AN13472

LPI2C Clock Stretching in RT1010

Rev. 0 — 3 December 2021 Application Note

Contents

1 Introduction

I^2C communication is popular in most MCU applications. i.MXRT series MCU provides strong features in LPI2C module. The clock stretching function is useful in the application. RT1010 provides four types of clock stretching functions. This application note introduces how to use the function and key points when using the function.

The hardware is based on RT1010 EVK (Rev. C) and the software is based on SDK2.10.0.

The path of this demo is:

SDK_2_10_0_EVK-MIMXRT1010\boards\evkmimxrt1010\driver_examples\lpi2c\interrupt_b2b_transfer

2 LPI2C clock stretching overview

The clock stretching function pauses the data transmission by holding the SCL line to LOW level. The transmission cannot continue until the line is released to HIGH level again.

For byte level, the device is able to receive data bytes with a fast rate. However, it takes more time to store a received byte or prepare another byte to be transmitted. After a byte is received and acknowledged, the slave device (described as **Slave** below) holds the SCL line to the LOW state. It forces the master device (described as **Master** below) into a **Wait** state until Slave is ready for the next byte transmission in the handshake procedure.

For the bit level, the device, such as a microcontroller with or without limited hardware for the I2C-bus, can slow down the bus clock by extending the LOW period. The speed of any Master is adapted to its internal operating rate.

NOTE

Before using the clock stretching function, read the manual to check whether the device supports the clock stretching function.

- Not all I^2C Slaves support the clock stretching function. For example, some sensors and memory devices do not support the function.
- Not all I^2C Masters support the clock stretching function. For example, the device, with I^2C simulated by GPIO peripheral or an I^2C peripheral in FPGA, does not support the function.

3 LPI2C clock stretching in RT1010

i.MX RT1010 Processor Reference Manual (document IMXRT1010RM) lists four types of clock stretching function, as shown in Figure 1.

- During the 9th clock pulse of the address byte and the Address Valid flag is set.
- During the 9th clock pulse of a slave-transmit transfer and the Transmit Data flag is set.
- During the 9th clock pulse of a slave-receive transfer and the Receive Data flag is set.
- During the 8th clock pulse of an address byte or a slave-receive transfer and the Transmit ACK flag is set. In high speed mode, this is disabled.

Figure 1. Clock stretching function supported by RT1010

In my view, the clock stretching function can be described as below:

> Using **Following the Nth clock pulse** instead of **During the Nth clock pulse**.

3.1 Enabling clock stretching after receiving address

When Slave receives an address from Master and AVF bit is set, Salve holds the SCL to the LOW state and starts the clock stretching process. Once AVF bit is cleared, the clock stretching process ends and Master controls the SCL state. Figure 2 shows the waveform that Slave receives an address and clears the AVF bit after 500 µs delay.

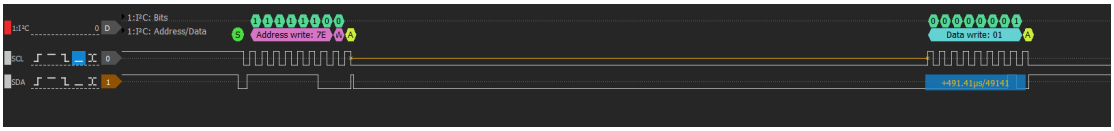

Figure 2. Enabling clock stretching after receiving address

As shown in Figure 2, the time interval, from the 9th pulse of address to the 1st pulse of data information, is 491 µs. The result is approximate to 500 µs because the delay API does not use a timer.

3.2 Enabling clock stretching before sending data

When Slave receives the address and the read command, the TDF bit is set. At the same time, the clock stretching process starts and Slave holds the SCL. During the process, Slave prepares the sent data according to the information from Master. Once the data is ready, the TDF bit is cleared and the clock stretching process ends. Master controls SCL again. For this time, the delay before clearing TDF bit is 800 µs. Figure 3 shows the waveform.

Figure 3. Enabling clock stretching before sending data

3.3 Enabling clock stretching before receiving data

When Slave receives data, the RDF bit is set. Slave holds the SCL to the LOW state. After clearing the TDF bit, Slave releases the SCL and Master controls the SCL. This time, there is a 1000 µs delay. Figure 4 shows the waveform.

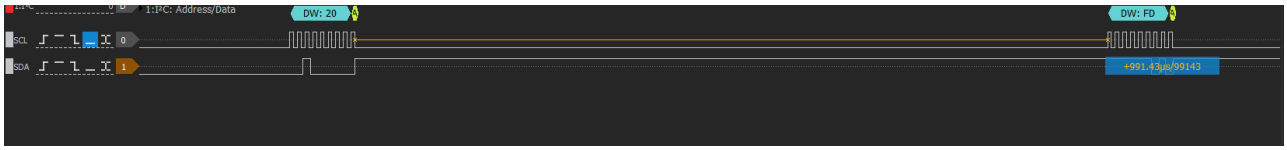

Figure 4. Enabling clock stretching before receiving data

3.4 Enabling clock stretching before sending ACK/NACK

Slave holds the SCL to the **LOW** state after receiving the 8th clock pulse. The clock stretching function is enabled and users can send the ACK/NACK. To send ACK or NACK to Master, write the bit0 of **STAR** register to **0** or **1**. Before writing **0** to **STAR**, add a 500 µs delay. Figure 5 shows the clock stretching and delay after Slave receives the address information. To form the waveform, there is a 500 ns delay between the 8th clock and 9th clock.

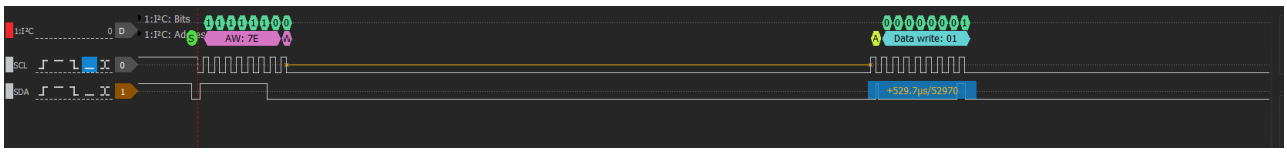

Figure 5. Enabling clock stretching before sending ACK/NACK

Figure 6 shows the ACK delay after Slave receives the data from Master.

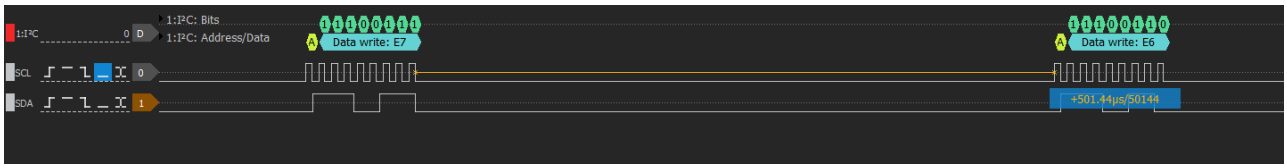

Figure 6. Enabling clock stretching before sending ACK/NACK

3.5 Enabling clock stretching

To enable the clock stretching function, configure the parameters as highlighted in Figure 7.

```
void LPI2C_SlaveGetDefaultConfig(lpi2c_slave_config_t *slaveConfig)
{
    /* Initializes the configure structure to zero. */
    (void)memset(slaveConfig, 0, sizeof(*slaveConfig));

    slaveConfig->enableSlave              = true;
    slaveConfig->address0                 = 0U;
    slaveConfig->address1                 = 0U;
    slaveConfig->addressMatchMode         = kLPI2C_MatchAddress0;
    slaveConfig->filterDozeEnable         = true;
    slaveConfig->filterEnable             = true;
    slaveConfig->enableGeneralCall        = false;
    slaveConfig->sclStall.enableAck       = false;
    slaveConfig->sclStall.enableTx        = true;
    slaveConfig->sclStall.enableRx        = true;
    slaveConfig->sclStall.enableAddress   = false;
    slaveConfig->ignoreAck                = false;
    slaveConfig->enableReceivedAddressRead = false;
    slaveConfig->sdaGlitchFilterWidth_ns  = 0U; /* TODO determine default width values */
    slaveConfig->sclGlitchFilterWidth_ns  = 0U;
    slaveConfig->dataValidDelay_ns        = 0U;
    slaveConfig->clockHoldTime_ns         = 0U;
}
```

Figure 7. Enabling clock stretching

4 Key Point

When enabling the clock stretching function, take care of the setup time and hold time. Parameters differ with speed. Figure 8 shows the AC timing parameters from I²C spec.

Table 10. Characteristics of the SDA and SCL bus lines for Standard, Fast, and Fast-mode Plus I²C-bus devices[1]

Symbol	Parameter	Conditions	Standard-mode		Fast-mode		Fast-mode Plus		Unit
			Min	Max	Min	Max	Min	Max	
f_{SCL}	SCL clock frequency		0	100	0	400	0	1000	kHz
$t_{HD;STA}$	hold time (repeated) START condition	After this period, the first clock pulse is generated.	4.0	-	0.6	-	0.26	-	µs
t_{LOW}	LOW period of the SCL clock		4.7	-	1.3	-	0.5	-	µs
t_{HIGH}	HIGH period of the SCL clock		4.0	-	0.6	-	0.26	-	µs
$t_{SU;STA}$	set-up time for a repeated START condition		4.7	-	0.6	-	0.26	-	µs
$t_{HD;DAT}$	data hold time[2]	CBUS compatible masters (see Remark in Section 4.1)	5.0	-	-	-	-	-	µs
		I²C-bus devices	0[3]	-[4]	0[3]	-[4]	0	-	µs
$t_{SU;DAT}$	data set-up time		250	-	100[5]	-	50	-	ns
t_r	rise time of both SDA and SCL signals		-	1000	20	300	-	120	ns
t_f	fall time of both SDA and SCL signals[3][6][7][8]		-	300	20 × (V_{DD} / 5.5 V)	300	20 × (V_{DD} / 5.5 V)[9]	120[8]	ns
$t_{SU;STO}$	set-up time for STOP condition		4.0	-	0.6	-	0.26	-	µs
t_{BUF}	bus free time between a STOP and START condition		4.7	-	1.3	-	0.5	-	µs
C_b	capacitive load for each bus line[10]		-	400	-	400	-	550	pF
$t_{VD;DAT}$	data valid time[11]		-	3.45[4]	-	0.9[4]	-	0.45[4]	µs
$t_{VD;ACK}$	data valid acknowledge time[12]		-	3.45[4]	-	0.9[4]	-	0.45[4]	µs
V_{nL}	noise margin at the LOW level	for each connected device (including hysteresis)	0.1V_{DD}	-	0.1V_{DD}	-	0.1V_{DD}	-	V
V_{nH}	noise margin at the HIGH level	for each connected device (including hysteresis)	0.2V_{DD}	-	0.2V_{DD}	-	0.2V_{DD}	-	V

Figure 8. AC timing parameters from I²C spec

For RT1010, in the **SCFGR2** register:

- CLKHOLD is used to configure the setup time.
- DATAVD is used to configure the hold time.

—	
13-8 DATAVD	Data Valid Delay Configures the SDA data valid delay time for the I2C slave, and is equal to FILTSCL+DATAVD+3 cycles. • The data valid delay must be configured to be less than the minimum SCL low period • The I2C slave data valid delay time is not affected by the PRESCALE configuration, and the I2C slave data valid delay time is disabled in high speed mode
7-4 —	Reserved
3-0 CLKHOLD	Clock Hold Time Configures the minimum clock hold time for the I2C slave, when clock stretching is enabled. • The minimum hold time is equal to CLKHOLD+3 cycles • The I2C slave clock hold time is not affected by the PRESCALE configuration, and the I2C slave clock hold time is disabled in high speed mode

Figure 9. CLKHOLD and DATAVD

To set the setup time and hold time, use the following formula:

$$t = (CLKHOLD + 3) * T_{clk}$$

The real setup time depends on two parameters: CLKHOLD value and T_{clk}. T_{clk} means the period of I²C functional clock. As shown in Figure 8, the hold time is **0** for an I²C device.

5 Revision history

Revision number	Date	Substantive changes
0	3 December 2021	Initial release

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Limited warranty and liability — Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

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.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer’s applications and products. Customer’s responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer’s applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

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

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

Date of release: 3 December 2021

Document identifier: AN13472

arm