NXP Semiconductors
Application Note

Document Number: AN5301
Rev. 0, 06/2016

Using LPI2C on KL28

1. Introduction

LPI2C is a brand new module. In this application note, the key features are introduced, and some key points in application are explained, including using LPI2C in both master and slave mode.

2. Key Features

The key features on LPI2C include:

- Standard, Fast, Fast+, HS, and Ultra-Fast modes are supported
- Separate I^2C master and slave registers to minimize software overhead due to master/slave switching
- Works with DMA in low power mode to maintain communication
- Master operation based on commands
- Slave operation with automatic timing match

Contents

1. Introduction 1
2. Key Features 1
3. About Standard, Fast, Fast+, HS and Ultra Fast Modes 2
4. LPI2C Block Diagram 3
5. LPI2C Master Operation 4
   5.1. Pin configuration 4
   5.2. Set baud rate 4
   5.3. Commands 5
   5.4. Work together with DMA 6
   5.5. Work together with DMA in low power mode 6
6. LPI2C Slave Operation 7
   6.1. Set slave address 7
   6.2. Configure stall 7
   6.3. A typical I2C communication implemented by interrupt mode 7
   6.4. Work together with DMA 8
   6.5. Work together with DMA in low power mode 8
7. Revision History 10

3. About Standard, Fast, Fast+, HS and Ultra Fast Modes

A definition of Standard, Fast, Fast+, HS, and Ultra-Fast modes can be found in the I^2C-bus specification and user manual, Rev. 4. The speed and direction supported are shown in *Table 1*.

Table 1. I^2C bus speed and direction in different mode

Mode	Bit rate up to	Direction
Standard-mode (Sm)	100 kbit/s	Bidirectional
Fast-mode (Fm)	400 kbit/s	Bidirectional
Fast-mode Plus (Fm+)	1 Mbit/s	Bidirectional
High-speed mode (Hs-mode)	3.4 Mbit/s	Bidirectional
Ultra Fast-mode (UFm)	5 Mbit/s	Unidirectional

HS mode devices can transfer information at bit rates of up to 3.4 Mbit/s, yet they remain fully downward compatible with Fast-mode Plus, Fast- or Standard-mode (F/S) devices for bidirectional communication in a mixed-speed bus system.

In the mixed-speed bus system, the data transfer process function is shown in the following image.

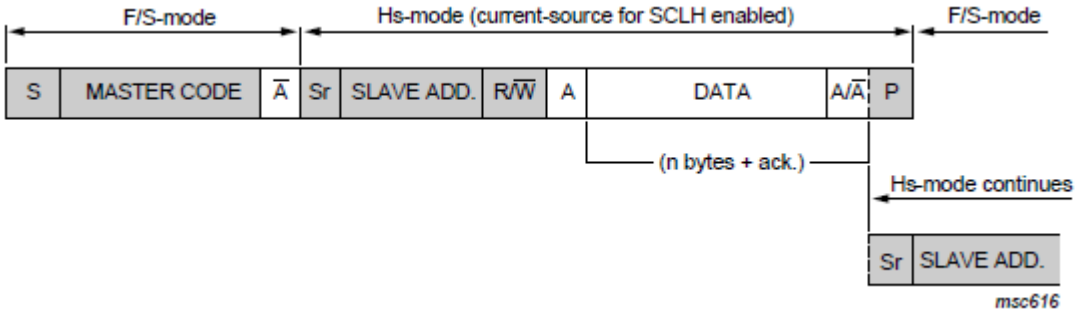

Figure 1. Data transfer format in Hs-mode

During the START+MASTER CODE+ACK transfer, it is Fast/Standard mode compatible. It then goes into HS mode, until a STOP is generated.

The master code must be 0000 1xxx according to the definition in the I^2C bus specification.

4. LPI2C Block Diagram

The LPI2C block diagram is shown in *Figure 2*.

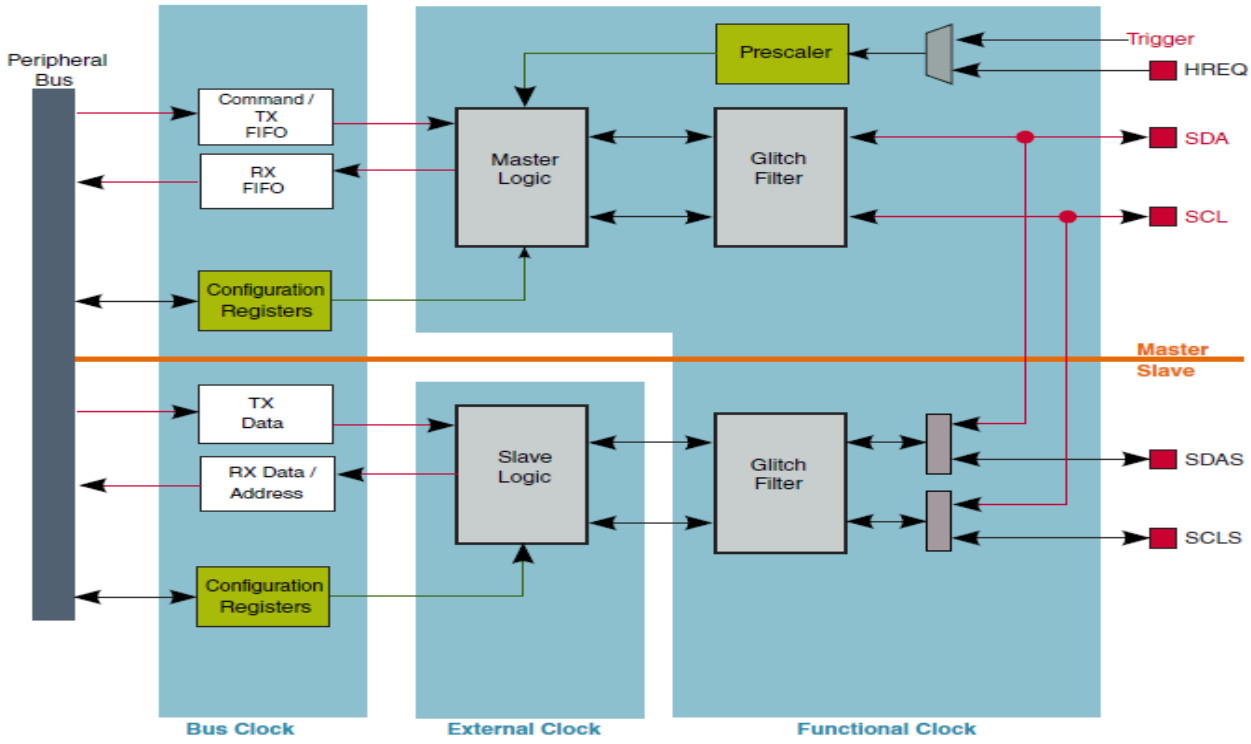

Figure 2. LPI2C block diagram

In this image, the clock domain is clearly visible. The bus clock is DIVSLOW_CLK from the SCG module. The functional clock is the output from the PCC module. The external clock is the clock provided by external I^2C master.

5. LPI2C Master Operation

5.1. Pin configuration

To change to a different I^2C mode, it is necessary to set the correct pin configuration. This is shown in *Table 2.*

Table 2. Pin configuration for different mode

Mode	Pin configuration
Standard-mode (Sm)	Open drain mode
Fast-mode (Fm)	Open drain mode
Fast-mode Plus (Fm+)	Open drain mode
High-speed mode (Hs-mode)	Push-pull mode
Ultra Fast-mode (UFm)	Output only mode

Pin configuration is set by PINCFG in LPI2C_MCFGR1[PINCFG].

5.2. Set baud rate

For master operation, the formula to set the baud rate is:

Baud rate divide = ((CLKLO+CLKHI+2)*2^PRESCALER) +

ROUNDDOWN((2+FILTSCL)/2^PRESCALER)

CLKLO and CLKHI is in LPI2C_MCCRx, PRESCALER is in LPI2C_MCFGR1, FILTSCL is in LPI2C_MCFGR2.

The following is an example to set the baud rate:

- The LPI2C function clock comes from the FIRC, according to scope the value is 47.17 MHz
- Set the PRESCALER to 2, set CLKLO to 12, and set CLKHI to 12
- The baud rate can then be estimated by: 47.17 MHz / 4 / (12+12+2) = 453 KHz
- The actual baud rate from the scope is: 438.6 KHz

In this example it is clear that the actual measured baud rate is lower than estimated. If the baud rate is set higher, for example 1 MHz, the resulting difference will be even larger. This is to be expected, as the rising edge is slow and it would introduce more cycles here. In the KL28ZRM this is described as follow:

This assumes SCL will pull high within 1 cycle of the LPI2C functional clock, this will depend on the pullup resistor and loading on the SCL pin.

LPI2C supports two configurations by MCCR0 and MCCR1. MCCR1 configuration is used for HS mode transfers. This is useful as the HS-mode transfer is a mixed-speed system.

For the LPI2C slave, as it is working with an external clock domain, the baud rate it supports is decided by an external input which should match the I^2C bus specification.

5.3. Commands

The LPI2C master operation is based on command. This design make it easy to use and work together with DMA.

The command list is shown in the following image:

Command Data	
000	Transmit DATA[7:0].
001	Receive (DATA[7:0] + 1) bytes.
010	Generate STOP condition.
011	Receive and discard (DATA[7:0] + 1) bytes.
100	Generate (repeated) START and transmit address in DATA[7:0].
101	Generate (repeated) START and transmit address in DATA[7:0]. This transfer expects a NACK to be returned.
110	Generate (repeated) START and transmit address in DATA[7:0] using high speed mode.
111	Generate (repeated) START and transmit address in DATA[7:0] using high speed mode. This transfer expects a NACK to be returned.

Figure 3. LPI2C commands

Due to the commands, the process is simplified. *Figure 4* show a quick start demo.

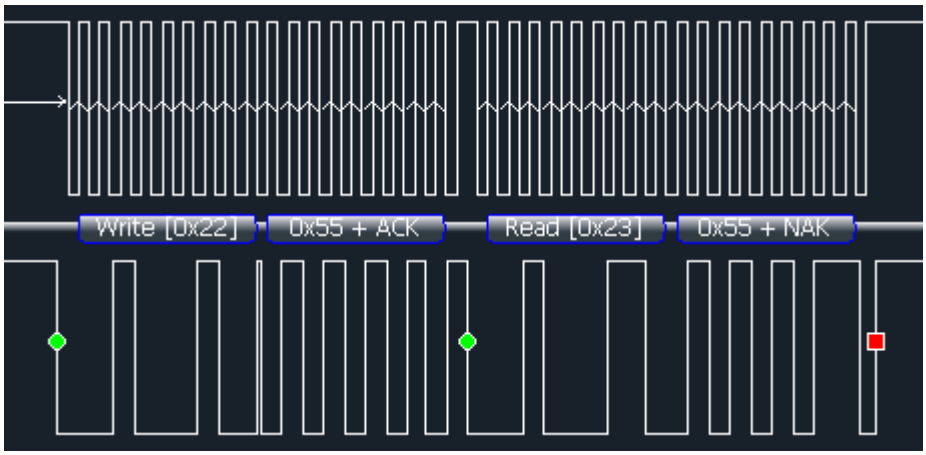

Figure 4. A quick start demo

To implement the timing in *Figure 4*, the following command array is necessary:

- Command 1: Generate START and transmit address (for writing)
- Command 2: Transmit DATA, 1 byte
- Command 3: Generate START and transmit address (for reading)
- Command 4: Receive DATA, 1 byte
- Command 5: Generate STOP

5.4. Work together with DMA

In *section 5.3*, the command list can be seen as a data array, DMA is then configured to send the array to LPI2C_MTDR. Another DMA channel is then used to receive the data. The following figure shows a typical command array to access the accelerometer FXOS8700CQ.

```
static unsigned int cmd_array[] =
{
    (LPI2C_CMD_START_SEND<<8) | (ACC_8700_ADDR<<1) | I2C_WRITE,
    (LPI2C_CMD_TX<<8)         | DATA_OFFSET,
    (LPI2C_CMD_START_SEND<<8) | (ACC_8700_ADDR<<1) | I2C_READ,
    (LPI2C_CMD_RX<<8)         | 5,
    (LPI2C_CMD_STOP<<8)       | 0
};
```

Figure 5. A typical command array

To make LPL2C work with DMA, the following steps are necessary:

- LPI2C_MDER[TDDE] and LPI2C_MDER[RDDE] should be set
- DMA MUX for TX should be set for LPI2C TX
- DMA MUX for RX should be set for LPI2C RX
- DMA_TCD_CSR[DREQ] should be set to disable DMA after transfer

5.5. Work together with DMA in low power mode

Another key feature supported on LPI2C is that it can work with DMA in lower mode. VLPS mode is usually used here. This means that I^2C communication can be maintained and go into low power mode to reduce power consumption.

To implement this application, the following steps are necessary:

- Initialize LPI2C with LPI2C_MCR[DOZEN] enabled
- Initialize DMA with DMA_EARS set for the channel used to enable asynchronous DMA request
- Enable DMA
- Go to VLPS mode
- Wake up after LPI2C transfer ends by DMA interrupt

6. LPI2C Slave Operation

NOTE

Pay attention to ensure that the LPI2C master and slave logic are totally independent, and there are two sets of control register for each mode.

6.1. Set slave address

When LPI2C is working in slave mode it supports two addresses. These can be configured in LPI2C_SAMR.

6.2. Configure stall

When LPI2C is working in slave mode it supports clock stretching. There are four types of stall in LPI2C:

- ACK STALL
- TX STALL
- RX STALL
- ADDRESS STALL

There are two typical configurations here for an application:

Configuration 1: ACK STALL + TX STALL

Configuration 2: TX STALL + RX STALL + ADDRESS STALL

For Configuration 1, ACK/NACK is given after a byte is received. The slave can decide to send ACK or NACK according to the data it received.

For Configuration 2, it stays in STALL mode until the related status flags are processed.

The user can choose one configuration according to the application requirement.

6.3. A typical I2C communication implemented by interrupt mode

This section will explain how to enable the LPI2C slave to match the timing showed in the following figure:

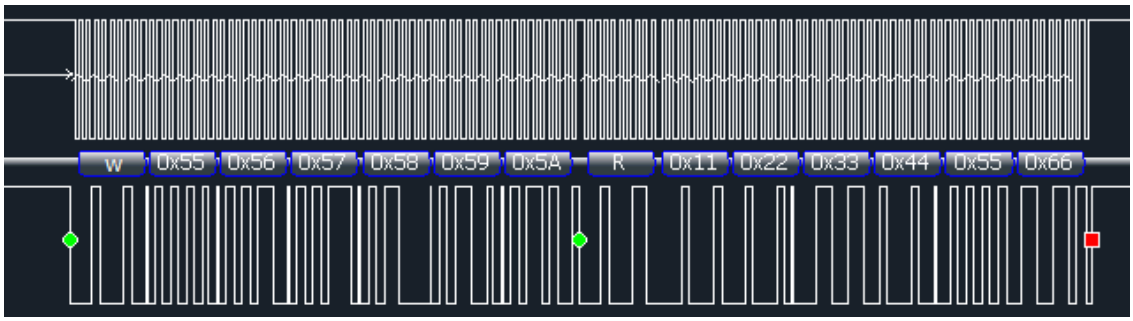

Figure 6. Implement of a typical LPI2C application

In this communication, the LPI2C slave receives 6 bytes 0x55-0x5A, and sends out 6 bytes 0x11-0x66.

To implement this, the following steps must be carried out:

- Enable stall for TX, RX and ADDR match
- Enable interrupt for TX, RX and ADDR match
- Receive a byte when SSR_RDF is set
- Send a byte when SSR_TDF is set

As LPI2C is user-friendly, less than 20 code lines in the LPI2C ISR are needed to implement the timing in *Figure 6*.

In this use-case, LPI2C_SSR[TDF] is set after ADDR+R occurs, which means that this bit is matching with the I^2C bus timing.

6.4. Work together with DMA

The slave works in half-duplex mode, as in LPI2C. Only one DMA channel is necessary.

To implement the application, we still need to respond to the interrupt caused by ADDR+W and ADDR+R. The slave must then process the data from the master, this enables it to know what kind of data to send back to the master.

DMA initialization must be done in ISR for ADDR + W and ADDR + R. For the steps, refer to *section 5.4*.

6.5. Work together with DMA in low power mode

Like the LPI2C master, the LPI2C slave can also work with DMA and stay in low power mode to reduce power consumption.

To implement this application, the steps are:

- Initialize LPI2C with LPI2C_SCR[FILTDZ] enabled
- Initialize DMA with DMA_EARS set for the channel used
- Enable DMA
- Go to VLPS mode
- Wake up after the ADDR + W interrupt, configure DMA for RX
- Wake up after the ADDR + R interrupt, prepare the data to be sent to master and configure DMA for TX

Figure 7 shows the LPI2C low power application in slave mode, with 4.9 uA current in VLPS mode. The LPI2C slave can respond to the master with DMA to make the timing showed in *Figure 6*.

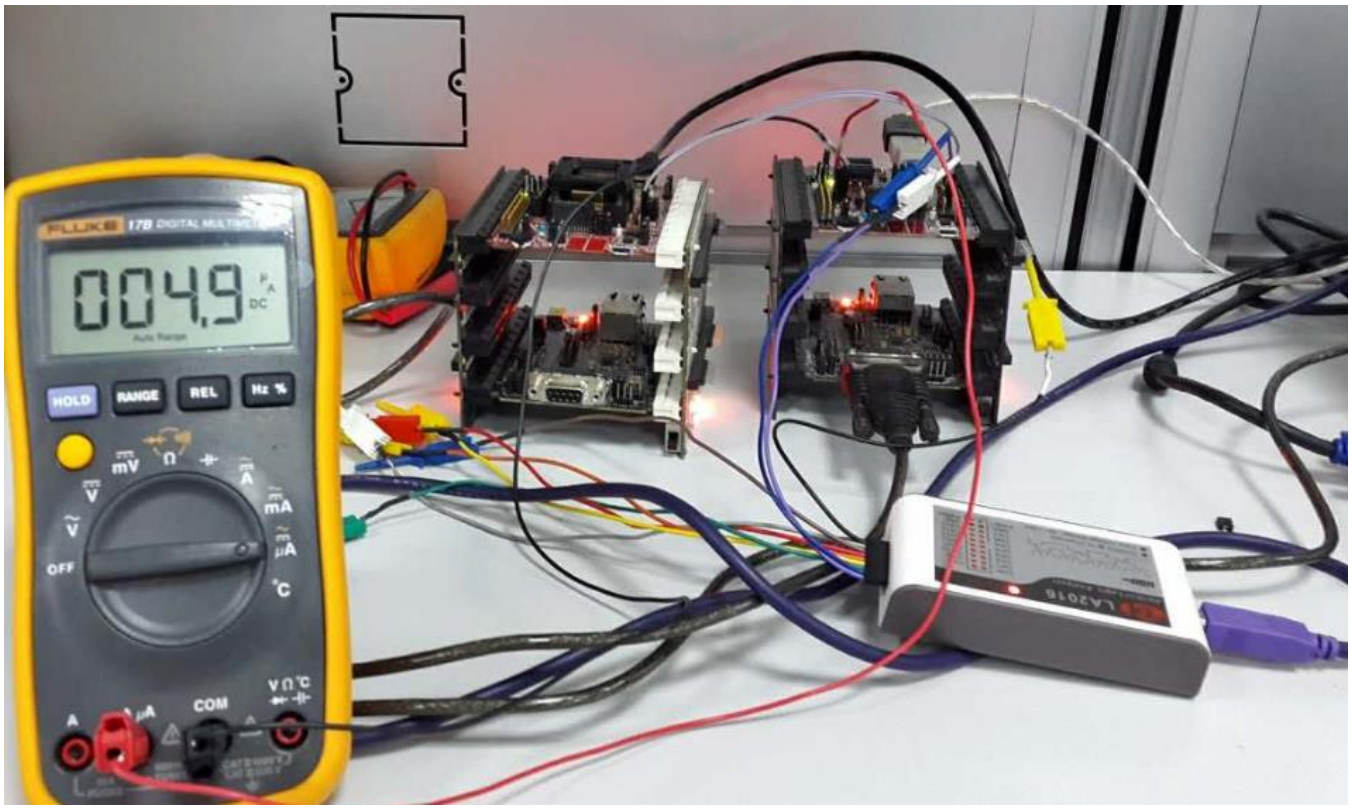

Figure 7. Implement of low power application on LPI2C in slave mode

In this application the current must be optimized. If it is not then it is not possible to get current as low as 4.9uA. The following steps help with optimization:

- Do not open the clock for the peripheral not used
- Do not configure any IO pins not used
- Disable the pull-up on PTA3/SWDIO, this would reduce the current from 70uA to 4.9uA on the KL28Z-TWR board

7. Revision History

Table 3. Revision history

Revision number	Date	Substantive changes
0	06/2016	Initial release

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number: AN5301
Rev. 0
06/2016