I2C Transaction Summary for NXP LPC Microcontrollers

March 5, 2011

Todd Comins

Introduction

The purpose of this document is to help people better understand the low-level operation of the I2C controller used in NXP LPC microcontrollers (LPC21xx, LPC17xx and others). This information contained here should apply to all NXP microcontrollers and microprocessors that have the same I2C controller hardware used in the NXP LPC21xx and LPC17xx series ARM microcontrollers.

The primary contribution of this document is a shorthand notation for I2C transactions that is annotated with specific interrupt and status information. This information helps to connect the progression of an I2C transaction (from start to finish) with the I2C controller state.

Below are links to key references used in the development of this document:

I2C Bus Specification and User Manual

I2C Manual - NXP Application Note 10216

SMBUS Protocol Summary

LPC2103 User Manual

LPC1768 User Manual

If you **do not** already understand I2C signalling and wire protocol then you must review the first two references (I2C Bus Specification and I2C Manual) before this document will be useful to you. If you already **do** understand I2C signalling and wire protocol you can skip these first two references.

The SMBus Protocol Summary provides a convenient shorthand notation for I2C transactions (SMBus is based on the I2C wire protocol). This shorthand notation was adapted and augmented with notations specific to operation of the I2C controller used in NXP LPC microcontrollers. These notations illustrate the points in I2C transacitons where interrupts occur and the specific I2C status code associated with each interrupt.

The LPC2103 User Manual provides the detailed technical information needed for code development (including the embedded I2C controllers).

I2C Transaction Summary

It is assumed that the reader understands that data transfer (master-to-slave, or slave-to-master) via I2C is accomplished by combining (compositing) sequences of low-level I2C wire protocol (such as send byte or receive byte) to construct higher-level I2C transactions (such as read byte or write byte).

A subset of the SMBus protocol transactions are included in this document. This subset is sufficient for most I2C applications. If it is not, then this document in combination with the SMus Protocol Summary should provide a useful reference for implementation of more complex I2C transactions (such as block reads and writes).

NOTE:
this document currently does not include comments on the code needed to implement I2C on LPC21xx microcontrollers, nor does it address exception conditions that may occur during I2C transactions. These details may be added in a subsequent revision.

NOTE:
This document currently does not detail exception conditions that may occur during I2C transactions.

Symbol Key

```
S     (1 bit) : Start bit

Sr    (1 bit) : Repeated start bit

P     (1 bit) : Stop bit

R     (1 bit) : Read bit (bit 0 of address byte)
                - Read = 1

W     (1 bit) : Write bit (bit 0 of address byte)
                - Write = 0

D     (1 bit) : Data bit (bit 0 of address byte)
                - Single data bit (used in Quick Command)

Addr  (7 bits): 7-bit I2C address (bits [7:1] of address byte)
                - Note that this can be extended for 10-bit I2C addresses

+             : Cocatenation of Addr and R, W, or D bit
                - Addr+R = ( (7-bit address) << 1 ) + read bit
                - Addr+W = ( (7-bit address) << 1 ) + write bit
                - Addr+D = ( (7-bit address) << 1 ) + data bit

A     (1 bit) : Acknowledge, or Not-Acknowledge bit
                - Indicates Acknowledge when driven low (0)

NA     (1 bit) : Not Acknowledge bit
                - Indicates Not-Acknowledge when high (1)

Comm  (8 bits): Command byte
                - Typically used to select (address) a register in the slave

Data  (8 bits): Data byte

^             : Marks occurance of an LPC21xx I2C interrupt

0xSS          : Status code reported for the LPC21xx I2C interrupt
                - Status code = 0xSS (one status byte in hex format)
```

Important Note: Brackets [ ] are used to indicate bits (or bytes) that are driven by the I2C slave. Data driven by the I2C master does not have brackets.

Quick Command

Quick Command sends a single data bit from the master to the slave device (i.e., a one-bit write). Quick Command **does not** include a command byte (typically used to select a specific register in the slave device).

The shorthand notation for a Quick Command is:

```
S Addr+D [A] P
```

NOTE: The "D" field above represents the single data bit to be written to the device. In other I2C transactions this bit is used to indicate that one or more read (R) or write (W) bytes follow the address byte.

Send Byte

Send Byte writes a single byte of data to a slave device. Send byte **does not** include a command byte (typically used to select a specific register in the slave device).

```
S Addr+W [A] Data [A] P
```

Write Byte

Write Byte writes a single byte of data to a slave device. Write Byte **does** include a command byte (typically used to select a specific register in the slave device).

```
S Addr+W [A] Comm [A] Data [A] P
```

Write Word

Write Word writes two bytes of data to a slave device. Write Word **does** include a command byte (typically used to select a specific register in the slave device). The low byte of data is transferred first followed by the high byte of data.

```
S Addr+W [A] Comm [A] Data [A] Data [A] P
```

Receive Byte

Receive Byte reads a single data byte from a slave device. Receive byte **does not** include a command byte (typically used to select a specific register in the slave device).

```
S Addr+R [A] [Data] NA P
```

Read Byte

Read Byte reads a single data byte from a slave devices. Read byte **does** include a command byte (typically used to select a spsecific register in the slave device).

```
S Addr+W [A] Comm [A] Sr Addr+R [A] [Data] NA P
```

Read Word

Read Word reads two bytes of data from a slave device. Read Word **does** include a command byte (typcially used to select a specific register in the slave device). The low byte of data is transferred first followed by the high byte of data.

```
S Addr+W [A] Comm [A] Sr Addr+R [A] [Data] A [Data] NA P
```

LPC21xx Master Summary

The ^ symbols below show where interrupts occur during I2C transactions when the LPC21xx is the MASTER of the I2C transaction. The specific interrupt condition that can be read from the I2CSTAT register is noted in hex format immediately below the ^ symbol.

Quick Command

```
S Addr+D [A] P
 ^           ^
 0x08        0x18 if D = 0
             0x40 if D = 1
```

Send Byte

```
S Addr+W [A] Data [A] P
 ^           ^         ^
 0x08        0x18      0x28
```

Write Byte

```
S Addr+W [A] Comm [A] Data [A] P
 ^           ^         ^         ^
 0x08        0x18      0x28      0x28
```

Write Word

```
S Addr+W [A] Comm [A] Data [A] Data [A] P
 ^           ^         ^         ^         ^
 0x08        0x18      0x28      0x28      0x28
```

Receive Byte

```
S Addr+R [A] [Data] NA P
 ^           ^          ^
 0x08        0x40       0x58
```

Read Byte

```
S Addr+W [A] Comm [A]     Sr Addr+R [A] [Data] NA P
 ^           ^         ^      ^           ^          ^
 0x08        0x18      0x28   0x10        0x40       0x58
```

Read Word

```
S Addr+W [A] Comm [A]     Sr Addr+R [A] [Data] A [Data] NA P
 ^           ^         ^      ^           ^          ^        ^
 0x08        0x18      0x28   0x10        0x40       0x50     0x58
```

LPC21xx Slave Summary

The ^ symbols below show where interrupts occur during I2C transactions when the LPC21xx is the SLAVE of the I2C transaction. The specific interrupt condition that can be read from the I2CSTAT register is noted in hex format immediately below the ^ symbol.

Quick Command

```
S Addr+D [A]              P
            ^              ^
            0x60 if D = 0  0xA0
            0xA8 if D = 1
```

Send Byte

```
S Addr+W [A] Data [A]     P
            ^        ^     ^
            0x60     0x80  0xA0
```

Write Byte

```
S Addr+W [A] Comm [A] Data [A]     P
            ^        ^        ^     ^
            0x60     0x80     0x80  0xA0
```

Write Word

```
S Addr W [A] Comm [A] Data [A] Data [A]     P
            ^        ^        ^        ^     ^
            0x60     0x80     0x80     0x80  0xA0
```

Receive Byte

```
S Addr+R [A] [Data] NA P
            ^         ^
            0xA8      0xC0
```

Read Byte

```
S Addr+W [A] Comm [A]     Sr Addr+R [A] [Data] NA P
             ^         ^      ^           ^          ^
             0xA8      0x80   0x10        0xA8       0xC0
```

Read Word

```
S Addr+W [A] Comm [A]     Sr Addr+R [A] [Data] A [Data] NA P
             ^         ^      ^           ^          ^          ^
             0xA8      0x80   0x10        0xA8       0xA8       0xC0
```