AN10900

FM+ I2C on LPC1300

Rev. 01 — 17 December 2009 Application note

Document information

Info	Content
Keywords	LPC1300, FM+, Fast-mode Plus, I^2C, Cortex-M3
Abstract	This application note introduces how to use the Fast-mode Plus I^2C of LPC1300 with demo code and an I^2C demo board. NXP PCF9674 and PCA9632 are used as FM+ slave examples. I^2C high speed design concerns are also discussed.

Revision history

Rev	Date	Description
01	20091217	Initial version.

Contact information

For additional information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

1. Introduction

The LPC1300 is a family of ARM Cortex-M3 based microcontrollers for embedded applications featuring a high level of integration and low power consumption. Each microcontroller contains an I^2C-bus interface.

This I^2C-bus interface supports the full I^2C-bus specification and Fast-mode Plus with a data rate of up to 1 Mbit/s. Also supported are multiple address recognition and monitor mode. In Fast-mode Plus, rates from 400 kHz to 1 MHz may be selected.

This document will give an example of how to implement Fast-mode Plus on the LPC1300 family of parts and describes I^2C high-speed design concerns.

2. How to implement FM+ on LPC1300

2.1 LPC1300 I^2C Fast-mode Plus

Fast-mode Plus supports a 1 Mbit/sec transfer rate to communicate with the I^2C-bus products that NXP Semiconductors is now providing.

PIO0_4 and PIO0_5 are the pins used for the I^2C interface. In order to use Fast-mode Plus, the I2C pins must be properly configured in the IOCONFIG register block as shown in Table 1 and Table 2.

Table 1. IOCON_PIO0_4 register (IOCON_PIO0_4 address 0x4004 4030) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function	000
		000	Selects function PIO0_4	
		001	Selects I^2C function SCL	
		010 to 111	Reserved	
7:3		-	Reserved	00000
9:8	I2CMODE		Selects I^2C mode	00
		00	Standard mode/ Fast-mode I^2C	
		01	Fast-mode Plus I^2C	
		11	Reserved	
31:10	-	-	Reserved	-

Table 2. IOCON_PIO0_5 register (IOCON_PIO0_5 address 0x4004 4034) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function	000
		000	Selects function PIO0_5	

Bit	Symbol	Value	Description	Reset value
		001	Selects I^2C function SDA	
		010 to 111	Reserved	
7:3		-	Reserved	00000
9:8	I2CMODE		Selects I^2C mode	00
		00	Standard mode/ Fast-mode I^2C	
		01	Fast-mode Plus I^2C	
		11	Reserved	
31:10	-	-	Reserved	-

For Fast-mode Plus, FUNC(Bit 2:0) should be 001 and I2CMODE(Bit 9:8) should be 01.

After configuring Fast-mode Plus, the I^2C pins are configured with the input glitch filter enabled (this includes an open-drain output according to the I^2C-bus specification). In this mode, the pins function as high-current sinks.

2.2 Example

The demo has been developed with Keil's MDK 3.70 environment and tested on an NXP LPC1300 board and an I^2C slave board. An IAR project based on EWARM 5.40 is also provided.

2.2.1 Hardware

NXP's LPC1300 and an I^2C FM+ demo board consist of the hardware environment. A 30cm I^2C cable connects LPC1300 and I^2C FM+ demo board.

The I^2C FM+ board schematic is shown in Fig 1.

The NXP PCA9674 and PCA9632 were selected as the Fast-mode Plus I^2C slaves.

The PCA9674 is an NXP Remote 8-bit I/O expander for FM+ I^2C-bus. In this example, P0~P3 are configured as inputs and P4~P7 are configured as outputs to drive LEDs.

The PCA9632 is an NXP 4-bit FM+ I^2C-bus low power LED driver. In this example, the PCA9632 is used to drive four LEDs. The LED blink frequency is changed to show whether the I^2C-bus is in FM+ mode or standard I^2C mode.

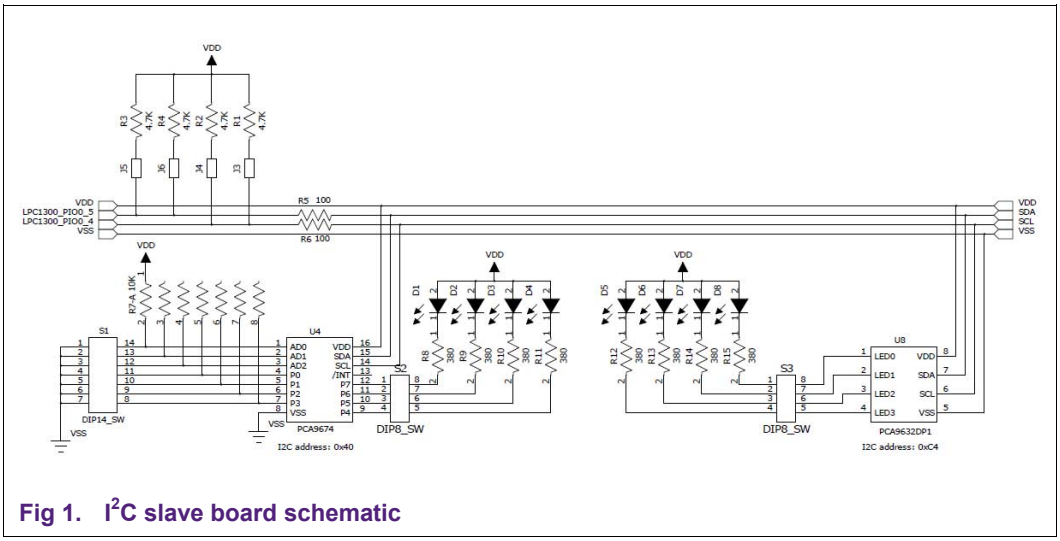

Fig 1. I²C slave board schematic

S1.1~S1.3 are the address selectors for the PCA9674. Setting all switches to ON will configure the PCA9674 address as 0x40.

S1.4 is the mode select switch. Switch it ON to configure the I²C-bus as FM+ mode and switch it OFF to configure the I²C-bus to standard I²C mode. D1~D4 are indicators for S1.4~S1.7.

LED D5~D8 are driven by the PCA9632 and are used to show the I²C mode via blinking frequency.

J3~J6 are jumpers to configure the I2C pull up resisters.

2.2.2 Software

The example software is designed to demonstrate Fast-mode Plus I²C using the LEDs. Two source code files are provided, I2C.c and I2Ctest.c.

2.2.2.1 I2C.c

I2C.c contains the I²C related functions which are called by the main routine.

Table 3. I2C.c functions

Functions	Description
void I2C_IRQHandler(void)	I²C interrupt handler- used for master mode only.
uint32_t I2CStart(void)	Create an I²C start condition.
uint32_t I2CStop(void)	Set the I²C stop condition.
uint32_t I2CInit(uint32_t I2cMode)	Initialize the I²C controller.
uint32_t I2CEngine(void)	Complete an I²C transaction from start to stop.

2.2.2.2 I2Ctest.c

I2Ctest.c is the main routine to read and write the I²C date/command to the slaves:

```
1    while(1)
2    {
3      /* Write PCA9674 0xFF to config IOs as input and read IOs back */
4      /* Write SLA(W), data(0xFF), SLA(R), and read 2 byte back. */
```

```
5       I2CWriteLength = 2;
6       I2CReadLength = 2;
7       I2CMasterBuffer[0] = PCA9674_ADDR;
8       I2CMasterBuffer[1] = 0xFF;
9       I2CMasterBuffer[2] = PCA9674_ADDR | RD_BIT;
10      I2CEngine();
11
12      /* PCA9674 P0 as switch for FM+(1Mbps) and standard/FM(400Kbps) I2C */
13      if (I2CSlaveBuffer[0] & 0x01)
14      {
15          /* I2C FM+ mode */
16          LPC_IOCON->PIO0_4      &= ~0x0000031F;   /*  I2C FM+ I/O config */
17          LPC_IOCON->PIO0_4      |= 0x00000101;    /* I2C FM+ SCL */
18          LPC_IOCON->PIO0_5      &= ~0x0000031F;
19          LPC_IOCON->PIO0_5      |= 0x00000101;    /* I2C FM+ SDA */
20          /*--- Reset Baud Rate to 1Mbps ---*/
21          LPC_I2C->SCLL   = 48;//24;//0x18;
22          LPC_I2C->SCLH   = 48;//24;//0x18;
23      }
24      else
25      {
26          /* I2C standard/FM */
27          LPC_IOCON->PIO0_4      &= ~0x1F;       /*  I2C I/O config */
28          LPC_IOCON->PIO0_4      |= 0x01;        /* I2C SCL */
29          LPC_IOCON->PIO0_5      &= ~0x1F;
30          LPC_IOCON->PIO0_5      |= 0x01;        /* I2C SDA */
31          /*--- Reset Baud Rate to 62.5Kbps ---*/
32          LPC_I2C->SCLL   = 96;//192;//384;//2400;//0x180;
33          LPC_I2C->SCLH   = 96;//192;//384;//2400;//0x180;
34      }
35
36      /* Write SLA(W), data1 */
37      I2CWriteLength = 2;
38      I2CReadLength = 0;
39      I2CMasterBuffer[0] = PCA9674_ADDR;
40      I2CMasterBuffer[1] = (I2CSlaveBuffer[0] << 4) | 0x0F;
41      I2CEngine();
42
43      /* Write PCA9632 config parameters for LED dimming */
44      /* Write SLA(W), CON, 00H ~ 0Ch */
45      I2CWriteLength = 6;
46      I2CReadLength = 0;
47      I2CMasterBuffer[0] = PCA9632_ADDR;
48      I2CMasterBuffer[1] = 0xC2;              // Control register
49      I2CMasterBuffer[2] = temp;              // PWM0
50      I2CMasterBuffer[3] = temp + 0x40;       // PWM1
51      I2CMasterBuffer[4] = temp + 0x80;       // PWM2
52      I2CMasterBuffer[5] = temp + 0xC0;       // PWM3
53
54      I2CEngine();
55
```

```
56      /* update the brightness */
57      temp1++;
58      temp = temp + (((temp1 % 2) == 0) ? 1 : 0);
59    }
```

2.2.3 Conditions and test results

On the I^2C FM+ demo board, we configure J3 and J5 closed, J4 and J6 open to select the pull up resistance as 4.7 KΩ.

Software can set values for the registers I2SCLH and I2SCLL to select the appropriate data rate and duty cycle. I2SCLH defines the number of PCLK_I2C cycles for the SCL high time; I2SCLL defines the number of PCLK_I2C cycles for the SCL low time. The frequency is determined by the below formula (PCLK_I2C is the frequency of the system clock):

$$I2Cbitfrequency = \frac{PCLKI2C}{I2SCLH + I2SCLL}$$

Software configures PCLKI2C as 48 MHz, and various I^2C speeds are generated by different I2SCLH AND I2SCLL configurations as shown in Table 4 below:

Table 4. I^2C speed configurations

Condition	I^2C bit frequency	I^2C mode	PCLKI2C	I2SCLH	I2SCLL
1	10 Kbps	Standard mode/Fast-mode I^2C	48 MHz	2400	2400
2	62.5 Kbps	Standard mode/Fast-mode I^2C	48 MHz	384	384
3	125 Kbps	Standard mode/Fast-mode I^2C	48 MHz	192	192
4	250 Kbps	Standard mode/Fast-mode I^2C	48 MHz	96	96
5	500 Kbps	Fast-mode Plus I^2C	48 MHz	48	48
6	1000 Kbps	Fast-mode Plus I^2C	48 MHz	24	24

2.2.3.1 Condition 1: I²C speed 10 Kbps, Standard mode/Fast-mode

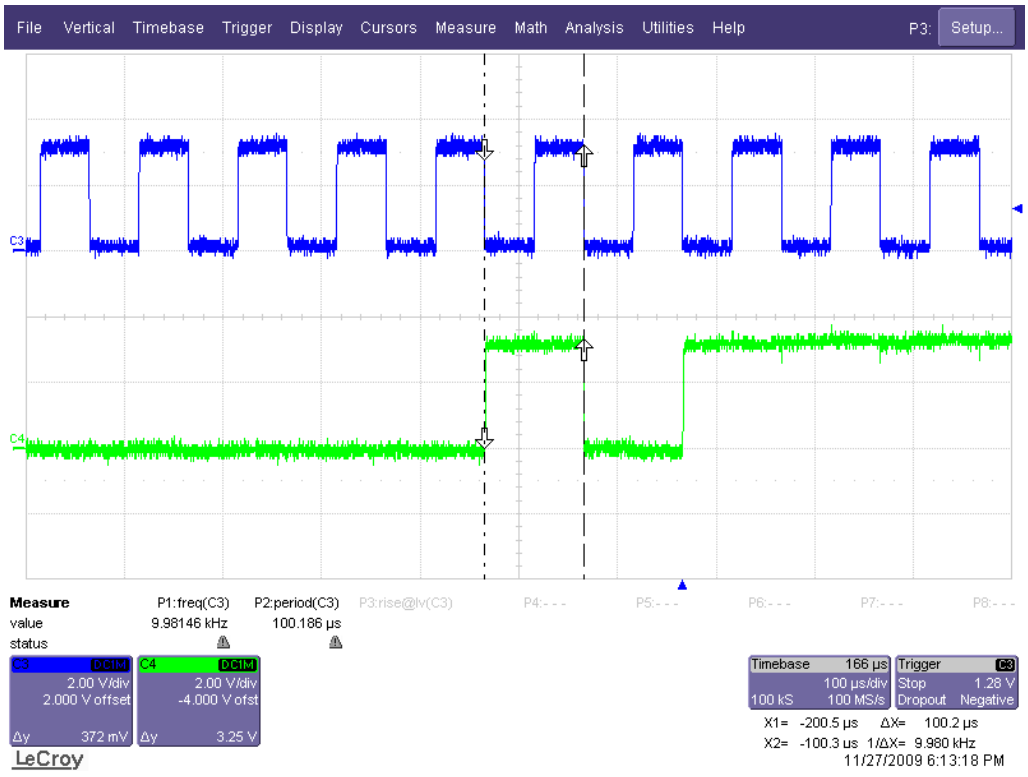

Fig 2. Condition 1: I²C speed 10 Kbps, Standard mode/Fast-mode

2.2.3.2 Condition 2: I²C speed 62.5 Kbps, Standard mode/Fast-mode

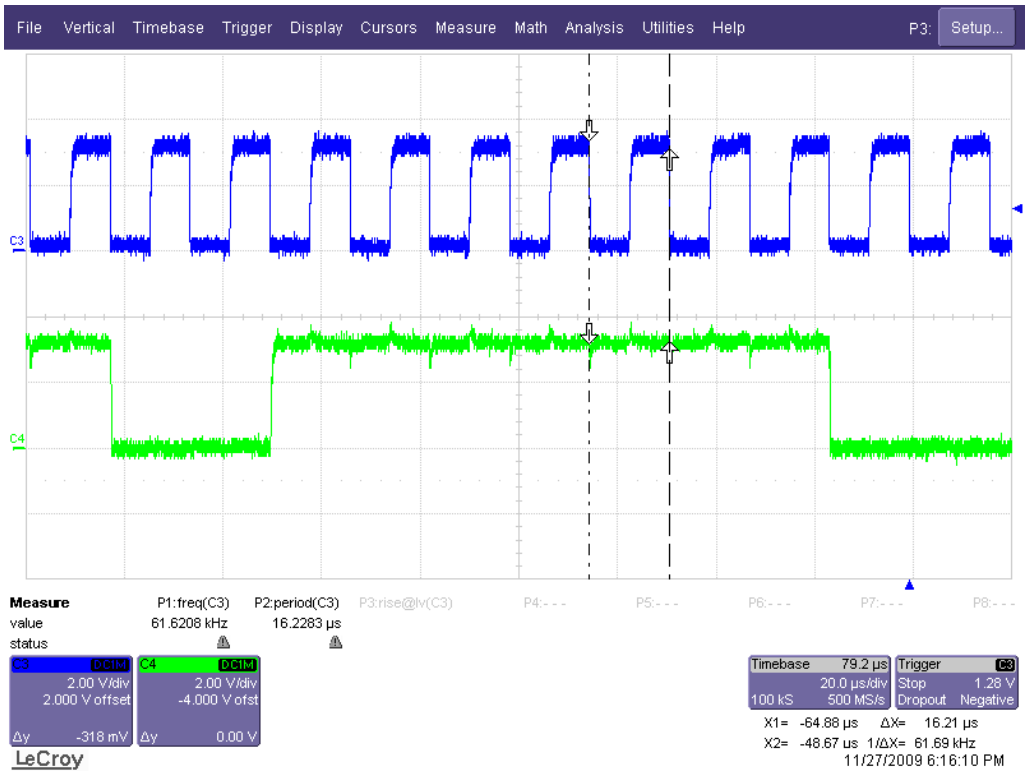

Fig 3. Condition 2: I²C speed 62.5 Kbps, Standard mode/Fast-mode

2.2.3.3 Condition 3: I²C speed 125 Kbps, Standard mode/Fast-mode

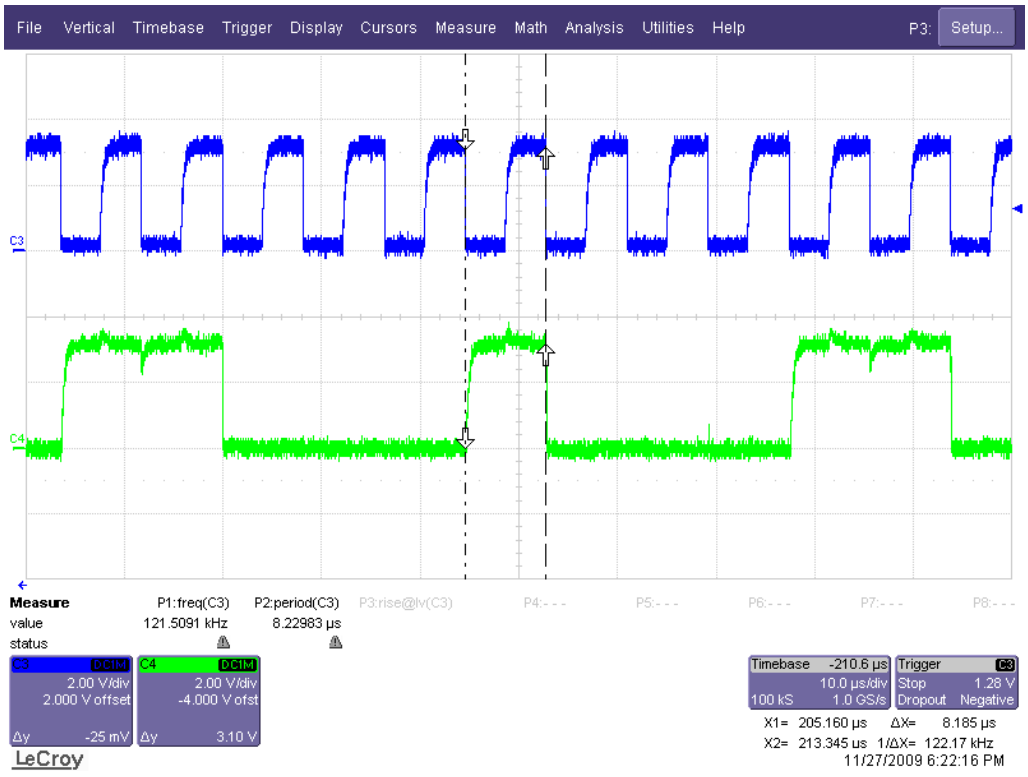

Fig 4. Condition 3: I²C speed 125 Kbps, Standard mode/Fast-mode

2.2.3.4 Condition 4: I²C speed 250 Kbps, Standard mode/Fast-mode

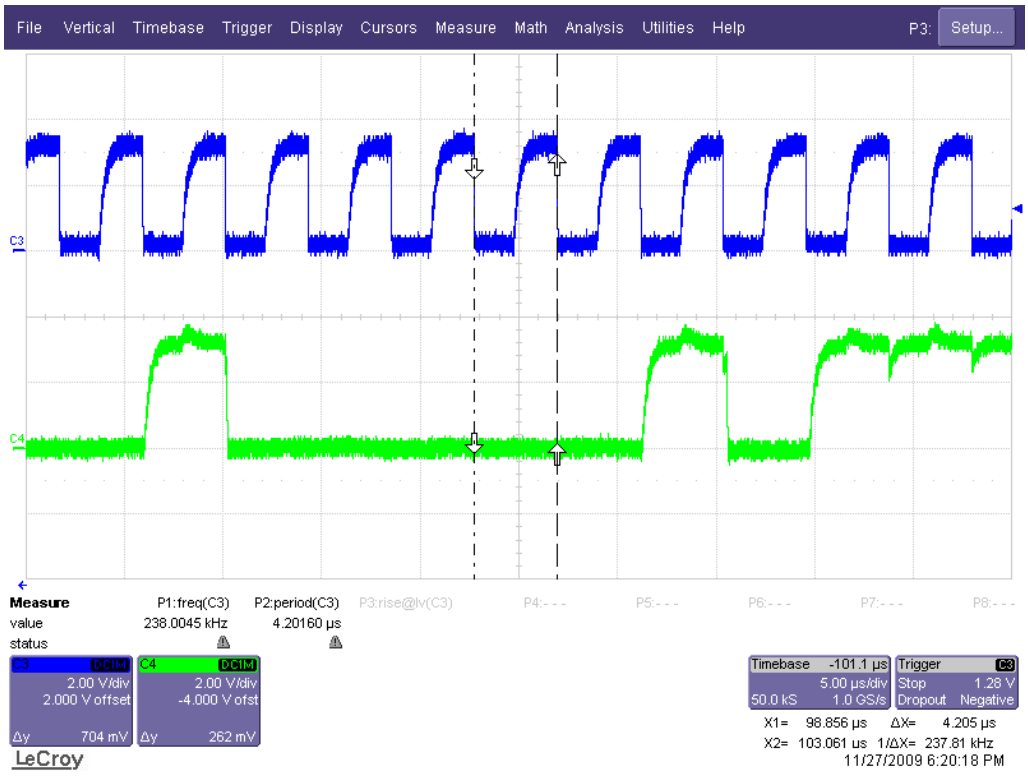

Fig 5. Condition 4: I²C speed 250 Kbps, Standard mode/Fast-mode

2.2.3.5 Condition 5: I²C speed 500 Kbps, Fast-mode Plus

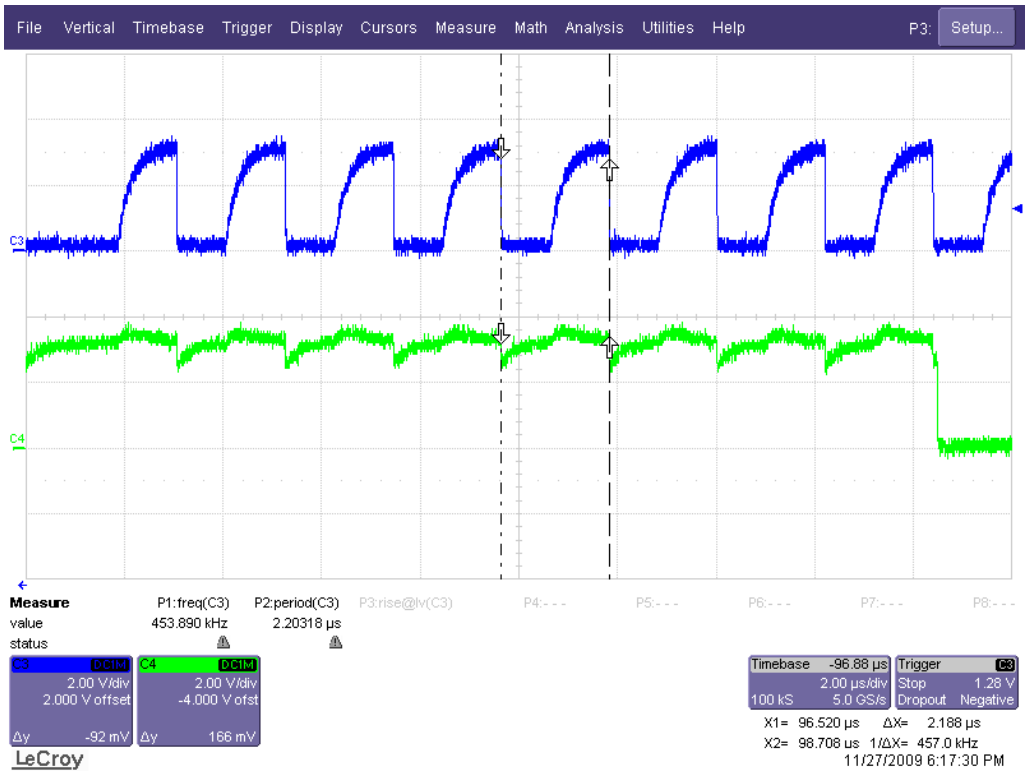

Fig 6. Condition 5: I²C speed 500 Kbps, Fast-mode Plus

2.2.3.6 Condition 6: I²C speed 1000 Kbps, Fast-mode Plus

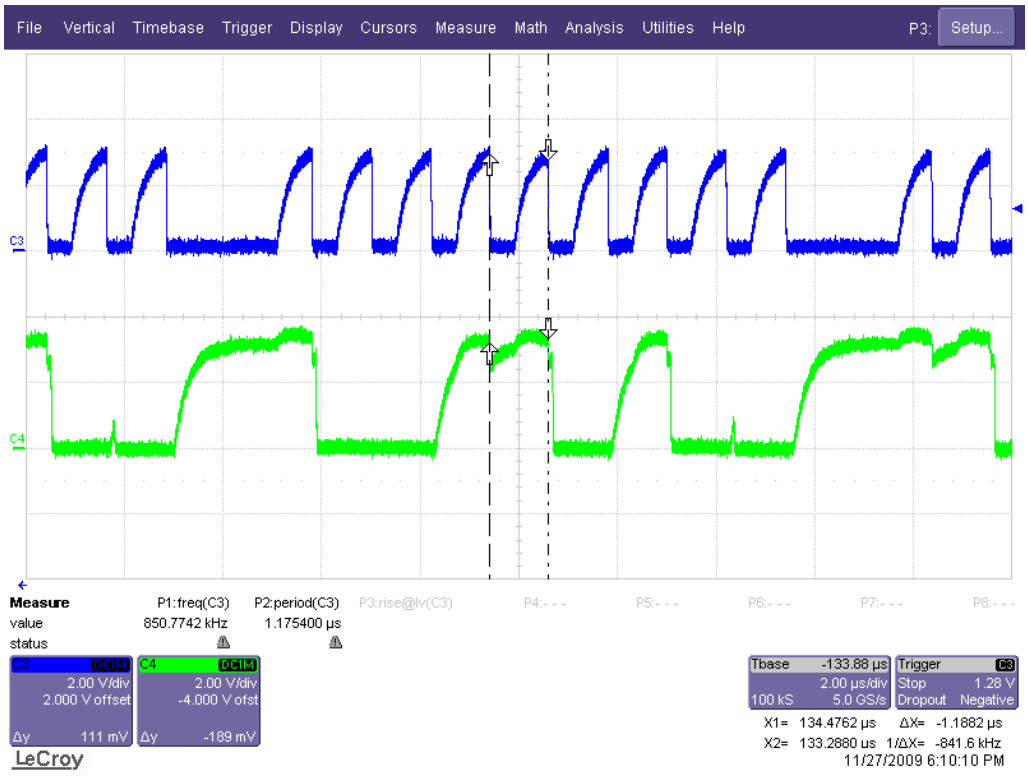

Fig 7. Condition 6: I²C speed 1000 Kbps, Fast-mode Plus

3. I²C baud rate error at high speed

3.1 Description

As shown in section 2.2.3 test results, the measured I²C speed does not match the configured speed. Table 5 shows the error in each condition.

Table 5. I²C baud rate error

I²C speed configured	I²C speed measured	Error
10 Kbps	9.98146 Kbps	−0.19 %
62.5 Kbps	61.6208 Kbps	−1.41 %
125 Kbps	121.5091 Kbps	−2.79 %
250 Kbps	238.0045 Kbps	−4.8 %
500 Kbps	453.890 Kbps	−9.22 %
1000 Kbps	850.7742 Kbps	−14.92 %

The error becomes larger as the I²C speed is increased. This is understood and explained here. The I²C master monitors its own outgoing clock edges, and each clock cycle cannot start until the SCL pin returns high. Since I²C pins are open-drain, the bus is pulled high by pull-up resistors. An I²C slave can implement flow control by holding the

clock SCL low. The master is designed to wait until the clock is released before sending the next clock.

Typically on a high-speed I^2C-bus you would see a sharp high-to-low transition and a slower low-to-high transition. The asymmetry is caused by using a pull-up resistor instead of an active driver to pull up the bus.

A per-clock delay can be caused by the inherent rise time seen on the bus. Slow rise time on the SCL pin is indistinguishable from slave clock stretching. The “electrical clock stretching” delay is constant relative to the speed of the bus, but variable relative to the capacitive loading on the SCL pin. This delay would increase the clock period by a constant amount. When the I^2C-bus is in a higher speed mode, this constant amount will take more share of the clock period, and make the I^2C speed lower than the speed configured.

3.2 Delay in 1000 Kbps mode

As an example, if in 1000 Kbps, the measured SCL frequency is 899.138 Kbps. There is a per-clock delay every SCL period.

So if the SCL period is counted from 70 % V_{DD}, as Fig 8 shows, the SCL frequency is more accurate, around 1000 Kbps.

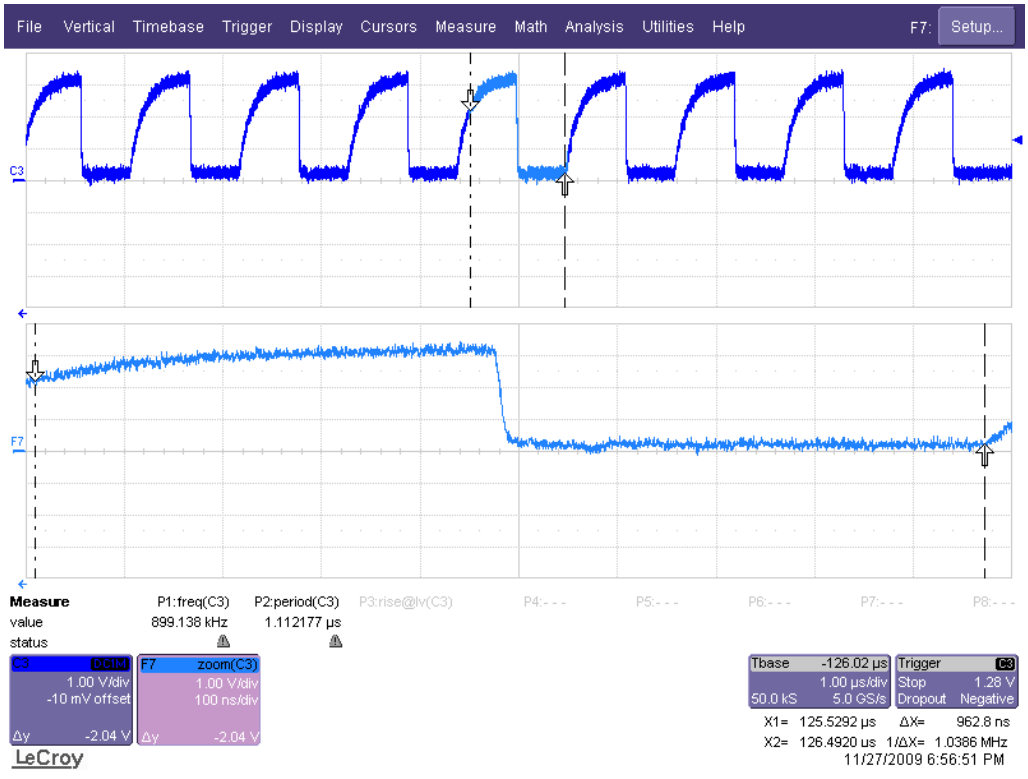

Fig 8. Delay in 1000 Kbps mode

In a real-world system this delay cannot be completely eliminated, but can be reduced.

Because this delay is mostly the charging time of the SCL, decreasing the pull up resister could significantly reduce the delay, thus making the I^2C speed closer to the designed speed.

4. Conclusion

Today, a single I^2C channel must support many diverse tasks simultaneously such as product authentication, EEPROM reads, temperature measurement, and power system margining. For this reason, performance has become critical. The LPC1300's integrated Fast-mode plus I^2C controller provides an easy way to improve I^2C-bus performance by more than 2x without adding cost or complexity. The only critical task is to make sure that the I^2C-bus meets the required electrical specs such as capacitance and includes proper termination. NXP's product line includes many FM+ capable I^2C devices besides the LPC1300 series microcontroller.

5. Legal information

5.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

5.2 Disclaimers

General — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in medical, military, aircraft, space or life support equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is for the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from national authorities.

5.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

I^2C-bus — is a trademark of NXP B.V.

6. Contents

For more information, please visit: http://www.nxp.com
For sales office addresses, email to: salesaddresses@nxp.com

Date of release: 17 December 2009
Document identifier: AN10900_1