LC29H (BA,CA,DA,EA)
DR&RTK Application Note

GNSS Module Series

Version: 1.2.0

Date: 2024-02-07

Status: Preliminary

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	**LC29H (BA,CA,DA,EA) DR&RTK Application Note**
Subtitle	GNSS Module Series
Document Type	Application Note
Document Status	Preliminary

Revision History

Version	Date	Description
-	2022-02-25	Creation of the document
1.0	2022-09-19	First official release
1.1	2023-04-27	1. Added applicable module LC29H (EA). 2. Added an overview on DR (*Chapter 1.1*). 3. Changed the driving speed of more than 3 m/s to 2 m/s in step 4 of DR calibration, updated the testing scenarios in open sky area, urban main road, and added a note about scenarios where the module exits DR mode (*Chapter 2.6*). 4. Added PQTMCFGDRRTD, PQTMCFGIMUTC, PQTMDRPVA, PQTMCFGDRHOT and PQTMCFGDR commands (*Chapters 4.1.12*, *4.1.13*, *4.1.14*, *4.1.15* and *4.1.16*).
1.2.0	2024-02-07	1. Added the details of DR, RTK (*Chapter 1*). 2. Added introductions of DR hot start, temperature compensation, DR calibration data saving, module mounting and DR calibration (*Chapters 2.2*, *2.3*, *2.4*, *2.5* and *2.6*). 3. Added the RTK application description (*Chapter 3.2*). 4. Added the note of only supporting by LC29H (BA) and LC29H (CA) with software versions dedicated for four-wheel vehicles (*Chapters 4.1.13*, *4.1.14* and *4.1.16*).

Version	Date	Description
		5. Added the new messages of PQTMDRSAVE, PQTMDRCLR, PQTMCFGRTK, PQTMCFGLA and PQTMCFGLAM (*Chapters 4.1.17*, *4.1.18*, *4.1.19*, *4.1.20* and *4.1.21*). 6. Added the note that it's recommended to use PQTMCFGMSGRATE to configure specific messages (*Chapters 4.2.1* and *4.2.2*). 7. Added the chapters of FAQ and Example (*Chapters 5* and *6*).

Contents

Table Index

Figure Index

1 Introduction

This document describes the dead reckoning (DR) and real-time kinematic (RTK) features, including DR and RTK configurations and DR related messages for Quectel LC29H (BA), LC29H (CA), LC29H (DA) and LC29H (EA) modules. The features supported by each module are as follows:

- LC29H (BA) supports DR and RTK.
- LC29H (CA) only supports DR.
- LC29H (DA) only supports RTK (Max update rate: 1 Hz).
- LC29H (EA) only supports RTK (Max update rate: 10 Hz, 10 Hz by default).

1.1. Overview on DR

1.1.1. Introduction to DR

LC29H series provide two DR modes: ADR (Automotive Dead Reckoning) and UDR (Untethered Dead Reckoning). ADR requires the vehicle to provide the speed and reversing data to the module. User can connect the module to vehicle sensors or inject the speed and reverse data via communication interface. On the contrary, the UDR does not require the speed and reversing data, and the module integrates high-end DR technology to provide enhanced positioning performance without vehicle sensors. Therefore, the UDR has the characteristics of simple installation. It's an ideal solution for aftermarket applications which has no possibility to access the vehicle speed data. LC29H series can auto identify the speed data exist or not, hence it can auto-switch the UDR and ADR mode after each power on.

In addition, the LC29H series modules provide two different firmware for different vehicles. One is for four wheels vehicle, such as a car, and the other one is for two wheels vehicle, such as bicycle, scooter, motorcycle.

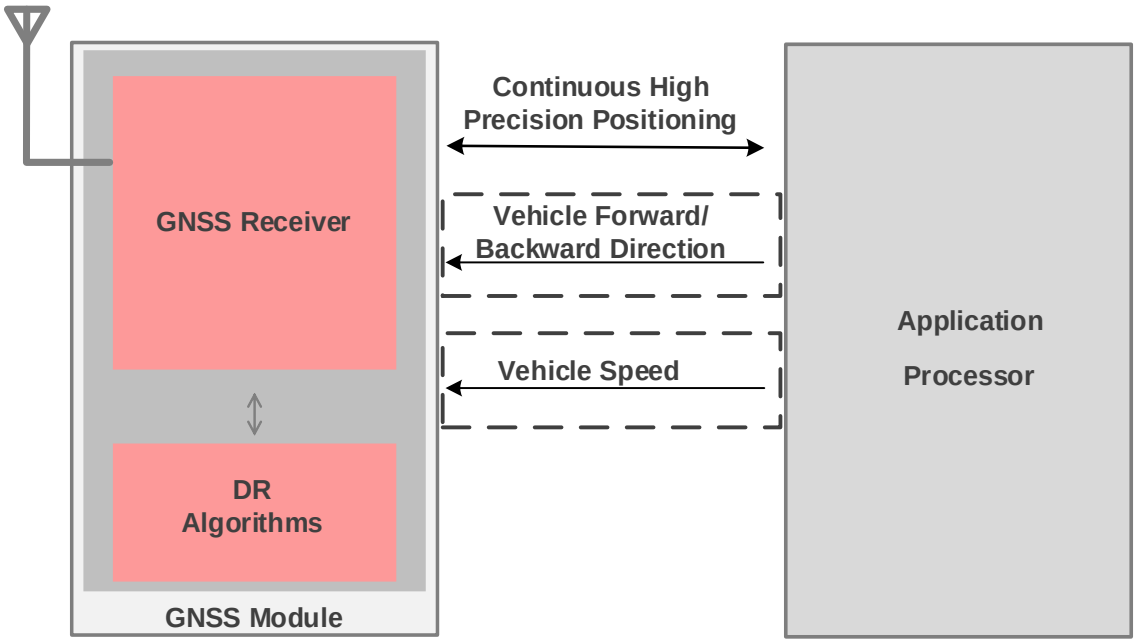


Figure 1: DR Application Architecture

1.1.2. Application Scenarios of DR

The application scenarios of DR mainly include wooded countryside, difficult urban environments, under viaducts, and even tunnels and underground parking, etc.

Figure 2: Underground Parking

1.2. Overview on RTK

1.2.1. Introduction to RTK

RTK (Real Time Kinematic) is a real-time differential GNSS technology based on carrier phase measurements, which can achieve high-precision positioning within a few centimeters. As we know, when the GNSS signal from satellites travels towards the receiver, it goes through 20,200 km (i.e., GPS) of ionosphere and atmosphere down to earth. The ionospheric effect significantly slows the signal and also can disturb it on the way. In addition, many factors, such as clouds, or obstacles, can affect the travel time and increase the position error. The RTK technology is based on the use of carrier measurements, with a reference station transmitting the corrections to a receiver needed to get the accurate position, which can eliminate the major errors in traditional GNSS positioning.

Both the RTK and traditional GNSS need to measure the distance from the satellite to the receiver, the traditional GNSS positioning uses PRN code-based ranging, however, the RTK uses carrier-based ranging. Because the difference of wavelength, carrier-based ranging can provide higher accuracy of the distance. For example, the wavelength of L1 signal is about 19 cm, but the wavelength of L1 PRN code frequency is about 300 m. The distance to a satellite is calculated by multiplying the carrier wavelength times the number of whole cycles between the satellite and the receiver and adding the phase difference. It's not easy to calculate the number of the wavelengths. The key to achieving centimeter-level accuracy in RTK is to resolve the unknown number of carrier cycle ambiguities into integers.

Depending on the number and signal status of satellites visible in the sky, RTK can obtain different position solution with different levels of accuracy. They are called "RTK float" and "RTK fixed", in a fixed solution, the number of wavelengths is an integer (whole number), and the algorithm is constrained to yield a whole number, in a float solution, the ambiguity is allowed to be a decimal or floating-point number. General, the accuracy of float solution is 10 ~ 50 cm or even worse, and a fixed solution can achieve an accuracy of less than 10 cm.

There are three parts in RTK solution:

- Reference station receiver with well-known position and is static, is called "**Base Station**".
- The communication link.
- The moving receiver, is called "**Rover**".

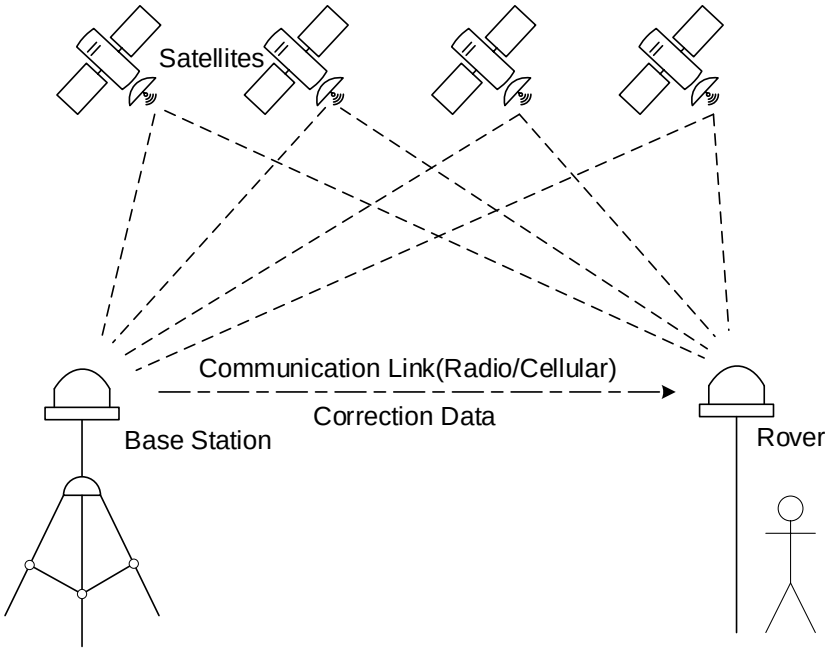


Figure 3: RTK Working Diagram

RTK working principle:

- Set up a base station on a known point to continuously receive all visible GNSS satellite signals.
- Send the base coordinates and correction data to the rover through data link (network, radio, etc.).
- The rover receives the data from the base and synchronously observes and collects the carrier phase data of GNSS satellites.
- The rover calculates a precise position with the received carrier phase and the correction data from base station.

Normally, RTK can be divided into traditional RTK and network RTK. In the traditional RTK, the transmission link of differential correction numbers is usually provided by wireless radio stations, and the distance between base stations and rover stations is generally within 20-30 km. Because the measurement deviation it forms is related to distance, exceeding this distance will cause a significant decrease in the positioning accuracy of the rover station. Network RTK technology is a real-time positioning technology that combines data from multiple base stations for error calculation and correction, and provides spatial location services to users. It requires data sharing between system centers and multiple reference stations through the network. Network RTK technology overcomes the shortcomings of traditional RTK technology, such as strong correlation between positioning accuracy and distance, and frequent installation of reference stations during operation. For more details about the RTK, see *document [1] RTK application*.

1.2.2. Application Scenarios of RTK

The main application scenarios of RTK include lawn mowers, drones, lane-level navigation, etc.

Figure 4: UAV

1.3. Overview on DR + RTK

1.3.1. Introduction to DR + RTK

LC29H (BA) supports DR and RTK, In many scenarios, we need to use both DR And RTK.

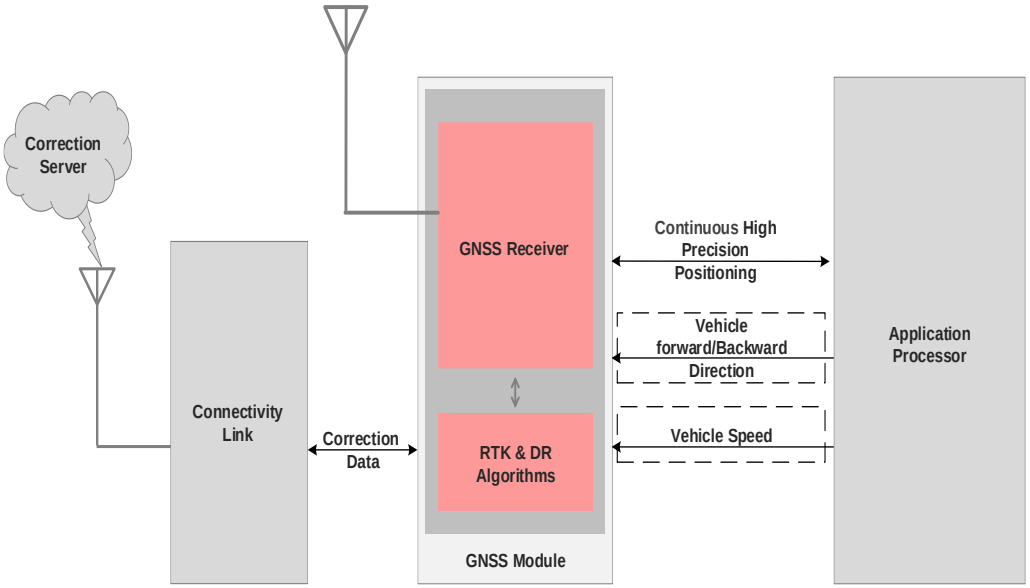


Figure 5: DR + RTK Application Architecture

1.3.2. Application Scenarios of DR + RTK

The main application scenarios of DR+RTK include: high-level autonomous driving, Shuttle Bus, automatic inspection robots, etc.

Figure 6: Autonomous Vehicle

2 DR

2.1. Orientation

The LC29H (BA) and LC29H (CA) modules are designed to work on two-wheel or four-wheel vehicles. Both modules integrate an IMU as well as the GNSS receiver. Therefore, you must ensure that the device incorporating the module is firmly fixed to vehicle body. No relative movement is allowed between vehicle and device and maximum isolation from shock or vibration must be applied. Manually holding the device is not acceptable. The best way to guarantee good installation is to firmly screw the device down to the vehicle frame. Mounting location should permit easy access to power supply and GNSS antenna, and should not be exposed to excessive heat.

Definitions of reference frame axes:

- X-axis points towards the right of the vehicle.
- Y-axis points towards the front of the vehicle.
- Z-axis points towards the roof of the vehicle.

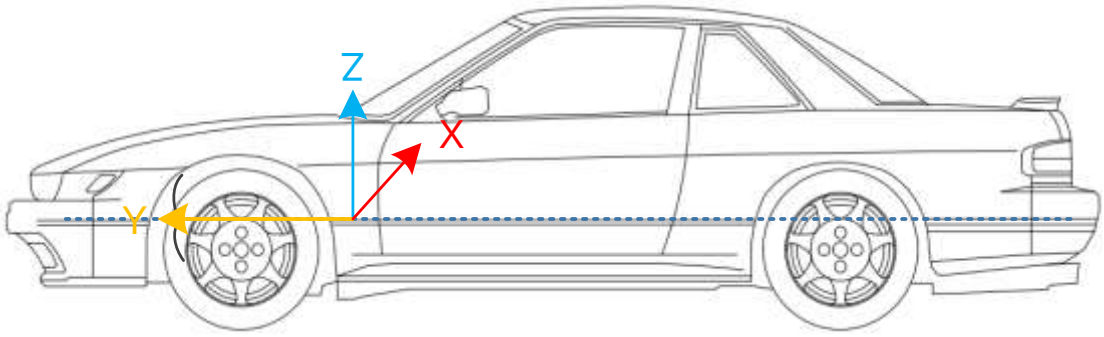


Figure 7: Reference Frame

Module orientation is shown below:

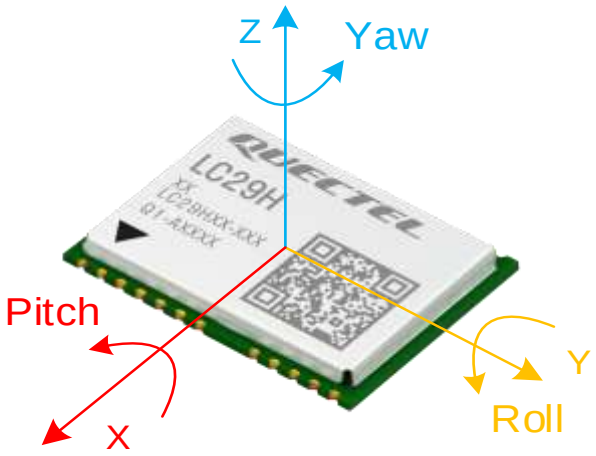


Figure 8: Module Orientation

2.2. DR Hot Start

For this function, the default configuration will automatically recognize the scene. When the vehicle comes to a complete stop, it will wait for 10 seconds and then shut down. After powering on again, the current position coordinates will be output without recalibrating the DR. A typical application scenario is an underground garage scenario, but all scenarios cannot be fully automatically recognized, so the command is opened to support the client to proactively enable this function when the current scenario requires it. For the command, refer to *Chapter 4.1.17 PQTMDRSAVE*. The prerequisite for this function to take effect is that the installation location does not change before and after the module is restarted. If the installation location is changed, it will not take effect. And after starting, the DR needs to be recalibrated, and the data needs to be cleared before testing. For the command, refer to *Chapter 4.1.18 PQTMDRCLR*. Otherwise trajectory abnormalities may occur before the DR is fully calibrated. Configure this feature can refer to *Chapter 4.1.15 PQTMCFGDRHOT*.

NOTE

Currently, since there is no external RTC connected, when the module enters a scene with GNSS signals, the timestamp may output errors or even go backwards.

2.3. Temperature Compensation

If the module working environment temperature is more than 25 degrees different from normal temperature, this function can be turned on. Before using this function, it needs to be run in the open air for about half an hour (stationary or running) to satisfy this function. The product itself needs to undergo complete high and low temperature changes (and the temperature change must go through a process greater than 25 degrees, such as from low temperature to high temperature or high temperature to low temperature). It is recommended not to change the installation position during each test after the module is fixedly installed, so as to accumulate the performance of different temperature changes and maximize the performance of this function. Configure this feature can refer to *Chapter 4.1.13 PQTMCFGIMUTC*.

2.4. DR Calibration Data Saving

The usage scenario of this function is that after the module completes DR calibration, in the scenario where there is a GNSS signal, the vehicle stops and sends the PQTMDRSAVE (see *Chapter 4.1.17 PQTMDRSAVE* for details) command to restart the module. After the module is restarted, the effect is consistent with DR hot start (see *Chapter 2.2 DR Hot Start* for details). There is no need to recalibrate the module and the position information is output immediately. The prerequisite for this function to take effect is that the installation position and vehicle position have not changed before and after the module is restarted. If changed, it will not take effect. And after starting, the DR needs to be recalibrated, and the data needs to be cleared before testing, command refer to *Chapter 4.1.18 PQTMDRCLR*, otherwise trajectory abnormalities may occur before the DR is fully calibrated.

2.5. Module Mounting

The LC29H series supports the free-mounting function and has no requirements for module installation, which means that the user does not need to configure the installation angle of the module, but only needs to confirm that the module is firmly mounted on the body and has no relative movement, and avoids the vehicle being placed in places such as frequent vibration and high temperature.

2.5.1. Mounting on Four-Wheel Vehicle

There are no mounting direction and angle limitations for mounting the Quectel LC29H (BA) or LC29H (CA) module on the four-wheel vehicle. The reference model is as follows:

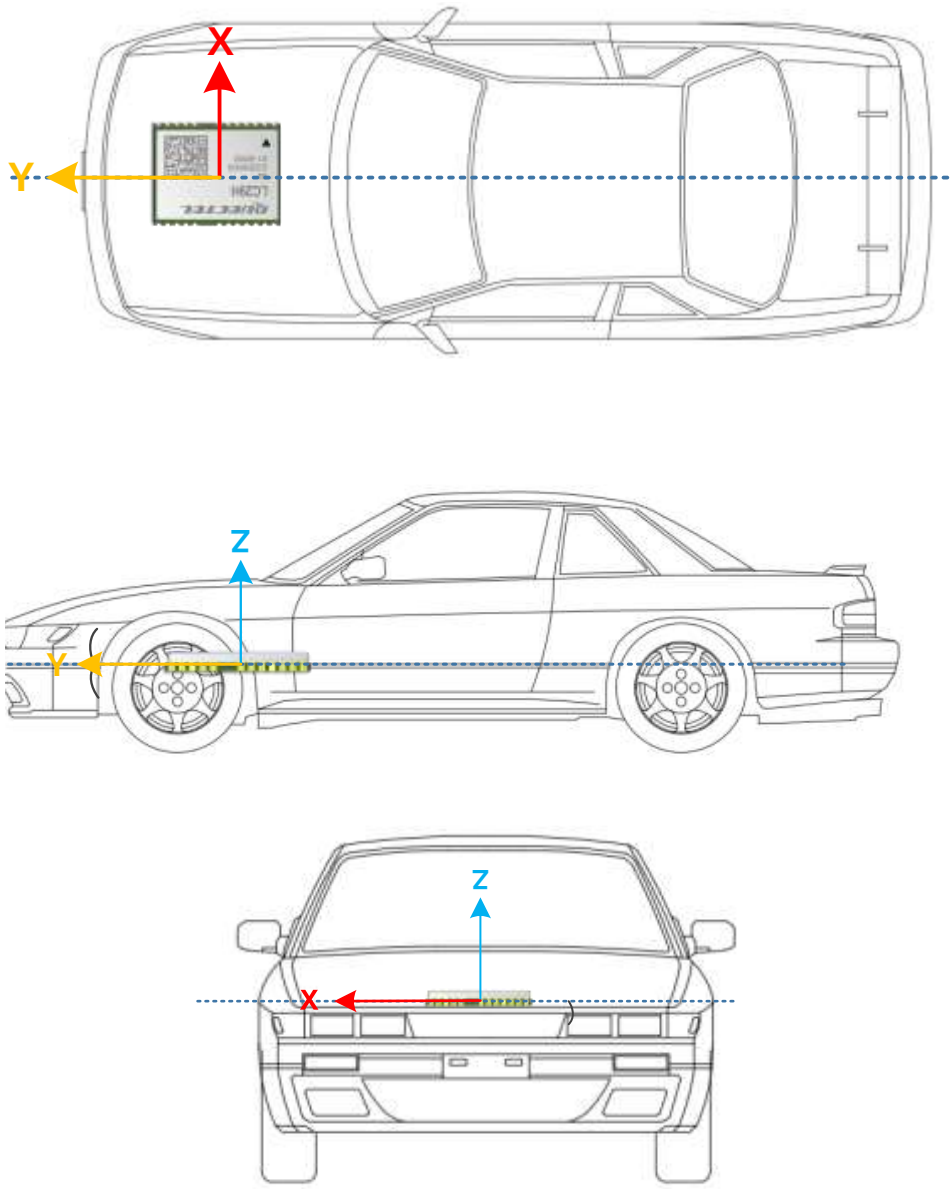
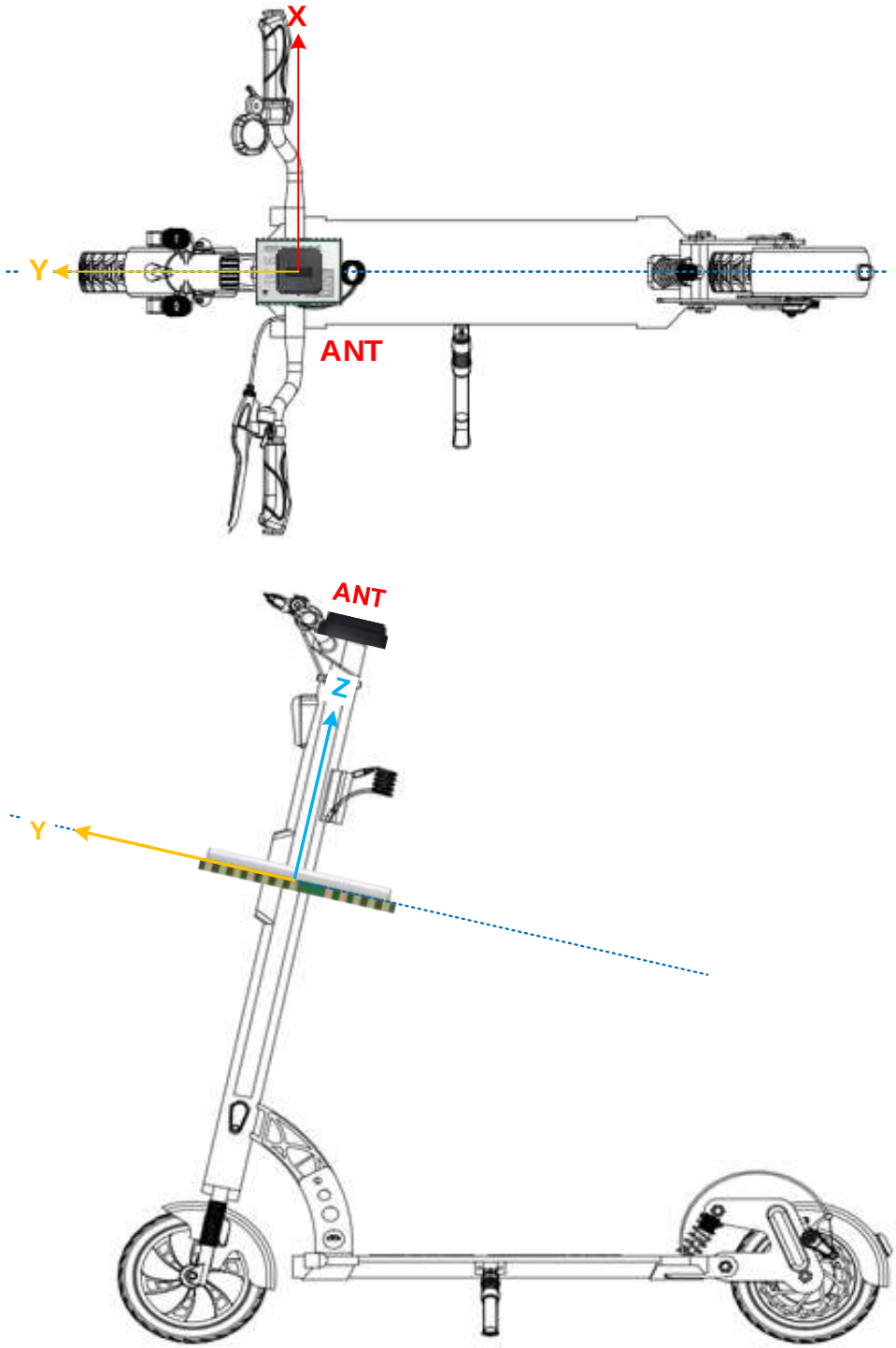


Figure 9: Module Mounting Example (4 wheel)

2.5.2. Mounting on Two-Wheel Vehicle

Recommendations for installing Quectel LC29H (BA) or LC29H (CA) modules to two-wheel vehicle. The reference model is as follows:


X
Y
ANT
ANT
Z
Y

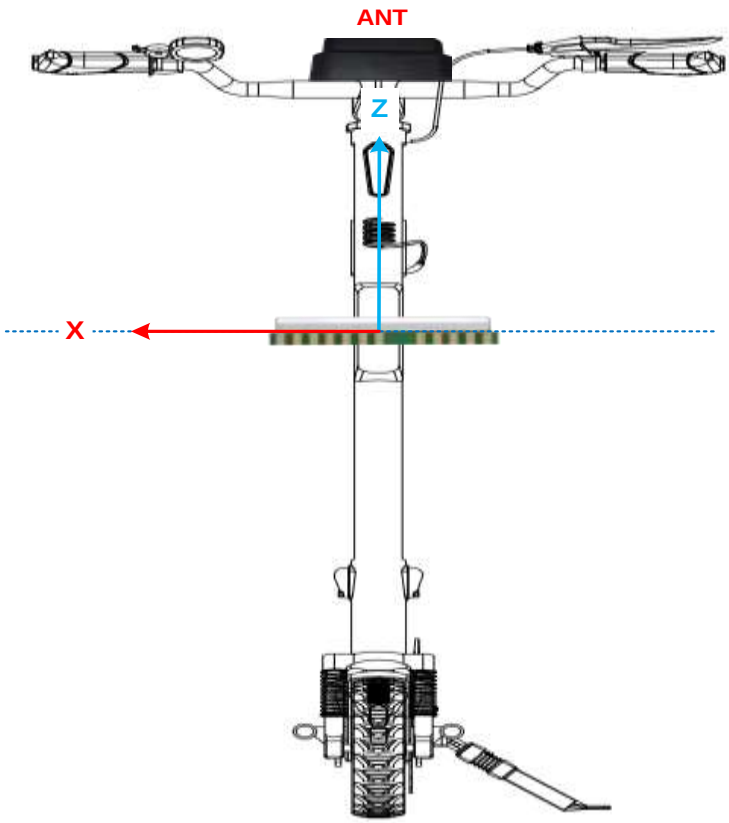


Figure 10: Module Mounting Example (2 wheel)

NOTE

1. Firmly affix the device incorporating the module to vehicle body. Select a structurally sound location that is not prone to flexing (no relative motion). GNSS antennas, modules and other supporting sensors should be unified on the same body frame.
2. Both two-wheel and four-wheel installations require relatively fixed rod arm displacements of the antenna and module. The module is calculated at Auto level, and manual calculation of rod arm information is not supported at the moment.
3. Two wheelers (electric scooters) work with odometry measurements such as wheel ticks or wheel speeds (if you want to use ADR). Ideally, the odometry data should come from the rear wheel when the module is in the deck and from the front wheel when the module is in the handlebar.
4. It is not recommended to install the module at the handlebars, otherwise navigation performance will be degraded. The reasons are as follows. Firstly, the module is non-fixed when installed on the handlebars, and it is easy to have relative displacement with the wheel speed center, which does not meet the installation rules. Secondly, the module is installed at the handlebars, which is prone to noise and will affect the measurement results.
5. GNSS antennas are mounted on IOT or handlebars, and the antennas should be straight up to the sky, which is conducive to the antenna receiving signals and ensuring positioning performance. The module is recommended to be installed inside the IOT, and if ADR is to be adopted, the wheel speed is injected into the corresponding vehicle speed.

2.6. DR Calibration

For the DR functional module, calibration is a key step to achieve DR extrapolation. The trajectory of the DR will be accurate only if the module is accurately calibrated.

So, the module needs to be calibrated before the DR function comes into play.

The DR calibration steps are as listed below:

Step 1: Fix the device incorporating the module on the vehicle frame firmly. Any displacement, rotation or tilt of the device relative to vehicle plane, however small, may cause performance issues and/or void the calibration.

Step 2: Calibration should be performed under good GNSS signal and clear sky conditions.

Step 3: Power up the module, then start the vehicle on a plain surface.

Step 4: Drive at a speed of more than 2 m/s, and perform 3–4 turning movements. The module will start self-calibration, which will be completed in approximately 3 minutes.

Step 5: The calibration process ends when **<CalState>** of **$PQTMDRCAL** message value is 2 (DR is fully calibrated). See *Chapter 4.1.1 PQTMDRCAL* for details about the message.

NOTE

1. If the wheel speed sensor of the vehicle is connected to the module, make sure that its precision is at least 0.05 m/tick.
2. The module will exit GNSS + DR or DR only mode in the following scenarios:
   - No speed data injection for 5 seconds in ADR mode;
   - Handling scenario;
   - High-speed elevator scenario;
   - IMU data interruption;
   - Inadequate inertial navigation calibration before entering a tunnel or underground garage;
   - Exceeding the set running time or distance of the receiver in environments with nonexistent GNSS coverage (See *Chapter 4.1.12 PQTMCFGDRRTD*).

3 RTK

3.1. RTCM Input

Quectel LC29H (BA), LC29H (DA) and LC29H (EA) modules support the RTCM 10403.3 input messages listed in the table below.

Table 1: Supported RTCM Input Messages

Message Type	Description
1005	Stationary RTK Reference Station ARP
1006	Stationary RTK Reference Station ARP with Antenna Height
1074	GPS MSM4
1075	GPS MSM5
1077	GPS MSM7
1084	GLONASS MSM4
1085	GLONASS MSM5
1087	GLONASS MSM7
1094	Galileo MSM4
1095	Galileo MSM5
1097	Galileo MSM7
1114	QZSS MSM4
1115	QZSS MSM5
1117	QZSS MSM7
1124	BDS MSM4
1125	BDS MSM5

Message Type	Description
1127	BDS MSM7

3.2. RTK Application Description

3.2.1. RTK Implementation with QGNSS

To implement RTK function by LC29H module, the host should have the ability to connect to a RTK server and get the correction data and then inject to the module. This chapter mainly introduce that how to implement the RTK with QGNSS. The premise is that you have a RTK server account, both the Ntrip account or QuecRTK account are available for LC29H module.

Preparation:

- A PC with Windows system (Windows 10 or higher versions).
- A LC29H EVB with USB cable, and dual band GNSS antenna under open sky.
- QGNSS tool: V1.9 or higher versions.

Click "**NTRIP**" in the "**Tools**" tab drop-down menu to open "**NTRIP Client**" window.

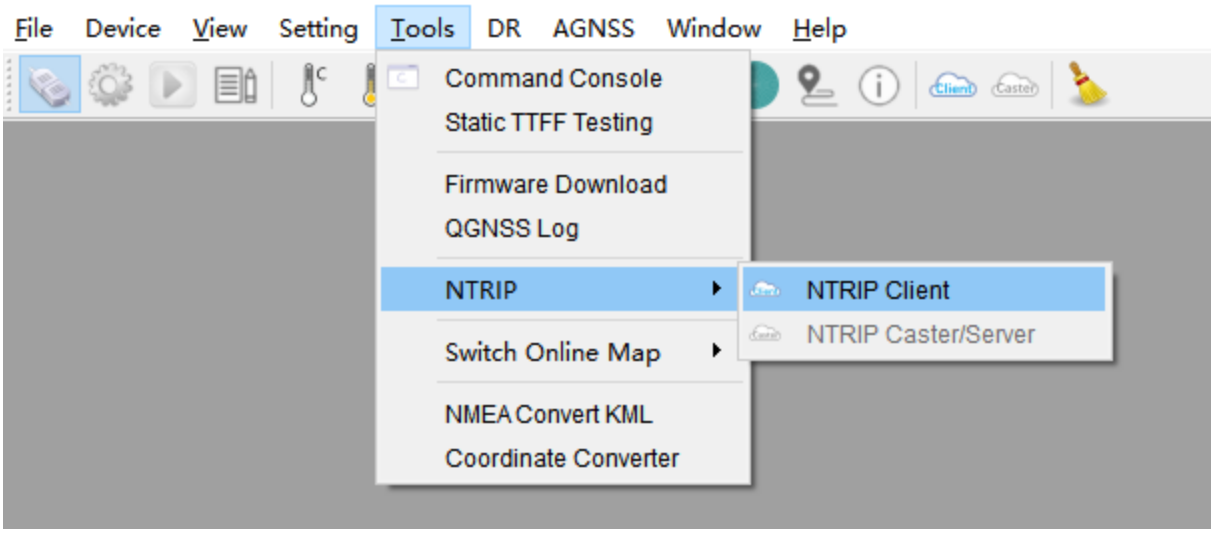


Figure 11: Open NTRIP Client

3.2.1.1. NTRIP Client

Use an NTRIP (V1.0) Client to connect to a standard NTRIP Caster, and follow these steps:

Step 1 Enter the Address, Port, Username and Password. Contact Quectel Technical Support to get the Username and Password if necessary.

Step 2 Click "**Update NTRIP source table**" and wait for the server to return mount point information.

Step 3 Select "**NTRIP mount point**".

Step 4 Enter "**Request Interval**".

Step 5 Tick the checkbox next to "**Use manual position**" and a window as shown in *Figure 13: Set Manual Position* pops up for entering the relevant data; otherwise the fixed position of the module will be used.

Step 6 Turn on the "**Connect to Host**" switch.

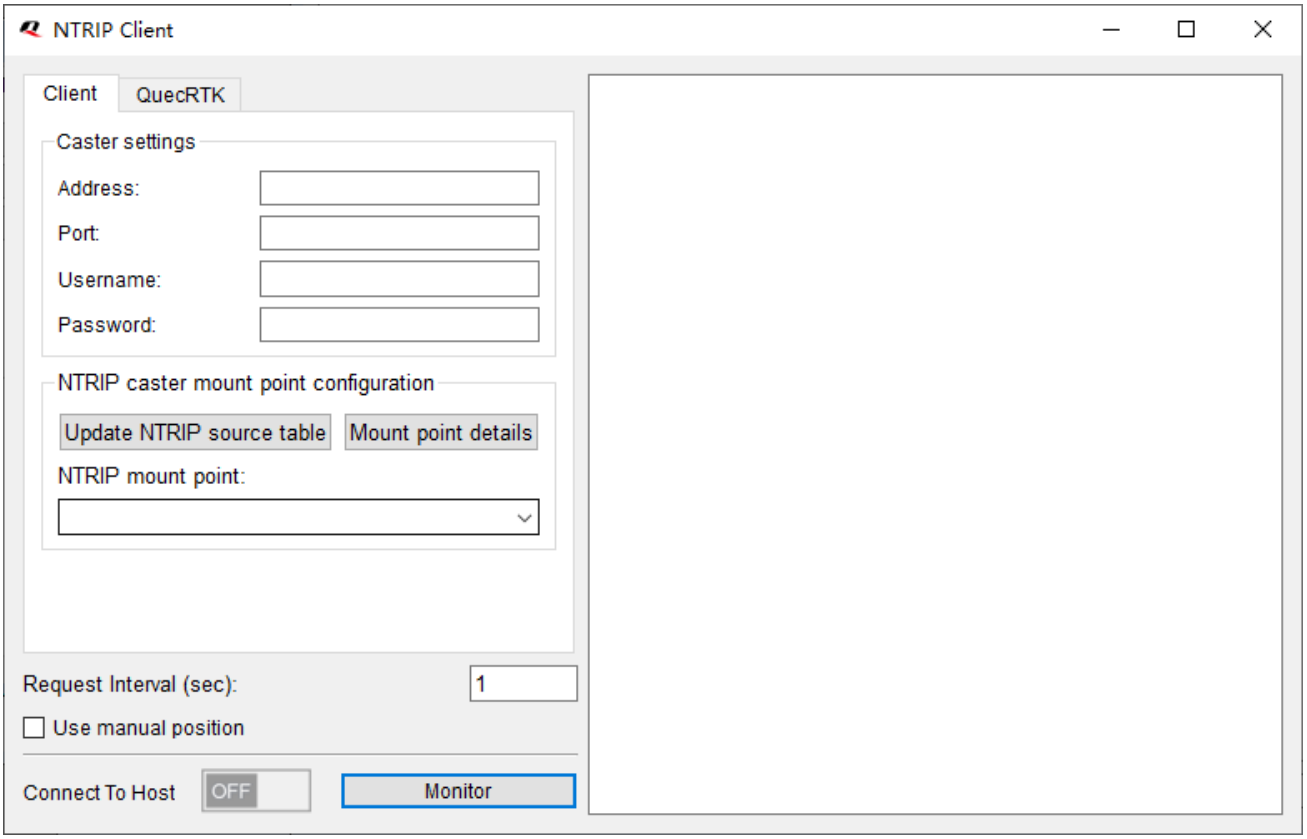


Figure 12: NTRIP Client

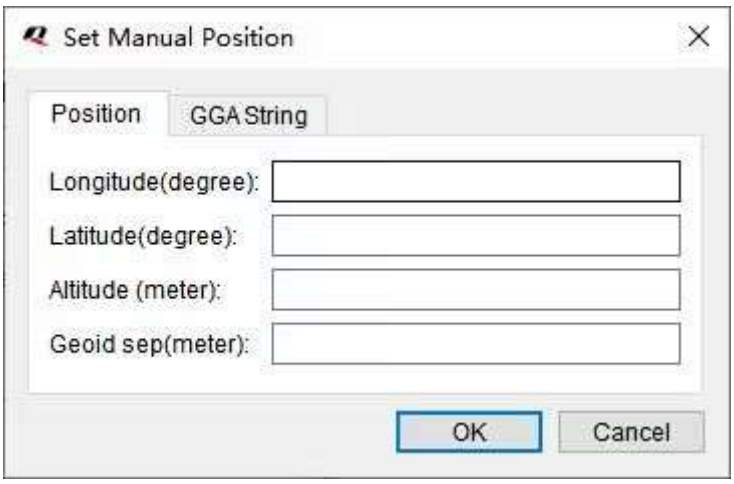


Figure 13: Set Manual Position

3.2.1.2. QuecRTK Client

QuecRTK is a high-precision RTK positioning and navigation service provided by Quectel, based on network RTK technology and supporting multiple constellations and frequency bands. The positioning accuracy can reach centimeter-level. By following the steps below, use QuecRTK to improve the module's positioning accuracy:

Step 1 Enter the MCC, Company ID, Device ID and Licence Key. Contact Quectel Technical Support to get the Device ID and Licence Key if necessary.

Step 2 Enter "**Request Interval**".

Step 3 Tick the checkbox next to "**Use manual position**" and a window as shown in *Figure 13: Set Manual Position* pops up for entering the relevant data; otherwise the fixed position of the module will be used.

Step 4 Turn on the "**Connect to Host**" switch.

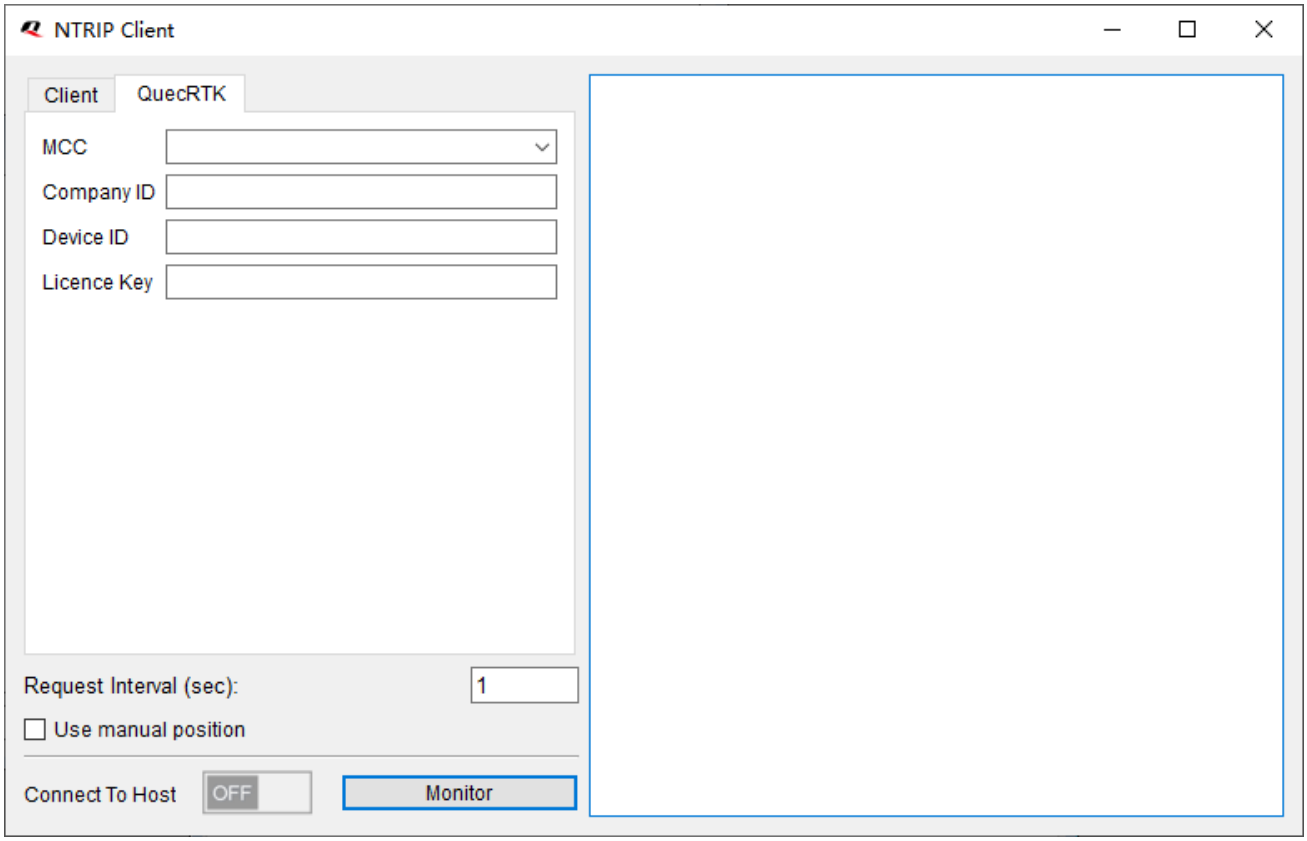


Figure 14: QuecRTK Client

Click "**Monitor**" to open the Data Monitor dialog box where you can view the differential correction data sent by the server.

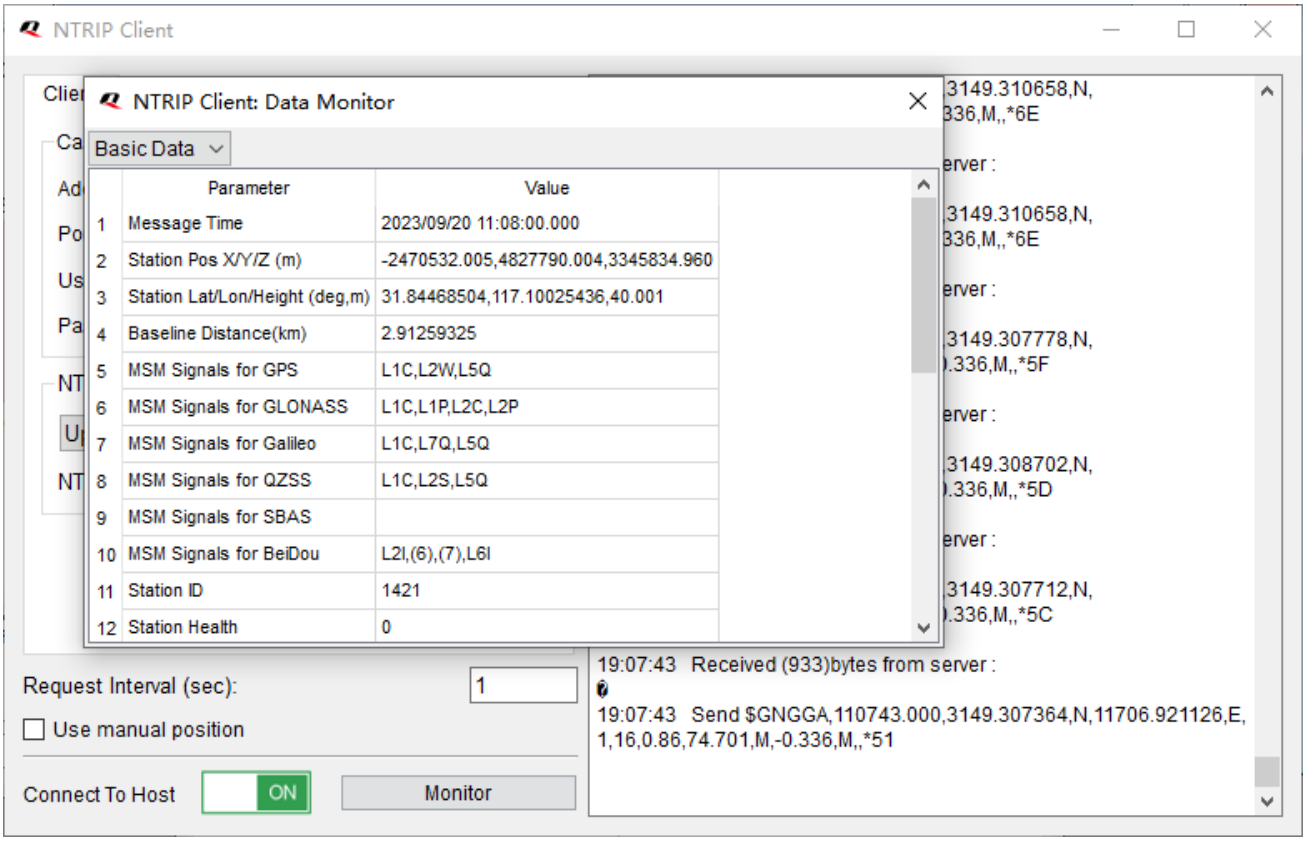


Figure 15: RTK Data Monitor

4 Related Messages

4.1. PQTM Messages

This chapter outlines the Quectel DR related PQTM (proprietary NMEA) messages supported by the Quectel LC29H (BA) and LC29H(CA) modules.

Table 2: Error Codes

Field	Format	Unit	Description
<ErrCode>	Numeric	-	Error code. 1 = Invalid parameters. 2 = Execution failed.

4.1.1. PQTMDRCAL

Indicates the DR calibration state.

Type:

Output

Synopsis:

```
$PQTMDRCAL,<MsgVer>,<CalState>,<NavType>*<Checksum><CR><LF>
```

Parameter:

Field	Format	Unit	Description
<MsgVer>	Numeric	-	Message version. 1 = Version 1 (Always 1 for this version.)
<CalState>	Numeric	-	DR calibration state. 0 = Not calibrated 1 = DR is lightly calibrated 2 = DR is fully calibrated 3 = DR is fully calibrated with high precision heading

Field	Format	Unit	Description
<NavType>	Numeric	-	Navigation type. 0 = No position 1 = GNSS only 2 = DR only 3 = Combination (GNSS + DR)

Example:

```
$PQTMDRCAL,1,0,1*5C
```

4.1.2. PQTMIMUTYPE

Outputs the IMU type once after each boot-up.

Type:

Output

Synopsis:

```
$PQTMIMUTYPE,<MsgVer>,<Type>*<Checksum><CR><LF>
```

Parameter:

Field	Format	Unit	Description
<MsgVer>	Numeric	-	Message version. 1 = Version 1 (Always 1 for this version.)
<Type>	Numeric	-	IMU type. 0 = IMU error 1 = LSM6DSR 2 = ICM-40608 3 = BMI160 4 = ICM-42670-P

Example:

```
$PQTMIMUTYPE,1,2*52
```

4.1.3. PQTMVEHMSG

Inputs/outputs vehicle information.

Type:

Input/output

Synopsis:

```
$PQTMVEHMSG,<MsgType>,<Timestamp>,<Par1>[,<Par2>,…,<ParN>]*<Checksum><CR><LF>
```

Parameter:

Field	Format	Unit	Description
<MsgType>	Numeric	-	Type of message input/output via UART interface. 1 = Vehicle speed (in m/s) 2 = Cumulative wheel tick 3 = Speeds of four wheels (in m/s) 4 = Cumulative wheel ticks of four wheels
<Timestamp>	Numeric	Millisecond	Timestamp since power-on. 32-bit unsigned integer. Always 0 when this message is input.
<Par1> to <ParN>	Numeric	-	Vehicle information. This field varies with the message type. See *Chapter 4.1.3.1 If <MsgType> = 1* to *4.1.3.4 If <MsgType> = 4* for details.

NOTE

<MsgType> can only be 2 for LC29H (BA) and LC29H (CA) with software versions dedicated for two-wheel vehicles. Contact Quectel Technical Support (support@quectel.com) for details about the software versions.

4.1.3.1. If <MsgType> = 1

Synopsis:

```
$PQTMVEHMSG,1,<Timestamp>,<VehSpeed>*<Checksum><CR><LF>
```

Parameter:

Field	Format	Unit	Description
<Timestamp>	Numeric	Millisecond	Timestamp since power-on. 32-bit unsigned integer. Always 0 when this message is input.
<VehSpeed>	Numeric	m/s	Speed. Range: -100 to 100.

Result:

Returns the input vehicle speed with timestamp:

```
$PQTMVEHMSG,1,<Timestamp>,<VehSpeed>*<Checksum><CR><LF>
```

Example:

```
$PQTMVEHMSG,1,0,3.6*1C
$PQTMVEHMSG,1,3748292,3.6*1D
```

4.1.3.2. If <MsgType> = 2

Synopsis:

```
$PQTMVEHMSG,2,<Timestamp>,<WheelTickCNT>,<FWD_Ind>*<Checksum><CR><LF>
```

Parameter:

Field	Format	Unit	Description
<Timestamp>	Numeric	Millisecond	Timestamp since power-on. 32-bit unsigned integer. Always 0 when this message is input.
<WheelTickCNT>	Numeric	Tick	Cumulative wheel ticks.
<FWD_Ind>	Numeric	-	Forward/backward indicator. 0 = Invalid state 1 = Forward 2 = Backward

Result:

Returns the input cumulative wheel ticks with timestamp:

```
$PQTMVEHMSG,2,<Timestamp>,<WheelTickCNT>,<FWD_Ind>*<Checksum><CR><LF>
```

Example:

```
$PQTMVEHMSG,2,0,100,1*18
$PQTMVEHMSG,2,153954,100,1*27
```

NOTE

1. When inputting cumulative wheel ticks through UART interface, make sure the input rate is at least 10 Hz.
2. For LC29H (BA) and LC29H (CA) with software versions dedicated for two-wheel vehicles:

1) Keep **<FWD_Ind>** always 1;
2) **$PQTMVEHMSG** will not be returned, but the input **<WheelTickCNT>** can be found in **$PQTMIMU**.

Contact Quectel Technical Support (support@quectel.com) for details about the software versions.

4.1.3.3. If <MsgType> = 3

Synopsis:

$PQTMVEHMSG,3,<Timestamp>,<LF_Spd>,<RF_Spd>,<LR_Spd>,<RR_Spd>*<Checksum><CR><LF>

Parameter:

Field	Format	Unit	Description
<TimeStamp>	Numeric	Millisecond	Timestamp since power-on. 32-bit unsigned integer. Always 0 when this message is input.
<LF_Spd>	Numeric	m/s	Left front wheel speed. Range: -100 to 100.
<RF_Spd>	Numeric	m/s	Right front wheel speed. Range: -100 to 100.
<LR_Spd>	Numeric	m/s	Left rear wheel speed. Range: -100 to 100.
<RR_Spd>	Numeric	m/s	Right rear wheel speed. Range: -100 to 100.

Result:

Returns the input speeds of four wheels with timestamp:

$PQTMVEHMSG,3,<Timestamp>,<LF_Spd>,<RF_Spd>,<LR_Spd>,<RR_Spd>*<Checksum><CR><LF>

Example:

```
$PQTMVEHMSG,3,0,3.6,3.6,3.6,3.6*19
$PQTMVEHMSG,3,3748292,3.6,3.6,3.6,3.6*18
```

4.1.3.4. If <MsgType> = 4

Synopsis:

$PQTMVEHMSG,4,<Timestamp>,<LF_TickCNT>,<RF_TickCNT>,<LR_TickCNT>,<RR_TickCNT><FWD_Ind>*<Checksum><CR><LF>

Parameter:

Field	Format	Unit	Description
<Timestamp>	Numeric	Millisecond	Timestamp since power-on. 32-bit unsigned integer. Always 0 when this message is input.
<LF_TickCNT>	Numeric	Tick	Left front wheel tick count.
<RF_TickCNT>	Numeric	Tick	Right front wheel tick count.
<LR_TickCNT>	Numeric	Tick	Left rear wheel tick count.
<RR_TickCNT>	Numeric	Tick	Right rear wheel tick count.
<FWD_Ind>	Numeric	-	Forward/backward indicator. 0 = Invalid state 1 = Forward 2 = Backward

Result:

Returns the input cumulative wheel ticks of four wheels with timestamp:

```
$PQTMVEHMSG,4,<Timestamp>,<LF_TickCNT>,<RF_TickCNT>,<LR_TickCNT>,<RR_TickCNT><FWD_Ind>*<Checksum><CR><LF>
```

Example:

```
$PQTMVEHMSG,4,0,100,100,100,100,1*03
$PQTMVEHMSG,4,153954,100,100,100,100,1*3C
```

4.1.4. PQTMSAVEPAR

Saves the configurations set via **$PQTM** commands or **$PAIR6010** into NVM. Reset the module after executing this command.

Type:

Command

Synopsis:

```
$PQTMSAVEPAR*<Checksum><CR><LF>
```

Parameter:

None

Result:

- If successful, the module returns:

```
$PQTMSAVEPAR,OK*72
```

- If failed, the module returns:

```
$PQTMSAVEPAR,ERROR,<ErrCode>*<Checksum><CR><LF>
```

For details about <**ErrCode**>, see *Table 2: Error Codes*.

Example:

```
$PQTMSAVEPAR*5A
$PQTMSAVEPAR,OK*72
```

4.1.5. PQTMRESTOREPAR

Restore all DR-related configurations to their default values. Reset the module after executing this command.

Type:

Command

Synopsis:

```
$PQTMRESTOREPAR*<Checksum><CR><LF>
```

Parameter:

None

Result:

- If successful, the module returns:

```
$PQTMRESTOREPAR,OK*3B
```

- If failed, the module returns:

```
$PQTMRESTOREPAR,ERROR,<ErrCode>*<Checksum><CR><LF>
```

For details about <**ErrCode**>, see *Table 2: Error Codes*.

Example:

```
$PQTMRESTOREPAR*13
$PQTMRESTOREPAR,OK*3B
```

4.1.6. PQTMINS

Outputs navigation results.

Type:

Output

Synopsis:

```
$PQTMINS,<Timestamp>,<SolType>,<Lat>,<Lon>,<Height>,<VEL_N>,<VEL_E>,<VEL_D>,<Roll>,<Pitch>,<Yaw>*<Checksum><CR><LF>
```

Parameter:

Field	Format	Unit	Description
<Timestamp>	Numeric	Millisecond	Timestamp since power-on. 32-bit unsigned integer.
<SolType>	Numeric	-	Solution type. 0 = DR not ready. Roll and pitch ready. 1 = DR not ready. GNSS, roll, pitch, and relative heading ready. 2 = GNSS + DR mode. DR calibrated. 3 = DR only mode.
<Lat>	Numeric	Degree	Latitude.
<Lon>	Numeric	Degree	Longitude.
<Height>	Numeric	Meter	Height.
<VEL_N>	Numeric	m/s	Northward velocity.
<VEL_E>	Numeric	m/s	Eastward velocity.
<VEL_D>	Numeric	m/s	Downward velocity.
<Roll>	Numeric	Degree	Roll angle.
<Pitch>	Numeric	Degree	Pitch angle.
<Yaw>	Numeric	Degree	Heading angle.

Example:

```
$PQTMINS,240951,1,31.82222216,117.11578436,62.555605,-0.004233,0.005535,-0.004011,0.00,0.00,127.41*40
```

NOTE

1. **<Yaw>** is scaled from -180.0 to 179.9 with a wrap-around to 0.0 at +180.0. -180.0/ +180 = South, 0.0 = North, +90.0 = East, and -90.0 = West.
2. This message is only supported by LC29H (BA) and LC29H (CA) with software versions dedicated for two-wheel vehicles. Contact Quectel Technical Support (support@quectel.com) for details about the software versions.

4.1.7. PQTMIMU

Outputs the IMU raw data: acceleration, angular rate, and hardware wheel ticks. These values should match module frame, and see *Figure 7: Reference Frame* for details.

Type:

Output

Synopsis:

```
$PQTMIMU,<Timestamp>,<ACC_X>,<ACC_Y>,<ACC_Z>,<AngRate_X>,<AngRate_Y>,<AngRate_Z>,<WheelTickCNT>,<LastTick_Timestamp>*<Checksum><CR><LF>
```

Parameter:

Field	Format	Unit	Description
<Timestamp>	Numeric	Millisecond	Timestamp since power-on. 32-bit unsigned integer.
<ACC_X>	Numeric	m/s^2	Acceleration in X-axis direction.
<ACC_Y>	Numeric	m/s^2	Acceleration in Y-axis direction.
<ACC_Z>	Numeric	m/s^2	Acceleration in Z-axis direction.
<AngRate_X>	Numeric	deg/s	Angular rate in X-axis direction.
<AngRate_Y>	Numeric	deg/s	Angular rate in Y-axis direction.
<AngRate_Z>	Numeric	deg/s	Angular rate in Z-axis direction.
<WheelTickCNT>	Numeric	Tick	Cumulative wheel ticks.
<LastTick_Timestamp>	Numeric	Millisecond	Last tick timestamp.

Example:

```
$PQTMIMU,45454,-1.356730,-0.210568,9.757930,0.564879,0.549612,-0.412209,0,0*77
```

NOTE

This message is only supported by LC29H (BA) and LC29H (CA) with software versions dedicated for two-wheel vehicles. Contact Quectel Technical Support (support@quectel.com) for details about the software versions.

4.1.8. PQTMGPS

Outputs the position status in GNSS only mode.

Type:

Output

Synopsis:

```
$PQTMGPS,<Timestamp>,<TOW>,<Lat>,<Lon>,<Altitude>,<Speed>,<Yaw>,<Accuracy>,<HDOP>,<PDOP>,<NumSatUsed>,<FixMode>*<Checksum><CR><LF>
```

Parameter:

Field	Format	Unit	Description
<Timestamp>	Numeric	Millisecond	Timestamp since power-on. 32-bit unsigned integer.
<TOW>	Numeric	Second	Time of week.
<Lat>	Numeric	Degree	Latitude.
<Lon>	Numeric	Degree	Longitude.
<Altitude>	Numeric	Meter	Altitude.
<Speed>	Numeric	m/s	Ground speed (two-dimensional).
<Yaw>	Numeric	Degree	Heading of vehicle (two-dimensional).
<Accuracy>	Numeric	Meter	Horizontal accuracy estimate.
<HDOP>	Numeric	-	Horizontal dilution of precision.
<PDOP>	Numeric	-	Position dilution of precision.
<NumSatUsed>	Numeric	-	Number of satellites used in navigation.
<FixMode>	Numeric	-	Fix mode. 0 = No fix 2 = 2D fix

Field	Format	Unit	Description
			3 = 3D fix

Example:

```
$PQTMGPS,86139,94183,31.82218794,117.11579022,65.755080,0.027,94.68,2.533952,0.555471,0.886183,29,3*6B
```

NOTE

1. This message is only supported by LC29H (BA) and LC29H (CA) with software versions dedicated for two-wheel vehicles. Contact Quectel Technical Support (support@quectel.com) for details about the software versions.
2. When there is no differential data, the **$PQTMGPS** output is only the PVT result, otherwise, the **$PQTMGPS** output is only the RTK result.
3. **<Yaw>** is scaled from -180.0 to 179.9 with a wrap-around to 0.0 at +180.0. -180.0/ +180 = South, 0.0 = North, +90.0 = East, and -90.0 = West.

4.1.9. PQTMCFGEINSMSG

Sets/gets **$PQTMINS**, **$PQTMIMU** and **$PQTMGPS** message configurations.

Type:

Set/get

Synopsis:

```
//Set message configurations:
$PQTMCFGEINSMSG,<Type>,<INS_Enabled>,<IMU_Enabled>,<GPS_Enabled>,<Rate>*<Checksum><CR><LF>
//Get message configurations:
$PQTMCFGEINSMSG,<Type>*<Checksum><CR><LF>
```

Parameter:

Field	Format	Unit	Description
<Type>	Numeric	-	Set/get message configurations. 0 = Get 1 = Set
<INS_Enabled>	Numeric	-	Enable/disable the output of **$PQTMINS** message. 0 = Disable 1 = Enable

Field	Format	Unit	Description
<IMU_Enabled>	Numeric	-	Enable/disable the output of **$PQTMIMU** message. 0 = Disable 1 = Enable
<GPS_Enabled>	Numeric	-	Enable/disable the output of **$PQTMGPS** message. 0 = Disable 1 = Enable
<Rate>	Numeric	Hz	Set the output rate of **$PQTMINS**, **$PQTMIMU** or **$PQTMGPS** message. It can be set to 0, 1, 2, 4, 5, 10, 20, 50, or 100. For **$PQTMGPS**, the output rate is fixed at 1 Hz. For **$PQTMINS**, when **<Rate>** is set to be greater than 10, the message will be output at 10 Hz. For **$PQTMIMU**, the output rate can be set to 0, 1, 10, 20, 50, 100 Hz.

Result:

- If successful, the module returns:

```
//Set:
$PQTMCFGEINSMSGOK*16
//Get:
$PQTMCFGEINSMSG,<Type>,<INS_Enabled>,<IMU_Enabled>,<GPS_Enabled>,<Rate>*<Checksum>
<CR><LF>
```

- If failed, the module returns:

```
$PQTMCFGEINSMSGERROR*4A
```

Example:

```
//Set message configurations:
$PQTMCFGEINSMSG,1,1,1,1,10*3F
$PQTMCFGEINSMSGOK*16
//Get message configurations:
$PQTMCFGEINSMSG,0*0E
$PQTMEINSMSG,0,1,1,1,10*3E
```

NOTE

1. Send **$PQTMSAVEPAR*5A** and reset the module for **$PQTMCFGEINSMSG** to take effect.
2. This command is only supported by LC29H (BA) and LC29H (CA) with software versions dedicated for two-wheel vehicles. Contact Quectel Technical Support (support@quectel.com) for details about the software versions.

4.1.10. PQTMVEHMOT

Outputs vehicle motion information after DR calibration.

Type:

Output

Synopsis:

```
$PQTMVEHMOT,<MsgVer>,<PeakAcceleration>,<PeakAngularRate>*<Checksum><CR><LF>
```

Parameter:

Field	Format	Unit	Description
<MsgVer>	Numeric	-	Message version. 1 = Version 1 (Always 1 for this version.)
<PeakAcceleration>	Numeric	m/s^2	Peak acceleration of vehicle.
<PeakAngularRate>	Numeric	deg/s	Peak angular rate of vehicle.

Example:

```
$PQTMVEHMOT,1,0.288124,0.159930*0A
```

NOTE

This message is only supported by LC29H (BA) and LC29H (CA) with software versions dedicated for four-wheel vehicles. Contact Quectel Technical Support (support@quectel.com) for details about the software versions.

4.1.11. PQTMSENMSG

Outputs sensor information.

Type:

Output

Synopsis:

```
$PQTMSENMSG,<MsgVer>,<TimeStamp>,<Par1>[,<Par2>,…,<ParN>]*<Checksum><CR><LF>
```

Parameter:

Field	Format	Unit	Description
<MsgVer>	Numeric	-	Message version. 4 = IMU sensor data matching vehicle reference frame. See *Figure 7: Reference Frame* for details.
<TimeStamp>	Numeric	Millisecond	Timestamp since power-on. 32-bit unsigned integer.
<Par1> to <ParN>	Numeric	-	Sensor information. See *Chapter 4.1.11.1 If <MsgVer> = 4* for details.

4.1.11.1. If <MsgVer> = 4

Synopsis:

```
$PQTMSENMSG,4,<TimeStamp>,<IMU_Temp>,<IMU_GYRO_X>,<IMU_GYRO_Y>,<IMU_GYRO_Z>,<IMU_ACC_X>,<IMU_ACC_Y>,<IMU_ACC_Z>*<Checksum><CR><LF>
```

Parameter:

Field	Format	Unit	Description
<TimeStamp>	Numeric	Millisecond	Timestamp since power-on. 32-bit unsigned integer.
<IMU_Temp>	Numeric	Celsius	IMU temperature.
<IMU_GYRO_X>	Numeric	dps	IMU X-axis gyro value.
<IMU_GYRO_Y>	Numeric	dps	IMU Y-axis gyro value.
<IMU_GYRO_Z>	Numeric	dps	IMU Z-axis gyro value.
<IMU_ACC_X>	Numeric	g	IMU X-axis accelerometer value.
<IMU_ACC_Y>	Numeric	g	IMU Y-axis accelerometer value.
<IMU_ACC_Z>	Numeric	g	IMU Z-axis accelerometer value.

Example:

```
$PQTMSENMSG,4,1977253,29.830917,1.727613,0.015743,0.804347,-0.250096,-0.467039,10.444151*24
```

NOTE

This message is only supported by LC29H (BA) and LC29H (CA) with software versions dedicated for four-wheel vehicles. Contact Quectel Technical Support (support@quectel.com) for details about the software versions.

4.1.12. PQTMCFGDRRTD

Sets/gets the DR running time and distance in DR only mode. If the running time or distance of the receiver exceeds the set values, the receiver will exit DR only mode.

Type:

Set/Get

Synopsis:

```
//Set:
$PQTMCFGDRRTD,W,<Time>,<Dist>*<Checksum><CR><LF>
//Get:
$PQTMCFGDRRTD,R*<Checksum><CR><LF>
```

Parameter:

Field	Format	Unit	Description
<Time>	Numeric	Second	Configuration of DR running time. 0 = No limitation
<Dist>	Numeric	Meter	Configuration of DR running distance. 0 = No limitation

Result:

- If successful, the module returns:

```
//Set:
$PQTMCFGDRRTD,OK*<Checksum><CR><LF>
//Get:
$PQTMCFGDRRTD,OK,<Time>,<Dist>*<Checksum><CR><LF>
```

- If failed, the module returns:

```
$PQTMCFGDRRTD,ERROR,<ErrCode>*<Checksum><CR><LF>
```

For details about <**ErrCode**>, see *Table 2: Error Codes*.

Example:

```
//Set:
$PQTMCFGDRRTD,W,600,10000*72
$PQTMCFGDRRTD,OK*26
//Get:
$PQTMCFGDRRTD,R*70
$PQTMCFGDRRTD,OK,600,10000*21
```

4.1.13. PQTMCFGIMUTC

Sets/gets the IMU temperature compensation feature.

Type:

Set/Get

Synopsis:

```
//Set:
$PQTMCFGIMUTC,W,<State>*<Checksum><CR><LF>
//Get:
$PQTMCFGIMUTC,R*<Checksum><CR><LF>
```

Parameter:

Field	Format	Unit	Description
<State>	Numeric	-	State of IMU temperature compensation feature. 0 = Disabled 1 = Enabled

Result:

- If successful, the module returns:

```
//Set:
$PQTMCFGIMUTC,OK*<Checksum><CR><LF>
//Get:
$PQTMCFGIMUTC,OK,<State>*<Checksum><CR><LF>
```

- If failed, the module returns:

```
$PQTMCFGIMUTC,ERROR,<ErrCode>*<Checksum><CR><LF>
```

For details about <**ErrCode**>, see *Table 2: Error Codes*.

Example:

```
//Enable temperature compensation feature.
$PQTMCFGIMUTC,W,1*7A
$PQTMCFGIMUTC,OK*34

//Query temperature compensation state.
$PQTMCFGIMUTC,R*62
$PQTMCFGIMUTC,OK,1*29
```

NOTE

This message is only supported by LC29H (BA) and LC29H (CA) with software versions dedicated for four-wheel vehicles. Contact Quectel Technical Support (support@quectel.com) for details about the software versions.

4.1.14. PQTMDRPVA

Outputs the DR position, velocity and attitude.

Type:

Output

Synopsis:

```
$PQTMDRPVA,<MsgVer>,<Timestamp>,<Time>,<SolType>,<Lat>,<Lon>,<Alt>,<Sep>,<VelN>,<VelE>,<VelD>,<Spd>,<Roll>,<Pitch><Heading>*<Checksum><CR><LF>
```

Parameter:

Field	Format	Unit	Description
<MsgVer>	Numeric	-	Message version. 1 = Version 1 (Always 1 for this version.)
<Timestamp>	Numeric	Millisecond	Milliseconds since module turn-on. 32-bit unsigned integer.
<Time>	hhmmss.sss	-	UTC time. hh: Hours (00–23) mm: Minutes (00–59) ss: Seconds (00–59) sss: Decimal fraction of seconds

Field	Format	Unit	Description
<SolType>	Numeric	-	Solution type. 0 = No fix 1 = GNSS only 2 = Combination (GNSS + DR) 3 = DR only
<Lat>	Numeric	Degree	Latitude. Note that this field is empty in case of an invalid value.
<Lon>	Numeric	Degree	Longitude. Note that this field is empty in case of an invalid value.
<Alt>	Numeric	Meter	Altitude above mean-sea-level. Note that this field is empty in case of an invalid value.
<Sep>	Numeric	Meter	Geoidal separation (the difference between the WGS84 earth ellipsoid surface and the mean-sea-level surface). Note that this field is empty in case of an invalid value.
<VelN>	Numeric	m/s	North velocity. Note that this field is empty in case of an invalid value.
<VelE>	Numeric	m/s	East velocity. Note that this field is empty in case of an invalid value.
<VelD>	Numeric	m/s	Down velocity. Note that this field is empty in case of an invalid value.
<Spd>	Numeric	m/s	Ground speed. Note that this field is empty in case of an invalid value.
<Roll>	Numeric	Degree	Roll angle. Note that this field is empty in case of an invalid value. Range: -180.000000 to 180.000000
<Pitch>	Numeric	Degree	Pitch angle. Note that this field is empty in case of an invalid value. Range: -90.000000 to 90.000000
<Heading>	Numeric	Degree	Heading. Note that this field is empty in case of an invalid value. Range: 0.000000–360.000000

Example:

```
//No fix.
$PQTMDRPVA,1,1000,163355.000,0,,,,,,,,,*7C

//GNSS + DR fix.
$PQTMDRPVA,1,75000,083737.000,2,31.12738291,117.26372910,34.212,5.267,3.212,2.928,0.238,4.346,0.392663,1.300793,0.030088*5E
```

NOTE

This message is only supported by LC29H (BA) and LC29H (CA) with software versions dedicated for four-wheel vehicles. Contact Quectel Technical Support (support@quectel.com) for details about the software versions.

4.1.15. PQTMCFGDRHOT

Sets/gets the DR hot start feature.

Type:

Set/Get

Synopsis:

```
//Set:
$PQTMCFGDRHOT,W,<Mode>*<Checksum><CR><LF>
//Get:
$PQTMCFGDRHOT,R*<Checksum><CR><LF>
```

Parameter:

Field	Format	Unit	Description
<Mode>	Numeric	-	Enable/disable the DR hot start mode. 0 = Disable DR hot start. DR needs to be recalibrated after each power-up. 1 = Enable DR hot start. DR does not need to be recalibrated after each power-up, and the module outputs position information immediately after power-up in the case of no signal. 2 = Enable DR hot start. DR does not need to be recalibrated after each power-up, but the module does not output position information after power-up in the case of no signal.

Result:

- If successful, the module returns:

```
//Set:
$PQTMCFGDRHOT,OK*<Checksum><CR><LF>
//Get:
$PQTMCFGDRHOT,OK,<Mode>*<Checksum><CR><LF>
```

- If failed, the module returns:

```
$PQTMCFGDRHOT,ERROR,<ErrCode>*<Checksum><CR><LF>
```

For details about <**ErrCode**>, see *Table 2: Error Codes*.

Example:

```
//Set:
$PQTMCFGDRHOT,W,1*79
$PQTMCFGDRHOT,OK*37
//Get:
$PQTMCFGDRHOT,R*61
$PQTMCFGDRHOT,OK,1*2A
```

NOTE

This message is only supported by LC29H (BA) and LC29H (CA) with software versions dedicated for four-wheel vehicles. Contact Quectel Technical Support (support@quectel.com) for details about the software versions.

4.1.16. PQTMCFGDR

Sets/gets the DR state.

Type:

Set/Get

Synopsis:

```
//Set:
$PQTMCFGDR,W,<State>*<Checksum><CR><LF>
//Get:
$PQTMCFGDR,R*<Checksum><CR><LF>
```

Parameter:

Field	Format	Unit	Description
<State>	Numeric	-	Enable/disable the DR feature. 0 = Disable DR feature 1 = Enable DR feature

Result:

- If successful, the module returns:

```
//Set:
$PQTMCFGDR,OK*<Checksum><CR><LF>
//Get:
$PQTMCFGDR,OK,<State>*<Checksum><CR><LF>
```

- If failed, the module returns:

```
$PQTMCFGDR,ERROR,<ErrCode>*<Checksum><CR><LF>
```

For details about <**ErrCode**>, see *Table 2: Error Codes*.

Example:

```
//Set:
$PQTMCFGDR,W,1*2A
$PQTMCFGDR,OK*64
//Get:
$PQTMCFGDR,R*32
$PQTMCFGDR,OK,1*79
```

NOTE

This message is only supported by LC29H (BA) and LC29H (CA) with software versions dedicated for four-wheel vehicles. Contact Quectel Technical Support (support@quectel.com) for details about the software versions.

4.1.17. PQTMDRSAVE

This command saves DR navigation data, including current DR position, velocity, attitude.

Type:

Command

Synopsis:

```
$PQTMDRSAVE*<Checksum><CR><LF>
```

Parameter:

None.

Result:

- If successful, the module returns:

```
$PQTMDRSAVE,OK*<Checksum><CR><LF>
```

- If failed, the module returns:

```
$PQTMDRSAVE,ERROR,<ErrCode>*<Checksum><CR><LF>
```

For details about <**ErrCode**>, see *Table 2: Error Codes*.

Example:

```
$PQTMDRSAVE*0F
$PQTMDRSAVE,OK*27
```

4.1.18. PQTMDRCLR

This command is used to clear DR calibration data, including initial align, mis-align angle, lever arm, and the last DR position, velocity, attitude are also be cleared.

Type:

Command

Synopsis:

```
$PQTMDRCLR*<Checksum><CR><LF>
```

Parameter:

None.

Result:

- If successful, the module returns:

```
$PQTMDRCLR,OK*<Checksum><CR><LF>
```

- If failed, the module returns:

```
$PQTMDRCLR,,ERROR,<ErrCode>*<Checksum><CR><LF>
```

For details about <**ErrCode**>, see *Table 2: Error Codes*.

Example:

```
$PQTMDRCLR*53
$PQTMDRCLR,OK*7B
```

4.1.19. PQTMCFGRTK

Configures the RTK mode.

Type:

Set/Get

Synopsis:

```
//Set:
$PQTMCFGRTK,W,<DiffMode>,<RelMode>*<Checksum><CR><LF>
//Get:
$PQTMCFGRTK,R*<Checksum><CR><LF>
```

Parameter:

Field	Format	Unit	Description
<DiffMode>	Numeric	-	Differential mode. 0 = Disable RTK/RTD feature, not using differential data. 1 = Auto mode. 2 = RTD only mode, only using the pseudoranges.
<RelMode>	Numeric	-	The absolute/relative mode. 1 = Absolute mode, ensuring absolute position accuracy. 2 = Relative mode, ensuring relative position accuracy. Note: This field only takes effect when the differential mode is RTK only mode (in auto mode configured by **<DiffMode>**).

Result:

- If successful, the module returns:

```
//Response to Set command:
$PQTMCFGRTK,OK*<Checksum><CR><LF>
//Response to Get command:
$PQTMCFGRTK,OK,<DiffMode>,<RelMode>*<Checksum><CR><LF>
```

- If failed, the module returns:

```
$PQTMCFGRTK,ERROR,<ErrCode>*<Checksum><CR><LF>
```

Example:

```
//Set:
$PQTMCFGRTK,W,1,1*6C
$PQTMCFGRTK,OK*3F
//Get:
$PQTMCFGRTK,R*69
$PQTMCFGRTK,OK,1,1*3F
```

4.1.20. PQTMCFGLA

Configures the lever arm.

Type:

Set/Get

Synopsis:

```
//Set:
$PQTMCFGLA,W,<Type>,<LA_X>,<LA_Y>,<LA_Z>*<Checksum><CR><LF>
//Get:
$PQTMCFGLA,R,<Type>*<Checksum><CR><LF>
```

Parameter:

Field	Format	Unit	Description
<Type>	Numeric	-	The type of the lever arm. 1 = IMU to antenna lever arm
<LA_X>	Numeric	m	The lever arm of X axis.
<LA_Y>	Numeric	m	The lever arm of Y axis.
<LA_Z>	Numeric	m	The lever arm of Z axis.

Result:

- If successful, the module returns:

```
//Response to set command:
$PQTMCFGLA,OK*<Checksum><CR><LF>
//Response to get command:
$PQTMCFGLA,OK,<Type>,<LA_X>,<LA_Y>,<LA_Z>*<Checksum><CR><LF>
```

- If failed, the module returns:

```
$PQTMCFGLA,ERROR,<ErrCode>*<Checksum><CR><LF>
```

Example:

```
//Set:
$PQTMCFGLA,W,1,0.212,0.514,0.113*31
$PQTMCFGLA,OK*7F
//Get:
$PQTMCFGLA,R,1*34
$PQTMCFGLA,OK,1,0.212,0.514,0.113*62
```

4.1.21. PQTMCFGLAM

Configures the mode of lever arm.

Type:

Set/Get

Synopsis:

```
//Set:
$PQTMCFGLAM,W,<Mode>*<Checksum><CR><LF>
//Get:
$PQTMCFGLAM,R*<Checksum><CR><LF>
```

Parameter:

Field	Format	Unit	Description
<Mode>	Numeric	-	The mode of the lever arm. 1 = Automatic estimation mode, the algorithm automatically estimates the lever arm information and never uses the lever arm settings configured by command **$PQTMCFGLA**. 2 = Strict mode, the algorithm uses the lever arm settings configured by command **$PQTMCFGLA**. 3 = Smart mode, the algorithm uses the lever arm settings configured by the command **$PQTMCFGLA** and intelligently corrects the lever arm information if the settings are incorrect.

Result:

- If successful, the module returns:

```
//Response to set command:
$PQTMCFGLAM,OK*<Checksum><CR><LF>
//Response to get command:
$PQTMCFGLAM,OK,<Mode>*<Checksum><CR><LF>
```

- If failed, the module returns:

```
$PQTMCFGLAM,ERROR,<ErrCode>*<Checksum><CR><LF>
```

Example:

```
//Set:
$PQTMCFGLAM,W,2*7F
$PQTMCFGLAM,OK*32
//Get:
$PQTMCFGLAM,R*64
$PQTMCFGLAM,OK,2*2C
```

4.2. PAIR Messages

This chapter explains DR related PAIR messages (proprietary NMEA messages defined by the chipset supplier). "**P**" means proprietary message, "**AIR**" means the command defined by the chipset supplier.

4.2.1. Packet Type: 6010 PAIR_CUSTOM_SET_MSG_OUTPUT

Enables/disables the output of **$PQTMVEHMSG**, **$PQTMSENMSG**, **$PQTMDRCAL**, **$PQTMIMUTYPE** and **$PQTMVEHMOT** messages.

Type:

Set

Synopsis:

```
$PAIR6010,<Type>,<Output_State>*<Checksum><CR><LF>
```

Parameter:

Field	Format	Unit	Description
<Type>	Numeric	-	Message type. -1 = Reset output state of all following sentence types to the default value 0 = $PQTMVEHMSG (Default: disabled) 1 = $PQTMSENMSG (Default: disabled) 2 = $PQTMDRCAL (Default: disabled) 3 = $PQTMIMUTYPE (Default: enabled) 4 = $PQTMVEHMOT (Default: disabled)
<Output_State>	Numeric	-	Message output state. 0 = Disabled 1 = Enabled

Result:

Returns **$PAIR001** message. See *document [2] protocol specification* for details.

Example:

```
$PAIR6010,0,1*0C
$PAIR001,6010,0*0C
```

NOTE

1. Send **$PQTMSAVEPAR*5A** and reset the module for **$PAIR6010** to take effect.
2. The output rate of **$PQTMVEHMSG** and **$PQTMSENMSG** is always 10 Hz. The output rate of **$PQTMDRCAL** and **$PQTMVEHMOT** depends on position fix rate. **$PQTMIMUTYPE** is only output once after each boot-up.
3. It's recommended to use **$PQTMCFGMSGRATE** to configure these messages instead of this command, for more information, please refer to *document [2] protocol specification*.

4.2.2. Packet Type: 6011 PAIR_CUSTOM_GET_MSG_OUTPUT

Gets whether the output of **$PQTMVEHMSG**, **$PQTMSENMSG**, **$PQTMDRCAL**, **$PQTMIMUTYPE** and **$PQTMVEHMOT** messages is enabled.

Type:

Get

Synopsis:

```
$PAIR6011,<Type>*<Checksum><CR><LF>
```

Parameter:

Field	Format	Unit	Description
<Type>	Numeric	-	Message type. 0 = $PQTMVEHMSG 1 = $PQTMSENMSG 2 = $PQTMDRCAL 3 = $PQTMIMUTYPE 4 = $PQTMVEHMOT

Result:

Returns **$PAIR001** message and query result. See *document [2] protocol specification* for details.

Query result message format:

$PAIR6011,<Type>,<Output_State>*<Checksum><CR><LF>

Parameters included in the result:

Field	Format	Unit	Description
<Type>	Numeric	-	Message type. 0 = $PQTMVEHMSG 1 = $PQTMSENMSG 2 = $PQTMDRCAL 3 = $PQTMIMUTYPE 4 = $PQTMVEHMOT
<Output_State>	Numeric	-	Message output state. 0 = Disabled 1 = Enabled

Example:

```
$PAIR6011,1*11
$PAIR001,6011,0*0D
$PAIR6011,1,0*0D
```

NOTE

1. This command is only supported by LC29H (BA) and LC29H (CA) with software versions dedicated for four-wheel vehicles. Contact Quectel Technical Support (support@quectel.com) for details about the software versions.
2. It's recommended to use **$PQTMCFGMSGRATE** to configure these messages instead of this command, for more information, please refer to *document [2] protocol specification*.

5 FAQ

5.1. DR General Problem Solving

- **Q: How to switch between ADR and UDR mode?**
  A: The module can intelligently identify UDR and ADR modes. When the module is powered on, it is in UDR mode by default. Once the vehicle speed data input is detected, it will automatically switch to ADR mode.

- **Q: What are the precautions for the module installation?**
  A: The module is strictly fixed on the rigid structure of the carrier and has no relative motion with the carrier. There is no relative motion between the module and the GNSS antenna, and the relative spatial position is fixed.

- **Q: Does DR support fast calibration?**
  A: Yes. The premise is that the module is fixed on the carrier and has been run several times. The module can save the calibration data in NVM. Please note that not change the installation after the module has been calibrated.

- **Q: Does DR support hot start function?**
  A: Yes. At present, the firmware turns on the DR hot function by default, and the premise is that the INS unit is fixed on the carrier and no installation changes occur in the two tests before and after. If the installation changes, you need to clear the DR positioning information before the next test. If not clear, the firmware will be abnormal before the DR completes the new calibration. **The DR hot start function can only take effect when the vehicle is static started, that is, "static storage, static effect"**. For configuring the DR hot feature, please refer to *Chapter 4.1.15 PQTMCFGDRHOT*.
  And also, the module supports saving DR hot related information by command **$PQTMDRSAVE*0F** and clearing DR hot related information by command **$PQTMDRCLR*53**.

NOTE

LC29H (BA) and LC29H (CA) have different firmware versions for two wheels and four wheels.
For example:

- LC29H (BA) has LC29HBA_CSA2 firmware version for two wheels and LC29HBA_CSA4 for four wheels.
- LC29H (CA) has LC29HCA_DSA2 firmware version for two wheels and LC29HCA_DSA4 for four wheels.

5.2. RTK General Problem Solving

- **Q: How to solve the problem that the module cannot to enter differential mode?**
  A: 1) Observe the environment to see whether there are interference sources such as high-voltage lines, tall buildings and dense forests around, and check the differential status after staying away from the interference source.
  2) Check the differential data is injected to the module correctly.
  3) Check if the RTK service (constellation and signal frequency band) matches the module.

- **Q: How to solve the problem that the differential mode is always in float state?**
  A: 1) Observe the environment to check whether there are houses, tall buildings, dense forests around, and stay away from occlusion.
  2) Check whether the module is connected to a correct RTK services and the GNSS constellation and satellite signal frequency band match the module.
  3) Check whether the differential data is stably injected into the module and check the injection period and data integrity. Any byte errors will cause checksum to be incorrect and the entire message frame to be unusable. This may even cause several subsequent differential data frames to be unusable. User can check the differential age in GGA message.
  4) Check the number of satellites searched and the number of satellites used (GSV and GSA messages).
  5) Check the satellite signal level in GSV message.

6 Example

6.1. EVB Top View

EVB top view is shown in the figure below. For more information, please refer to *document [3] EVB user guide*.

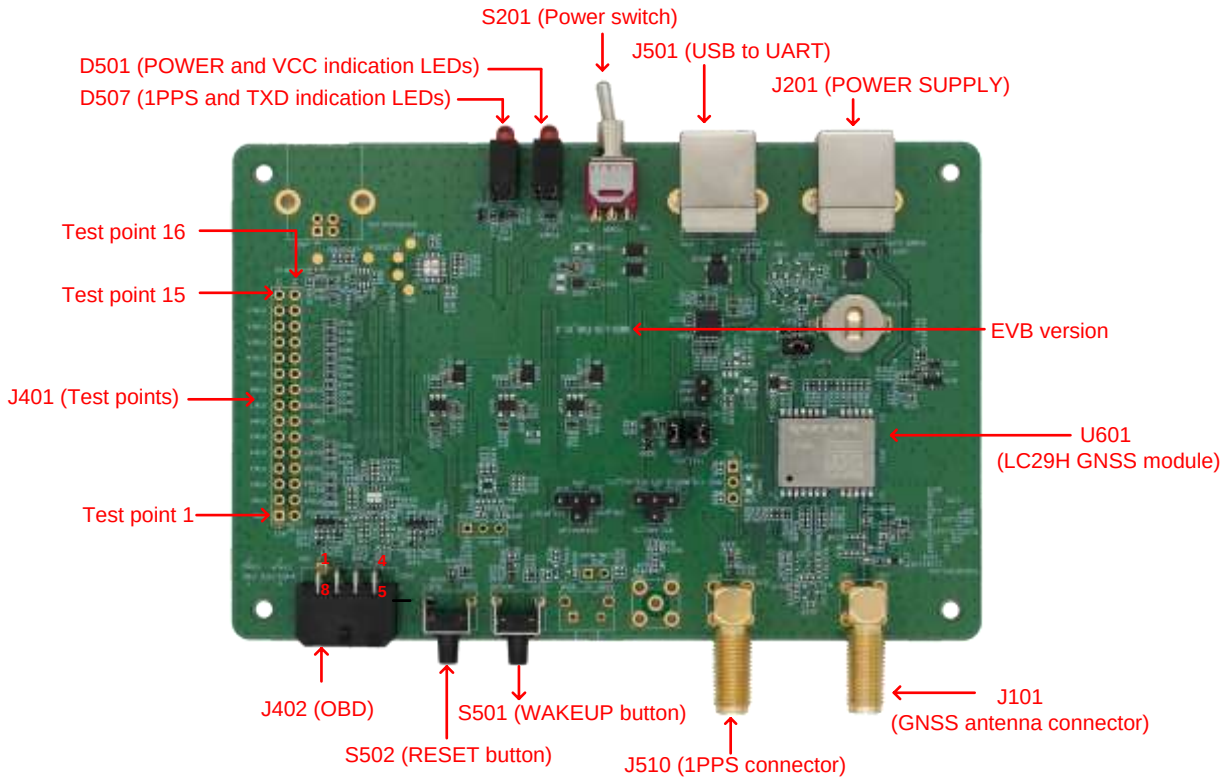


Figure 16: EVB Top View

6.2. Speed Data Injection

The module supports two speed injection methods. One is the module's WHEELTICK pin connect to a vehicle speed sensor which can provide the wheel tick pulse, and the FWD pin connect to the vehicle reverse indicator. The other one is injecting the speed data via communication interface (using PQTM message), such as UART. For each method, users need to ensure that the vehicle speed data is real-time and accurate, otherwise the DR performance is reduced, or even the DR cannot be calibrated or takes a long time for DR calibration.

6.2.1. Connect to Vehicle Speed Sensor

In this way, a speed sensor needs to connect to the module's WHEELTICK pin, the speed sensor generates the speed pulse, and the vehicle reserve indicator connect to the module's FWD pin, the module will auto calculate the vehicle speed and inject to DR engine. For WHEELTICK pin connection, please refer to *document [4] hardware design*.

NOTE

1. The requirement of the speed sensor resolution is ≤ 0.05 m/tick.
2. The vehicle direction defaults to forward if the FWD pin is not connected.

6.2.2. PQTM Speed Data Injection

In this way, HOST can inject the speed data via communication interface with **$PQTMVEHMSG** message. The speed data can be cumulative wheel-tick count or digital speed (in m/s). For details of the message format, refer to *Chapter 4.1.3 PQTMVEHMSG*.

Cumulative wheel-tick injection through the communication interface (See *Chapter 4.1.3.2 If <MsgType> = 2* for the command), with a minimum injection frequency of 10 Hz, and the maximum distance increment is 0.05 m for each wheel-tick pulse.

The digital vehicle speed data can only be injected through the communication interface, with the minimum injection frequency of 10 Hz, and the maximum error of 0.1 m/s between the injected speed and actual speed.

It is crucial to ensure stable vehicle speed injection. Any interruption during driving will affect performance or even cause DR to be unavailable.

6.3. Wheel-Tick Speed Injection Example

To use EVB to inject vehicle speed, you need to connect the external vehicle speed pulse signal to the WHEELTICK Pin and GND Pin in J402 Pin. The voltage range is 1.8–5 V. There is no requirement for pulse width. The rising edge of each pulse will be counted. After the pulse is successfully connected, you will see the WheelTickCNT accumulating when the vehicle is driving. If you are using the two-wheel version, the WheelTickCNT is in the PQTMIMU message, so you need to request the output of the PQTMIMU message. If you are using the four-wheel version, you need to request the PQTMVEHMSG message.

Pin description of J402 is shown below:

Table 3: J402 Pin Description

Pin Number	Pin Label	Pin Function	I/O	Description
1	-	GND	-	Ground
2	CAN_L	-	-	NC
3	CAN_H	-	-	NC
4	-	GND	-	Ground
5	-	-	-	-
6	WHEELTICK	WHEELTICK	DI	Odometer/Wheel-tick pulse input
7	FWD	FWD	DI	Forward/Backward status signal input
8	-	-	-	-

6.4. Installation and Calibration

After the module is installed, the module installation please refer to *Chapter 2.5 Module* Mounting. After completing the calibration, you can test the DR performance in a weak signal scenario. Calibration process please refer to *Chapter 2.6 DR Calibration.*

7 Appendix A References

Table 4: Related Documents

Document Name
[1] Quectel_GNSS_RTK_Application
[2] Quectel_LC29H_Series&LC79H(AL)_GNSS_Protocol_Specification
[3] Quectel_LC29H_Series_EVB_User_Guide
[4] Quectel_LC29H_Series_Hardware_Design

Table 5: Terms and Abbreviations

Abbreviation	Description
ADAS	Advanced Driver Assistance Systems
ARP	Antenna Reference Point
BDS	BeiDou Navigation Satellite System
CORS	Continuously Operating Reference Stations
DR	Dead Reckoning
Galileo	Galileo Satellite Navigation System (EU)
GLONASS	Global Navigation Satellite System (Russian)
C/N_0	Carrier-to-Noise-Density Ratio
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
MSM	Multiple Signal Message

Abbreviation	Description
NMEA	NMEA (National Marine Electronics Association) Interface Standard
PQTM	Proprietary Protocol of Quectel
PVT	Position Velocity and Time
QZSS	Quasi-Zenith Satellite System
RTC	Real-Time Clock
RTCM	Radio Technical Commission for Maritime Services
RTK	Real-Time Kinematic

8 Appendix B Special Characters

Table 6: Special Characters

Special Character	Definition
<CR>	Carriage return character.
<LF>	Line feed character.
<...>	Parameter name. Angle brackets do not appear in the message.
[...]	Optional field of a message. Square brackets do not appear in the message.
{…}	Repeated field of a message. Curly brackets do not appear in the message.
Underline	Default setting of a parameter.