LG69T (AA,AJ)
IMU Standalone Solution Application Note

GNSS Module Series

Version: 1.0

Date: 2024-01-19

Status: Released

At Quectel, our aim is to provide timely and comprehensive services to our customers. If you require any assistance, please contact our headquarters:

Quectel Wireless Solutions Co., Ltd.
Building 5, Shanghai Business Park Phase III (Area B), No.1016 Tianlin Road, Minhang District, Shanghai 200233, China
Tel: +86 21 5108 6236
Email: info@quectel.com

Or our local offices. For more information, please visit:
http://www.quectel.com/support/sales.htm.

For technical support, or to report documentation errors, please visit:
http://www.quectel.com/support/technical.htm.
Or email us at: support@quectel.com.

Legal Notices

We offer information as a service to you. The provided information is based on your requirements and we make every effort to ensure its quality. You agree that you are responsible for using independent analysis and evaluation in designing intended products, and we provide reference designs for illustrative purposes only. Before using any hardware, software or service guided by this document, please read this notice carefully. Even though we employ commercially reasonable efforts to provide the best possible experience, you hereby acknowledge and agree that this document and related services hereunder are provided to you on an “as available” basis. We may revise or restate this document from time to time at our sole discretion without any prior notice to you.

Use and Disclosure Restrictions

License Agreements

Documents and information provided by us shall be kept confidential, unless specific permission is granted. They shall not be accessed or used for any purpose except as expressly provided herein.

Copyright

Our and third-party products hereunder may contain copyrighted material. Such copyrighted material shall not be copied, reproduced, distributed, merged, published, translated, or modified without prior written consent. We and the third party have exclusive rights over copyrighted material. No license shall be granted or conveyed under any patents, copyrights, trademarks, or service mark rights. To avoid ambiguities, purchasing in any form cannot be deemed as granting a license other than the normal non-exclusive, royalty-free license to use the material. We reserve the right to take legal action for noncompliance with abovementioned requirements, unauthorized use, or other illegal or malicious use of the material.

Trademarks

Except as otherwise set forth herein, nothing in this document shall be construed as conferring any rights to use any trademark, trade name or name, abbreviation, or counterfeit product thereof owned by Quectel or any third party in advertising, publicity, or other aspects.

Third-Party Rights

This document may refer to hardware, software and/or documentation owned by one or more third parties ("third-party materials"). Use of such third-party materials shall be governed by all restrictions and obligations applicable thereto.

We make no warranty or representation, either express or implied, regarding the third-party materials, including but not limited to any implied or statutory, warranties of merchantability or fitness for a particular purpose, quiet enjoyment, system integration, information accuracy, and non-infringement of any third-party intellectual property rights with regard to the licensed technology or use thereof. Nothing herein constitutes a representation or warranty by us to either develop, enhance, modify, distribute, market, sell, offer for sale, or otherwise maintain production of any our products or any other hardware, software, device, tool, information, or product. We moreover disclaim any and all warranties arising from the course of dealing or usage of trade.

Privacy Policy

To implement module functionality, certain device data are uploaded to Quectel's or third-party's servers, including carriers, chipset suppliers or customer-designated servers. Quectel, strictly abiding by the relevant laws and regulations, shall retain, use, disclose or otherwise process relevant data for the purpose of performing the service only or as permitted by applicable laws. Before data interaction with third parties, please be informed of their privacy and data security policy.

Disclaimer

a) We acknowledge no liability for any injury or damage arising from the reliance upon the information.
b) We shall bear no liability resulting from any inaccuracies or omissions, or from the use of the information contained herein.
c) While we have made every effort to ensure that the functions and features under development are free from errors, it is possible that they could contain errors, inaccuracies, and omissions. Unless otherwise provided by valid agreement, we make no warranties of any kind, either implied or express, and exclude all liability for any loss or damage suffered in connection with the use of features and functions under development, to the maximum extent permitted by law, regardless of whether such loss or damage may have been foreseeable.
d) We are not responsible for the accessibility, safety, accuracy, availability, legality, or completeness of information, advertising, commercial offers, products, services, and materials on third-party websites and third-party resources.

About the Document

Document Information	
Title	**LG69T (AA,AJ) IMU Standalone Solution Application Note**
Subtitle	GNSS Module Series
Document Type	Application Note
Document Status	Released

Revision History

Version	Date	Description
-	2023-09-27	Creation of the document
1.0	2024-01-19	First official release

Contents

Table Index

Figure Index

1 Introduction

This document outlines the functions and usage of the Inertial Measurement Unit (IMU) in LG69T (AA, AJ) modules, presenting an IMU standalone solution tailored for LG69T (AA, AJ). In the remainder of the document we will simply refer to it as module IMU.

The module IMU provides the following functions:

- Support the output of temperature data, 3-axis accelerometer data and 3-axis gyroscope data
- Accelerometer full scale range of ±16g (g stands for gravitational acceleration)
- Gyroscope full scale range is from ±125 dps to ±4000 dps
- 3 KB data cache
- Data transmission using I2C interface

2 Module IMU Introduction

This chapter mainly introduces the module I2C interface, IMU device specifications and main registers.

2.1. Module I2C Interface

The module IMU uses the I2C interface of LG69T (AA, AJ) for data transmission. The I2C interface supports the following features:

- Standard mode (100 kbps) and fast mode (400 kbps)
- Slave mode only
- 7-bit addresses. Default slave address: 11010010b

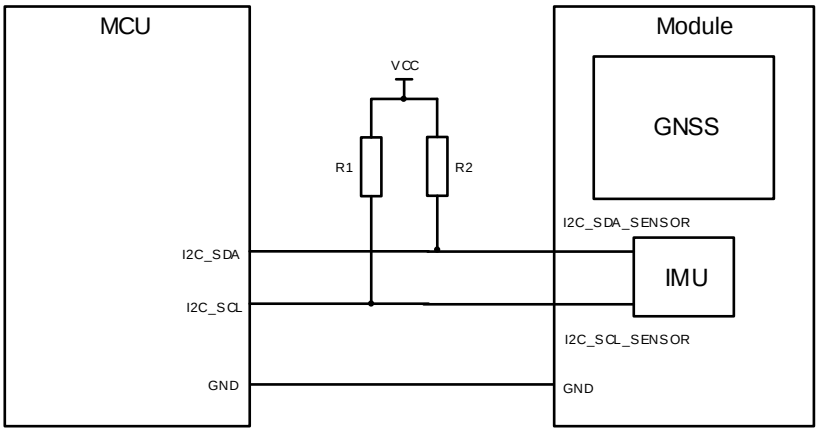

Figure 1: I2C Interface Connection Diagram

NOTE

The diagram above is for illustration purpose only. For details on the reference design, the pin definition and specifications of I2C interface, see *documents [1]* and *[2] hardware designs*.

2.2. IMU Device Specifications

2.2.1. Mechanical Specifications

Table 1: Mechanical Specifications

Symbol	Parameter	Test conditions	Typical Value	Unit
LA_So	Linear acceleration sensitivity	@LA_FS = ±2g @LA_FS = ±4g @LA_FS = ±8g @LA_FS = ±16g	0.061 0.122 0.244 0.488	mg/LSB
G_So	Angular rate sensitivity	@G_FS = ±125 dps @G_FS = ±250 dps @G_FS = ±500 dps @G_FS = ±1000 dps @G_FS = ±2000 dps @G_FS = ±4000 dps	4.37 8.75 17.5 35.0 70.0 140.0	mdps/LSB

2.2.2. Temperature Sensor Specifications

Table 2: Temperature Sensor Specifications

Symbol	Parameter	Typical Value	Unit
TODR	Temperature refresh rate	52	Hz
TSen	Temperature sensitivity	256	LSB/°C

2.3. Main Registers

2.3.1. WHO_AM_I (0x0F)

This register (R) is used to confirm the IMU mode. Its value is fixed at 0x6B.

WHO_AM_I Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	1	0	1	0	1	1

2.3.2. CTRL1_XL (0x10)

This register (R/W) is used to configure the 3-axis accelerometer parameters.

CTRL1_XL Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ODR_XL3	ODR_XL2	ODR_XL1	ODR_XL0	FS1_XL1	FS1_XL0	LPF2_XL_EN	0

Table 3: CTRL1_XL Parameter Description

Parameter	Parameter Description
ODR_XL [3:0]	Accelerometer ODR: 0000: Power-down 0001:12.5 Hz 0010: 26 Hz 0011: 52 Hz 0100: 104 Hz 0101: 208 Hz 0110: 417 Hz 0111: 833 Hz 1000: 1667 Hz 1001: 3333 Hz 1010: 6667 Hz Note: It is not allowed to set ODR_XL [3:0] (i.e., Bit 7–4) to 1011 or 11xx.
FS [1:0]_XL	Full scale for accelerometer. g is the gravitational acceleration. 00: 2g 01: 16g 10: 4g 11: 8g
LPF2_XL_EN	Accelerometer high-resolution selection: 0: Output from first stage of digital filtering selected 1: Output from LPF2 second filtering stage selected

2.3.3. CTRL2_G (0x11)

This register (R/W) is used to configure the 3-axis gyroscope parameters.

CTRL2_G Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ODR_G3	ODR_G2	ODR_G1	ODR_G0	FS1_G	FS0_G	FS_125	FS_4000

Table 4: CTRL2_G Parameter Description

Parameter	Parameter Description
ODR_G [3:0]	Gyroscope ODR: 0000: Power-down 0001: 12.5 Hz 0010: 26 Hz 0011: 52 Hz 0100: 104 Hz 0101: 208 Hz 0110: 417 Hz 0111: 833 Hz 1000: 1667 Hz 1001: 3333 Hz 1010: 6667 Hz Note: It is not allowed to set ODR_G [3:0] (i.e., Bit 7–4) to 1011 or 11xx.
FS [1:0]_G	Full scale for gyroscope. 00: 250 dps 01: 500 dps 10: 1000 dps 11: 2000 dps
FS_125	Minimum range: 0: FS selected through **FS [1:0]_G** 1: FS set to 125 dps
FS_4000	Maximum range: 0: FS selected through **FS [1:0]_G** or **FS_125** 1: FS set to 4000 dps

NOTE

1. When **FS_4000** is set to 1, **FS_4000** takes effect, **FS_125** and **FS [1:0]_G** does not take effect.
2. When **FS_4000** is set to 0:
   1) If **FS_125** is set to 1, **FS_125** takes effect, and **FS [1:0]_G** does not take effect.
   2) If **FS_125** is set to 0, **FS [1:0]_G** takes effect.

2.3.4. CTRL3_C (0x12)

This register (R/W) is control register 3.

CTRL3_C Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BOOT	BDU	0	0	0	IF_INC	0	SW_RESET

Table 5: CTRL3_C Parameter Description

Parameter	Parameter Description
BOOT	Reboots memory content. 0: Normal mode 1: Reboot memory content Note: The accelerometer must be ON, i.e., the **ODR_XL [3:0]** in **CTRL1_XL** is not 0; see *Chapter 2.3.2 CTRL1_XL (0x10)*. Once BOOT (Bit 7) is configured to 1 and takes effect, it is automatically updated to 0.
BDU	Data block update. 0: Continuous update; 1: Output registers are updated once **MSB** and **LSB** have been read
IF_INC	Register address automatically incremented during a multiple byte access with a serial interface (I2C). 0: Disabled; 1: Enabled
SW_RESET	Software reset. 0: Do not reset device 1: Reset device Note: Once **SW_RESET** (Bit 0) is configured to 1 and takes effect, it is automatically updated to 0.

2.3.5. CTRL4_C (0x13)

This register (R/W) is control register 4.

CTRL4_C Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	SLEEP_G	0	0	0	I2C_disable	LPF1_SEL_G	0

Table 6: CTRL4_C Parameter Description

Parameter	Parameter Description
SLEEP_G	Gyroscope sleep mode. 0: Disabled 1: Enabled
I2C_disable	Enable/disable I2C interface. 0: Enable 1: Disable
LPF1_SEL_G	Enable/disable gyroscope low-pass filter (LPF1). Bandwidth can be selected through **FTYPE [2:0]** in **CTRL6_G**. 0: Disable 1: Enable

2.3.6. CTRL6_G (0x15)

This register (R/W) is control register 6.

CTRL6_G Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	0	FTYPE_2	FTYPE_1	FTYPE_0

Table 7: CTRL6_G Parameter Description

Parameter	Parameter Description
FTYPE [2:0]	Gyroscope low-pass filter (LPF1) bandwidth. Selectable bandwidth values, see *Table 8: Gyroscope LPF1 Bandwidth (Unit: Hz)*.

Table 8: Gyroscope LPF1 Bandwidth (Unit: Hz)

FTYPE [2:0]	12.5 Hz	26 Hz	52 Hz	104 Hz	208 Hz	417 Hz	833 Hz	1.67 kHz	3.33 kHz	6.67 kHz
000	4.3	8.3	16.7	33	67	133	222	274	292	297
001	4.3	8.3	16.7	33	67	128	186	212	220	223
010	4.3	8.3	16.7	33	67	112	140	150	153	154
011	4.3	8.3	16.7	33	67	134	260	390	451	470
100	4.3	8.3	16.7	34	62	86	96	99	-	-
101	4.3	8.3	16.9	31	43	48	49	50	-	-
110	4.3	8.3	13.4	19	23	24.6	25	25	-	-
111	4.3	8.3	9.8	11.6	12.2	12.6	12.6	12.6	-	-

2.3.7. Output Registers (0x20–0x2D)

This chapter provides an overview of the main output registers of the module IMU. Register L represents low-order data, and register H represents high-order data. It is important to note that these registers have no actual physical meaning and must be converted for practical use. For specific conversion methods, see *Chapter 4.5 Data Conversion*.

2.3.7.1. OUT_TEMP_L (0x20), OUT_TEMP_H (0x21)

Temperature data output register (R). Temperature data is represented by the two's complement of the combination of registers L and H.

OUT_TEMP_L (0x20)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Temp7	Temp6	Temp5	Temp4	Temp3	Temp2	Temp1	Temp0

OUT_TEMP_H (0x21)

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Temp15	Temp14	Temp13	Temp12	Temp11	Temp10	Temp9	Temp8

2.3.7.2. OUTX_L_G (0x22), OUTX_H_G (0x23)

Angular rate sensor pitch axis (X) angular rate output register (R). Angular rate output for the pitch axis (x) is represented by the two's complement of the combination of registers L and H.

OUTX_L_G (0x22)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
D7	D6	D5	D4	D3	D2	D1	D0

OUTX_H_G (0x23)

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
D15	D14	D13	D12	D11	D10	D9	D8

2.3.7.3. OUTY_L_G (0x24), OUTY_H_G (0x25)

Angular rate sensor roll axis (Y) angular rate output register (R). Angular rate output for the roll axis (Y) is represented by the two's complement of the combination of registers L and H.

OUTY_L_G (0x24)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
D7	D6	D5	D4	D3	D2	D1	D0

OUTY_L_G (0x25)

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
D15	D14	D13	D12	D11	D10	D9	D8

2.3.7.4. OUTZ_L_G (0x26), OUTZ_H_G (0x27)

Angular rate sensor yaw axis (Z) angular rate output register (R). Angular rate output for the yaw axis (Z) is represented by the two's complement of the combination of registers L and H.

OUTZ_L_G (0x26)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
D7	D6	D5	D4	D3	D2	D1	D0

OUTZ_H_G (0x27)

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
D15	D14	D13	D12	D11	D10	D9	D8

2.3.7.5. OUTX_L_A (0x28), OUTX_H_A (0x29)

Linear acceleration sensor X-axis output register (R). Linear acceleration output for the X-axis is represented by the two's complement of the combination of registers L and H.

OUTX_L_A (0x28)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
D7	D6	D5	D4	D3	D2	D1	D0

OUTX_H_A (0x29)

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
D15	D14	D13	D12	D11	D10	D9	D8

2.3.7.6. OUTY_L_A (0x2A), OUTY_H_A (0x2B)

Linear acceleration sensor Y-axis output register (R). Linear acceleration output for the Y-axis is represented by the two's complement of the combination of registers L and H.

OUTY_L_A (0x2A)

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
D7	D6	D5	D4	D3	D2	D1	D0

OUTY_H_A (0x2B)

bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8
D15	D14	D13	D12	D11	D10	D9	D8

2.3.7.7. OUTZ_L_A (0x2C), OUTZ_H_A (0x2D)

Linear acceleration sensor Z-axis output register (R). It is represented by the two's complement of the combination of registers L and H.

OUTZ_L_A (0x2C)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
D7	D6	D5	D4	D3	D2	D1	D0

OUTZ_H_A (0x2D)

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
D15	D14	D13	D12	D11	D10	D9	D8

..

3 I2C Configuration for Module IMU

3.1. I2C Bus Introduction

3.1.1. START and STOP Signals

The transaction on the bus is initiated through a START signal (S). A START signal (S) is defined as a HIGH to LOW transition on the SDA line while the SCL line is held HIGH. The bus is considered busy until the master generates a STOP signal (P) on the bus. A STOP signal (P) is defined as a LOW to HIGH transition on the SDA line while the SCL line is held HIGH. In addition, the bus remains busy if a repeated START (S) is generated instead of a STOP signal.

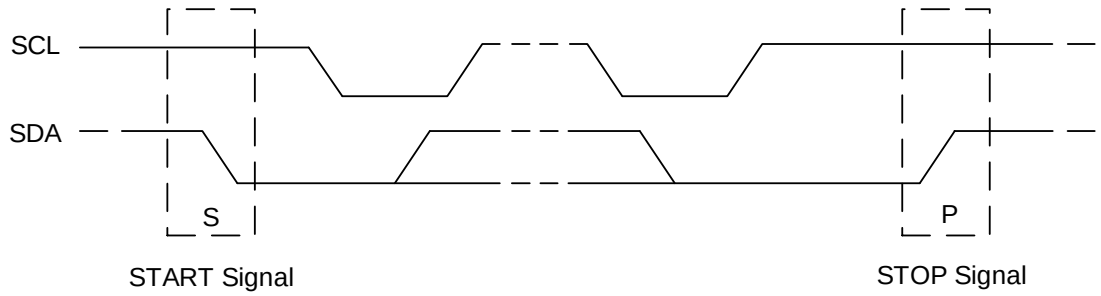

Figure 2: START and STOP Signals

3.1.2. Data Transfer and Acknowledgement

Each I2C data byte is 8 bits long. Any number of bytes can transmitted per transfer, provided that the number does not exceed the I2C buffer size. Each transferred byte must be followed by an acknowledgement signal (ACK). The clock for the acknowledgement signal (ACK) is generated by the master, while the slave generates the actual acknowledgement signal by pulling down the SDA line and holding it LOW during the HIGH portion of the acknowledgement clock pulse.

If the slave is busy and cannot transmit or receive another byte of data until the current processing task has been performed, it can hold SCL LOW, thus forcing the master into a wait state. Normal data transfer resumes when the slave is ready for another byte of data and releases the clock line.

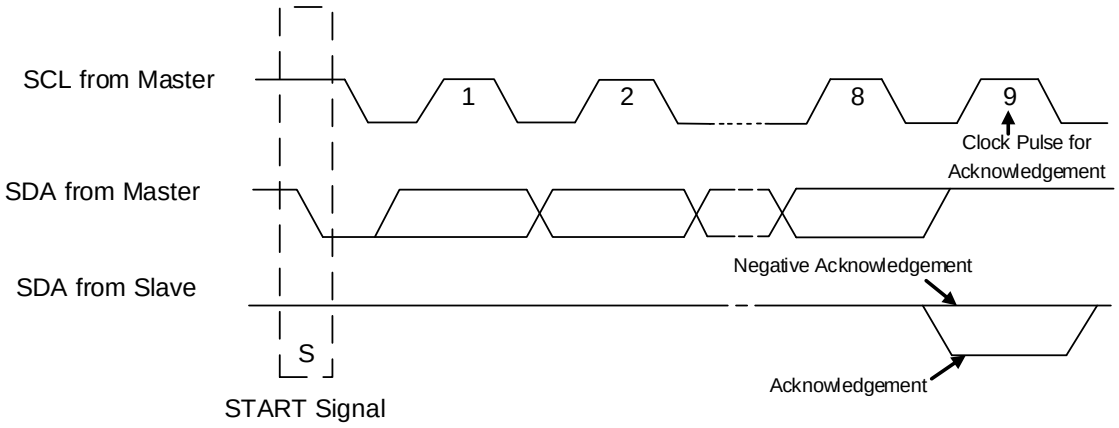

Figure 3: Acknowledgement on I2C Bus

3.1.3. Communication

After initiating the communication with the START signal (S), the master sends a 7-bit slave address (0xD4) followed by an additional 8th bit, the read/write bit, to inform the slave if the master is writing to it or reading from it. If the bit is 0, the master is writing to the slave. If the bit is 1, the master is reading from the slave.

Data is transferred on the SDA line with the Most Significant bit (MSB) first. Then, the master releases the SDA line and waits for the acknowledgement signal (ACK) from the slave device. For each transferred byte, the slave device must return an acknowledgement bit. To acknowledge, the slave device pulls the SDA line LOW and keeps it LOW during the HIGH period of the SCL line.

Data transmission is always terminated by the master with a STOP signal (P), thus freeing the communication line. However, the master can generate a repeated START signal (S), and initiate comunication with another slave without first generating a STOP signal (P). All SDA changes should take place while SCL is LOW, the only exception to this rule is during the generation of START and STOP signals.

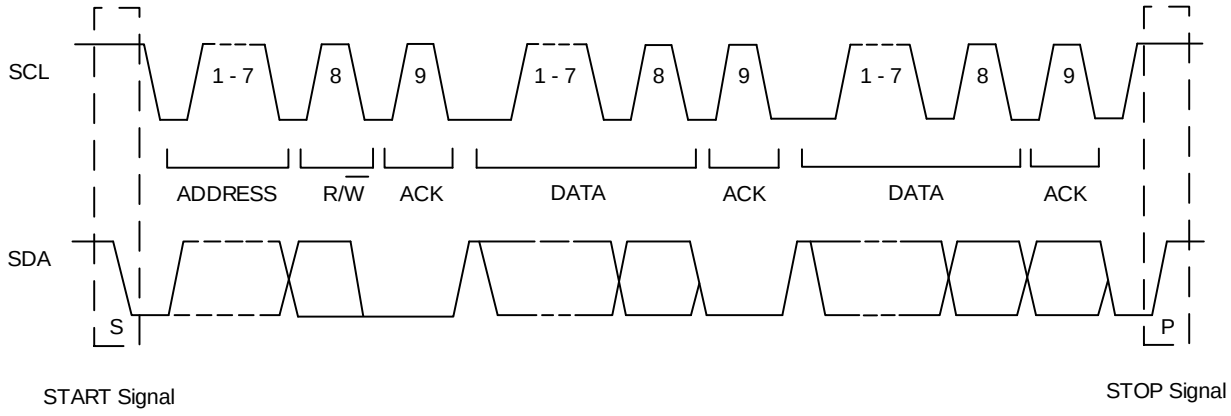

Figure 4: I2C Data Transfer Completed

3.1.4. Communication Address

The address of the module IMU as a slave is 0xD4 (1101010xb, x=0 (W) or 1 (R)).

3.2. I2C Operation Sequence Diagram

3.2.1. Write Sequence

The sequence charts for writing data to the I2C slave are shown below.

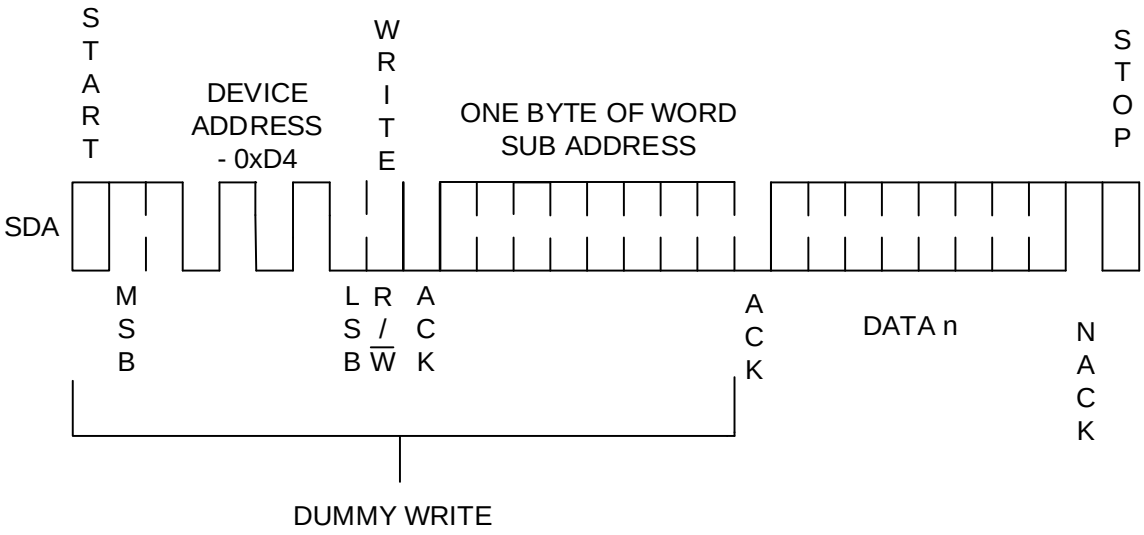

Figure 5: Write Sequence

To write data to the module, the master sends a START signal (S), followed by the I2C slave address (0xD4) and a write bit (W). At the 9th clock cycle (when the clock is high), the slave acknowledges the master (ACK). Then, the master sends the slave sub-address (register address) and puts the data onto the bus after the slave acknowledges the master (ACK). The slave responds with ACK for every 8 bits transferred to indicate that it has received the data. If it generates a low ACK bit, then it has received the data and is ready to accept another byte. If it generates a high ACK bit, then it is indicating it cannot accept any more data and the master should terminate the transfer by sending a STOP signal (P).

3.2.2. Read Sequence

The sequence charts for reading data from the I2C slave are shown below.

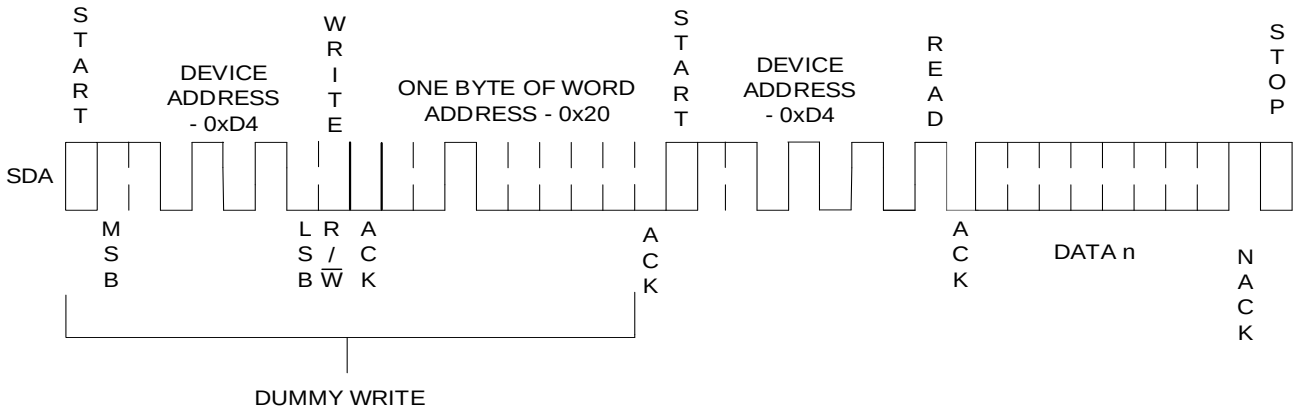

Figure 6: Read Sequence

To read the slave data, the master sends a START signal (S), followed by the I2C slave address (0xD4) and a write bit (W). The slave responds with an ACK signal to indicate that it was addressed by the master. Then, the master sends the fixed byte indicating the output register address of the slave. Upon receiving the ACK signal from the slave, the master transmits a START signal (S) again followed by the slave address (0xD4) and a read bit (R). As a result, the salve responds with an ACK signal followed by the transmission of the data. If the host responds with an ACK signal to the slave's 8-bit data transmission, the slave will keep sending 8-bit data and repeat this procedure for reading multi bytes. The communication ends with a negative acknowledgement signal (NACK) and a STOP signal (P) from the master. The NACK signal means that the SDA line remains high at the 9th clock cycle.

3.3. Configuration Procedure

The module IMU supports data transmission through the I2C interface. This chapter provides a detailed overview of the operational procedures for configuring I2C.

3.3.1. Recommended I2C Configuration

This routine uses the MCU model of STM32H7 as the host, and the configuration operation of STM32H7 on the I2C bus is as shown in figure below:

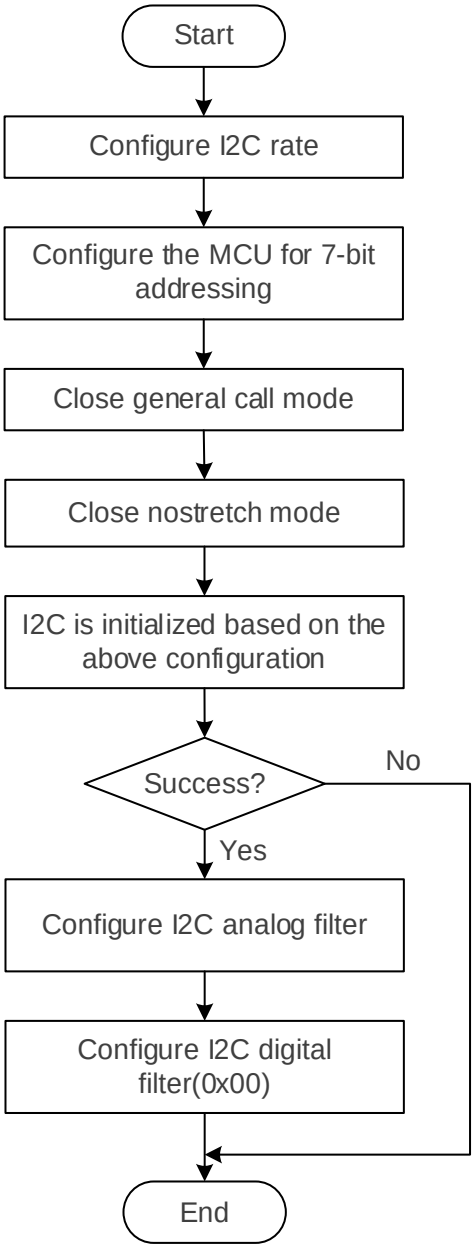

Figure 7: Master I2C Initialization Process

NOTE

1. Because the MCU is the master, there is no need to configure slave-related parameters.
2. The process outlined above is for reference purposes only. For details about I2C initialization, see *Chapter 4 Pseudocode Example*.

3.3.2. Recommended IMU Register Configuration

This routine uses the module IMU as the slave, and the register configuration process of the module IMU is shown in figure below:

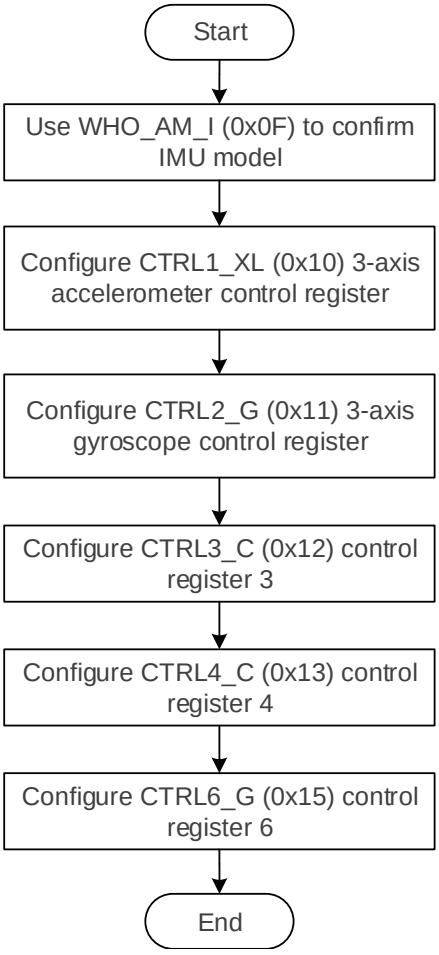

Figure 8: Slave IMU Configuration Process

Operation steps:

Step 1 Use the **WHO_AM_I (0x0F)** register to confirm IMU model. Determine the IMU model based on the return value. The expected return value is 0x6B. See *Chapter 2.3.1 WHO_AM_I (0x0F)* for details.

Step 2 Set the **CTRL1_XL (0x10)** register value to 0x4A, i.e., configure **ODR_XL** to 104 Hz, **FS_XL** to ±4g and **LPF2_XL_EN** to 1. See *Chapter 2.3.2 CTRL1_XL (0x10)* for details.

Step 3 Set the **CTRL2_G (0x11)** register value to 0x42, i.e., configure **ODR_G** to 104 Hz and **FS_125** to 125 dps. See *Chapter 2.3.3 CTRL2_G (0x11)* for details.

Step 4 Set the **CTRL3_C (0x12)** register value to 0x04, i.e., configure **IF_INC** to 1, enabling auto-incrementing register address in serial interface (I2C) multibyte accesses. See *Chapter 2.3.4 CTRL3_C (0x12)* for details.

Step 5 Set the **CTRL4_C (0x13)** register value to 0x02, i.e., configure **LPF1_SEL_G** to 1, enabling gyroscope low-pass filtering (LPF1). See *Chapter 2.3.5 CTRL4_C (0x13)* for details.

Step 6 Set the **CTRL6_G (0x15)** register value to 0x01, i.e., configure **FTYPE** to 001. For gyroscope low-pass filter (LPF1) bandwidth, see *Chapter 2.3.6 CTRL6_G (0x15)* for details.

NOTE

1. It is recommended to verify the results of the configuration by reading and writing the return values of the IMU functions. See *Chapter 4.3 Read IMU Data* and *Chapter 4.4 Write IMU Data* for details.
2. For IMU output registers, all critical data (temperature, 3-axis accelerometer and 3-axis gyroscope) can be retrieved in a single read operation without configuration, because the memory addresses of temperature velocity register, 3-axis accelerometer output register and 3-axis gyroscope output register are consecutive (0x20 to 0x2D). See *Chapter 2.3.7 Output Registers (0x20–0x2D)* for details.
3. The process outlined above is for reference purposes only. For details about reading and writing IMU data, see *Chapter 4 Pseudocode Example*.

4 Pseudocode Example

This chapter presents the pseudo-code of the data interaction between the I2C interface and the module IMU. For the explanation of specific parameters, see the official HAL library of STM32H7. The specific version is STM32H7xx HAL Driver version number V1.6.0.

4.1. Initialize I2C

- **Prototype**

```
bool Ql_I2C_Open(void)
{
    HI2Cx.Instance = I2Cx;
    HI2Cx.Init.Timing = 0x00B03FDB;
    HI2Cx.Init.OwnAddress1 = 0;
    HI2Cx.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
    HI2Cx.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
    HI2Cx.Init.OwnAddress2 = 0;
    HI2Cx.Init.OwnAddress2Masks = I2C_OA2_NOMASK;
    HI2Cx.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
    HI2Cx.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;

    if(HAL_OK != HAL_I2C_Init(&HI2Cx))
    {
        return false;
    }

    /* Configure Analogue filter */
    if(HAL_OK != HAL_I2CEx_ConfigAnalogFilter(&HI2Cx, I2C_ANALOGFILTER_ENABLE))
    {
        return false;
    }

    /* Configure Digital filter */
    if(HAL_OK != HAL_I2CEx_ConfigDigitalFilter(&HI2Cx, 0))
    {
```

```
        return false;
    }
    return true;
}
```

- **Return Value**

1 Successful execution
0 Failed execution

4.2. Initialize IMU

- **Prototype**

```
bool Ql_ACC_GYRO_Init(void)
{
    unsigned char rx_buff[1] = {0};
    unsigned char tx_reg_value = 0;

    Ql_I2C_Open();// I2C must be opened before initializing the IMU

    /* Read the IMU model and determine whether it is a given IMU model */
    if(true != Ql_I2C_Read(SLAVE_ADDRESS, WHO_AM_I, &rx_buff[0], 1, 100))
    {
        return false;
    }
    if(WHO_AM_I_Value != rx_buff[0]) //0x6B is the return value of WHO_AM_I register
    {
        return false;
    }

    /* Config CTRL1_XL*/
    tx_reg_value = 0x4A;    // ODR = 104Hz, FS = +-4g, LPF2_EN
    if(true != Ql_I2C_Write(SLAVE_ADDRESS, CTRL1_XL, &tx_reg_value, 1, 100))
    {
        return false;
    }

    /* Config CTRL2_G */
    tx_reg_value = 0x42;    // ODR = 104Hz, FS = 125dps
    if(true != Ql_I2C_Write(SLAVE_ADDRESS, CTRL2_G, &tx_reg_value, 1, 100))
    {
```

```
        return false;
    }

    /* Config CTRL3_C */
    tx_reg_value = 0x04;    // IF_INC = 1;
    if(true != Ql_I2C_Write(SLAVE_ADDRESS, CTRL3_C, &tx_reg_value, 1, 100))
    {
        return false;
    }

    /* Config CTRL4_C 4*/
    tx_reg_value = 0x02;    // LPF1_EN
    if(true != Ql_I2C_Write(SLAVE_ADDRESS, CTRL4_C, &tx_reg_value, 1, 100))
    {
        return false;
    }

    /* Config CTRL6_G*/
    tx_reg_value = 0x01;    // FTYPE=001
    if(true != Ql_I2C_Write(SLAVE_ADDRESS, CTRL6_G, &tx_reg_value, 1, 100))
    {
        return false;
    }
    return true;
}
```

- **Return Value**

1 Successful execution
0 Failed execution

4.3. Read IMU Data

- **Prototype**

```
bool Ql_I2C_Read(uint8_t SLA_Addr, uint8_t Reg, uint8_t *pData, uint32_t Size,
uint32_t Timeout)
{
    if(HAL_OK == HAL_I2C_Mem_Read(&HI2Cx, SLA_Addr, Reg, I2C_MEMADD_SIZE_8BIT, pData,
Size, Timeout))
    {
        return true;
```

```
    }
    else
    {
        return false;
    }
}
```

- **Parameter**

SLA_Addr:
[In] I2C slave address, i.e., the address of the IMU (0xD4).

Reg:
[In] The output register address of the IMU, which is the address where the data reading starts. (0x20).

pData:
[Out] Starting address of data storage.

Size:
[In] The size of the read data. Unit: Byte.

Timeout:
[In] Timeout. Unit: ms.

- **Return Value**

1 Successful execution
0 Failed execution

4.4. Write IMU Data

- **Prototype**

```
bool Ql_I2C_Write(uint8_t SLA_Addr, uint8_t Reg, uint8_t *pData, uint32_t Size,
uint32_t Timeout)
{
    if(HAL_OK == HAL_I2C_Mem_Write(&HI2Cx, SLA_Addr, Reg, I2C_MEMADD_SIZE_8BIT,
    pData, Size, Timeout))
    {
        return true;
    }
    else
    {
```

```
        return false;
    }
}
```

- **Parameter:**

SLA_Addr:
[In] I2C slave address, i.e., the address of the IMU (0xD4).

Reg:
[In] IMU control register address.

pData:
[Out] Written data.

Size:
[In] The size of the written data. Unit: Byte.

Timeout:
[In] Timeout. Unit: ms.

- **Return Value**

1 Successful execution
0 Failed execution

4.5. Data Conversion

The output registers of module IMU provide raw data that needs to be converted to obtain meaningful values in terms of physical measurements, such as temperature. Therefore, the data from *Chapter 2.3.7 Output Registers (0x20–0x2D)* must be converted.

4.5.1. Temperature Data Conversion

- **Prototype:**

```
double Ql_Senor_Raw_To_Temperature(uint16_t raw_data)
{
    double result = raw_data;// binary raw data
    if ((raw_data & 0x8000) == 0x8000)
    {
        result = (-1.0f) * ((uint16_t)(~raw_data) + 1);
    }
```

```
    result = result / 256.0f + 25;

    return result;
}
```

- **Parameter：**

raw_data:
[In] The raw two's complement data in the temperature output register.

- **Return Value**

Physically meaningful temperature data.

4.5.2. Accelerometer Data Conversion

- **Prototype**

```
double Ql_Senor_Raw_To_ACC(uint16_t raw_data)
{
    double result = raw_data; // binary raw data
    if ((raw_data & 0x8000) == 0x8000)
    {
        result = (-1.0f) * ((uint16_t)(~raw_data) + 1);
    }

    return result * 0.122f / 1000.0f;// Please select the corresponding sensitivity
                              according to the //measuring range.

}
```

- **Parameter**

raw_data:
[In] The raw two's complement data in the acceleration output register.

- **Return Value**

Physically meaningful 3-axis acceleration data.

4.5.3. Gyroscope Data Conversion

- **Prototype**

```
double Ql_Senor_Raw_To_Gyro(uint16_t raw_data)
{
    double result = raw_data; // binary raw data
    if ((raw_data & 0x8000) == 0x8000)
    {
        result = (-1.0f) * ((uint16_t)(~raw_data) + 1);
    }

    return result * 4.37f / 1000.0f; // Please select the corresponding sensitivity
                            according to the        // measuring range.
}
```

- **Parameter**

raw_data:
[In] Raw two's complement data in the gyroscope output register.

- **Return Value**

Physically meaningful 3-axis gyroscopes data.

5 Appendix A References

Table 9: Related Documents

Document Name	
[1]	Quectel_LG69T(AA,AD,AF)_Hardware_Design
[2]	Quectel_LG69T(AI,AJ,AR)_Hardware_Design

Table 10: Terms and Abbreviations

Abbreviation	Description
ACK	Acknowledgement
FS	Full Scale
I2C	Inter-integrated Circuit
IMU	Inertial Measurement Unit
LPF	Low-pass Filter
LSB	Least Significant Bit
MCU	Microcontroller Unit
NACK	Negative Acknowledgement
ODR	Output Data Rate
SAD	Slave Address
SCL	Serial Clock
SDA	Serial Data

6 Appendix B Special Character

Table 11: Special Character

Special Character	Description
Underline	Default setting of a parameter