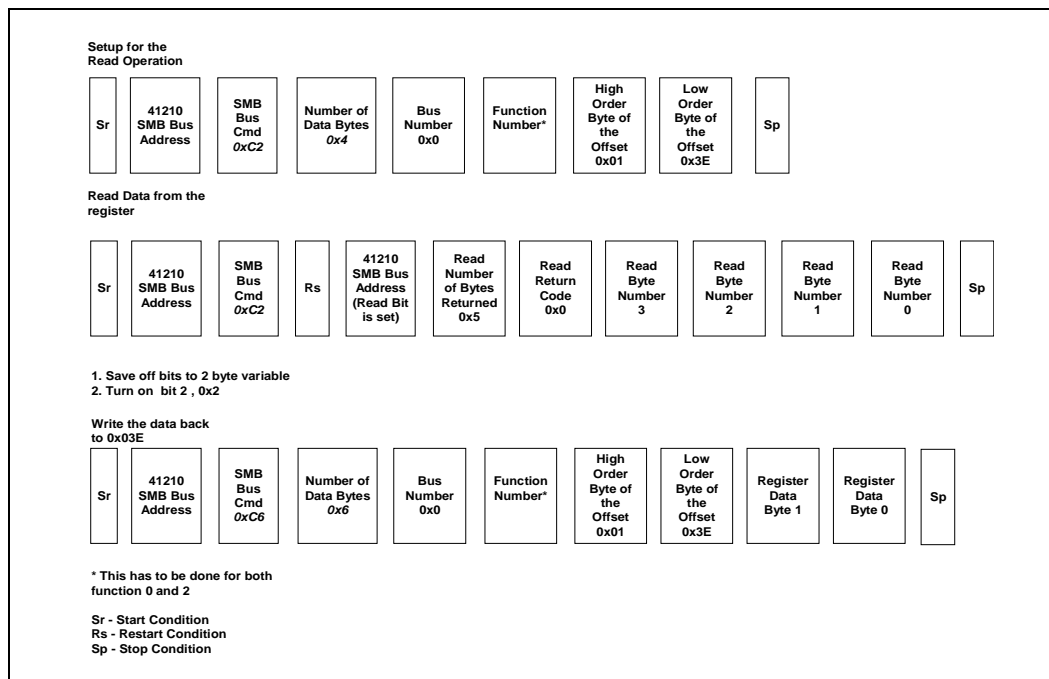


Figure 8. Errata 25 Implementation by SM Bus Protocol (EXP_CTL: Offset 0x04C)

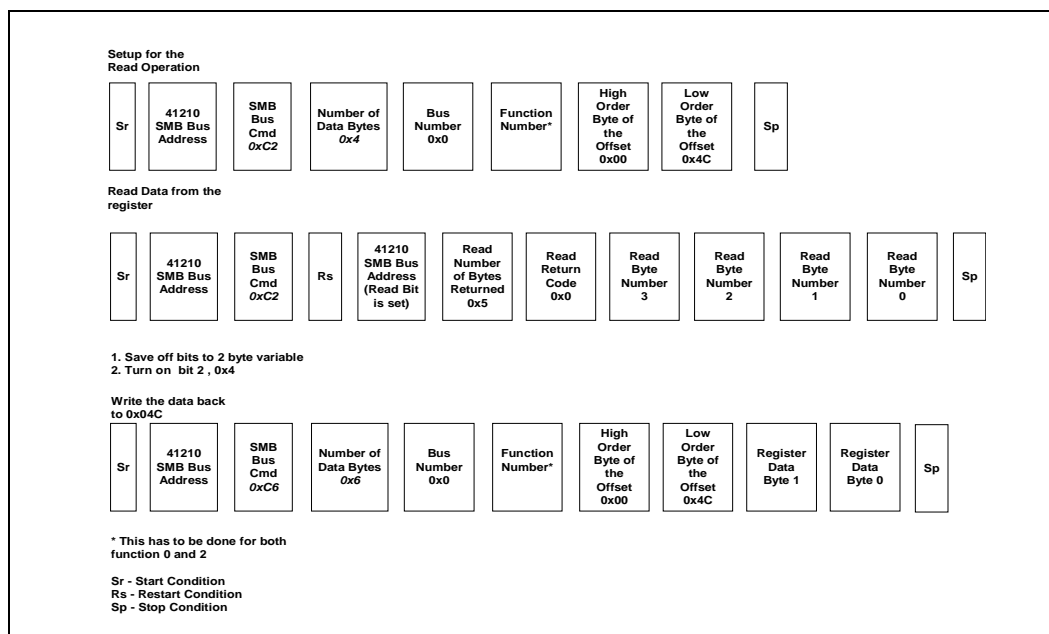


Figure 9. Errata 25 Implementation by SM Bus Protocol (PCIEXERRUNC_MSK: Offset 0x130)

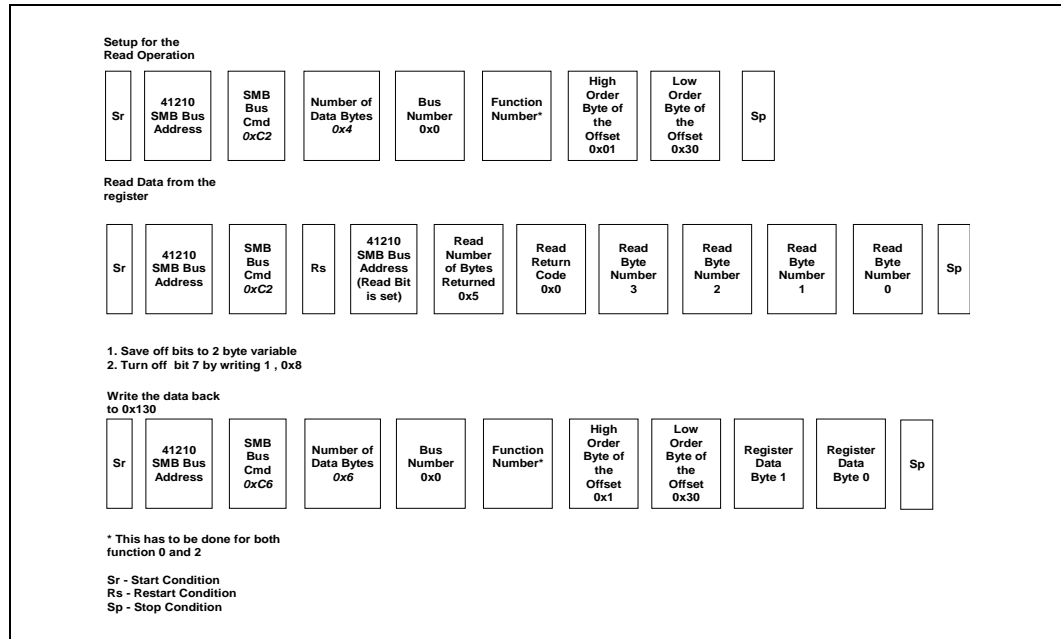
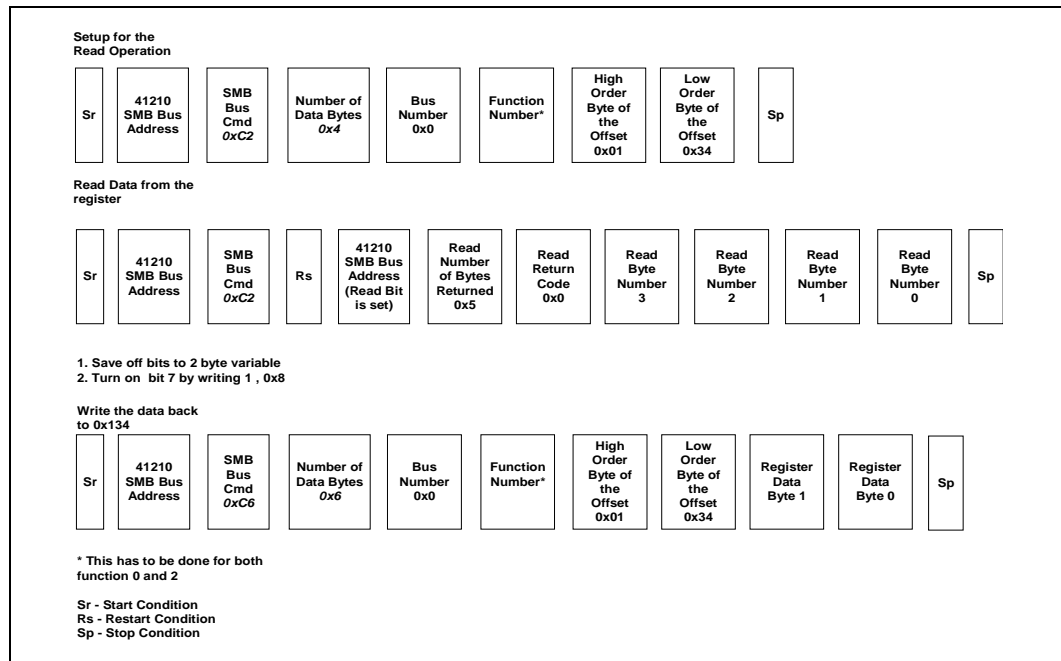


Figure 10. Errata 25 Implementation by SM Bus Protocol (PCIEXERRUNC_SEV: Offset 0x134)



Appendix A How Workarounds Are Implemented

This appendix provides information that may be helpful when implementing 41210 Bridge errata workarounds located in [Section 2.0](#) of this document.

Workarounds are discussed in two categories:

1. **General Logic**, description of the workaround in generic pseudo code logic.
2. **Customer Reference Source Code**, the source code implements storing of register information as packets on the internal EEPROM of the PIC16F876A Microcontroller. The register data is extracted from the internal EEPROM and transferred to the 41210 configuration space using SM Bus protocols. Byte 0 of the internal EEPROM will be set to 1. The following is a brief definition of the register definition packet.

A.1 Byte Definitions

This section provides byte definitions for the 41210 Bridge.

A.1.1 Byte 0

- **High Order Nibble- Read-Modify Write Bits (6..4)** : Setting these bits identifies that register data associated with this packet is used to update specific bits in a register. If either one these bits are not set then the *entire* register is set to the value of the register data associated. Each packet can only have only one of these bits turn on.
- **Low Order Nibble-Number of Data Bytes**: The number of bytes data associated to the packet. For example, if the value is 0x6, the value would be b'0110'.

Table 1. Byte 0 Definition

High Order Nibble				Low Order Nibble			
Bit 7 Not Used	Bit 6 (Read Modify Write – Bits And Mask)	Bit 5 (Read Modify Write – Bits Off)	Bit 4 (Read Modify Write – Bits On)	Bit 3 (Byte Count Bit 3)	Bit 2 (Byte Count Bit 2)	Bit 1 (Byte Count Bit 1)	Bit 0 (Byte Count Bit 0)
	1 - RModW – On 0 - RModW - Off	1 - RModW – On 0 - RModW - Off	1 - RModW – On 0 - RModW - Off	Depends on the desired value	Depends on the desired value	Depends on the desired value	Depends on the desired value

A.1.2 Byte 1

- **High Order Nibble-Target Function Bit**: Identifies which 41210 device function the register data is to be written.
 - 0 – Function 0
 - 2 – Function 2
 - 5 – Both Functions

- **Low Order Nibble-High Order Offset:** This is the high order bit of the register offset. In cases where the offset is less than 0x100, the value would be 0. If the value is 0x2, the value would be b'0010'.

Table 2. Byte 1 Definition

High Order Nibble				Low Order Nibble			
Bit 7 Not Used	Bit 6 (Function)	Bit 5 (Function)	Bit 4 (Function)	Bit 3 Not Used	Bit 2 (High Order Bit2)	Bit 1 (High Order Bit 1)	Bit 0 (High Order Bit 0)
	5 - Function 0 - 2	2 - Function 2	0 - Function 0 5 - Function 0 - 2		Depends on the desired value	Depends on the desired value	Depends on the desired value

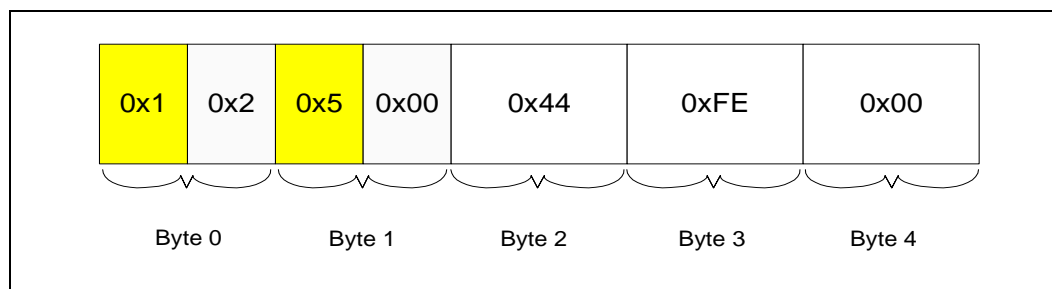
A.1.3 Byte 2

Low Order Offset, the low order byte of the register offset.

A.1.4 Byte 3... Byte n

Register Data, register data defined from high to low order bytes.

Figure 11. Updating bit 15 - 9 of the BCNF Register (0x40) for both functions.



A.2 SM Bus Protocol

The *Intel® 41210 Serial to Parallel PCI Bridge Developers Manual* describes how the workarounds are implemented using SM Bus command set. This information is provided for those who want to write their own implementation.

Appendix B EEPROM MAP

This appendix provides information on the internal EEPROM map.



Table 3. Internal EEPROM Memory Map

Address	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07
0x00	0x01	0x14	0x52	0x24	0x3F	0xFE	0x00	0x00
0x08	0x22	0x50	0x54	0x0	0x03	0x12	0x50	0x3E
0x10	0x00	0x02	0x12	0x50	0x4C	0x00	0x04	0x12
0x18	0x51	0x30	0x00	0x08	0x12	0x51	0x34	0x00
0x20	0x08	0xDB	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF
0x28	0xD5	0xFE	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF
0x30	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF
0x38	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF
0x40	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF
0x48	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF
0x50	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF
0x58	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF
0x60	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF
0x68	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF
0x70	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF
0x78	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF
0x80	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF
0x88	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF
0x90	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF
0x98	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF
0Xa0	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF
0Xa8	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF
0xb0	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF
0Xb8	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF
0xc0	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF
0xc8	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF
0xd0	0xFF	0xFF	0xFF	0xFF	0xFF	0xFE	0xFF	0xFF
0xd8	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF
0xe0	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF

Address	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07
0xe8	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF
0xf0	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF
0xf8	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF

Note: This table is setup for 0x100 byte internal EEPROM.

The following table identifies the Internal EEPROM Memory Map Color Coding (used in Table 3).

Table 4. Internal EEPROM Memory Map Color Coding

Color	Identifier	Notes
	Software Control Byte	This byte supports software controlling the processing of register data. 0x01 – Process Register Data. 0x00 – Do not process register data
	Register Definition Packet	Register Definition Packet coded by color to help identify individual register definition packets
	Register Definition Packet	Register Definition Packet coded by color to help identify individual register definition packets
	No Operation Packet	Internal EEPROM bytes to be ignored. Register Definition Packet will be set to the following values: <ol style="list-style-type: none"> 1. Byte 0 – Length of bytes to be ignored. The value is calculated by using the following formula: <i>Internal EEPROM Size – 3 -1-Highest of the previous packet</i>. 3 takes into account the packet header and 1 corrects for the memory starting at 0. In Note: This table is setup for 0x100 byte internal EEPROM. The table is setup for 0x100 byte internal EEPROM 2. The value of 0xDB Table 3 Internal EEPROM Memory Map is derived from 0x100- 0x03-0x01-0x20. 3. Byte 1 – 0xFF 4. Byte 3 – 0xFF



The following table describes the EEPROM register data.

Table 5. EEPROM Data by Register

Errata	Register	Mnemonic	Color	Packet Data
19	0x054	EXP_LCTL	Orange	0x22, 0x50, 0x54, 0x00, 0x03
20	0x224	Compensation Register	Magenta	0x14, 0x52, 0x24, 0x3F, 0xFE, 0x00, 0x00
25	0x03E	BCTRL	Magenta	0x12, 0x50, 0x3E, 0x00, 0x02
	0x04C	EXP_DCTL	Orange	0x12, 0x50, 0x4C, 0x00, 0x04
	0x130	PCIXERRUNC_MSK	Magenta	0x12, 0x51, 0x30, 0x00, 0x08
	0x134	PCIXERRUNC_SEV	Orange	0x12, 0x51, 0x34, 0x00, 0x08

This page intentionally left blank.