Q1 Series

产品手册

Revision 0.50

2019/01

COPYRIGHT

GENERAL NOTES

FOHEART OFFERS THE INFORMATION AS A SERVICE TO ITS CUSTOMERS. THE INFORMATION PROVIDED IS BASED UPON CUSTOMERS' REQUIREMENTS. FOHEART MAKES EVERY EFFORT TO ENSURE THE QUALITY OF THE INFORMATION IT MAKES AVAILABLE. FOHEART DOES NOT MAKE ANY WARRANTY AS TO THE INFORMATION CONTAINED HEREIN, AND DOES NOT ACCEPT ANY LIABILITY FOR ANY INJURY, LOSS OR DAMAGE OF ANY KIND INCURRED BY USE OF OR RELIANCE UPON THE INFORMATION. ALL INFORMATION SUPPLIED HEREIN IS SUBJECT TO CHANGE WITHOUT PRIOR NOTICE.

For technical support, please visit:

http://www.foheart.com/support.html or
http://www.foheart.com/question.html

北京总部
Tel：(+86)010-56106165
Email：contact@foheart.com

地址：北京市海淀区黑山扈路红山口 8 号 D2-南-3 号
邮编：100091

Revision History

Revision No.	Date	Description	Author(s)
0.10	Feb. 15, 2017	First Draft	HMX
0.20	Oct. 27, 2017	修改了 2.4.1 配置文件的存放顺序。 修改了 2.4.2 算法配置区为 512 字节。	HMX
0.40	Jan. 30, 2018	添加加速度计、磁力计校准等新指令。	HMX
0.50	Jan. 2019	添加了安装尺寸、订货信息、更改了部分指令格式	HMX

名词解释：

四元数：模值为 1，发送顺序按照 WXYZ 发送，坐标系定义见附录。

欧拉角：单位为 degree，发送顺序按照 XYZ 发送，坐标系定义见附录。

加速度值：单位为 g，发送顺序按照 XYZ 发送，坐标系定义见附录。

角速度值：单位为 degree/s，发送顺序按照 XYZ 发送，坐标系定义见附录。

磁力计值：单位为 mGauss，发送顺序按照 XYZ 发送，坐标系定义见附录。

一、基本命令

1.1 查询模块固件信息

命令字/Code	指令	描述
0x01（Write）	REQFW	查询节点固件信息
0x02（Read）		

1.1.1 发送端数据帧格式

数据帧格式
<0x01>+<1Byte:Code>+<CRC16>

C/C++结构体参考

```
typedef struct
{
    uint8_t     addr;
    uint8_t     code;
    uint16_t    crc16;
}ReqFw_UART_PC2ND;
```

1.1.2 接收端返回数据帧格式

数据帧格式
<地址 0x01>+<Code：0x02>+[4x3Bytes：CPUID]+[4Bytes：Serial Number]+[3Bytes：固件版本号]+[2Bytes：CPU 频率，单位 MHz]+[6Bytes：保留信息]+<CRC16>

C/C++结构体参考

```
typedef struct
{
    uint8_t     addr;
    uint8_t     code;
    uint32_t    cpuid[3];
    uint32_t    sn;
    uint8_t     fwversion[3];
    uint16_t    cpuClkMHz;
    uint8_t     resv[6];
    uint16_t    crc16;
}ReqFw_UART_ND2PC;
```

1.2 查询/设置加速度计参数

命令字	指令	描述
0x03（Write）	ACCELINFO	查询加速度计参数
0x04（Read）		

1.2.1 发送端数据帧格式

数据帧格式
<地址 0x01>+<0x03>+<CRC16>

C/C++结构体参考

```
typedef struct {
    uint8_t addr;
    uint8_t code;
    /*0x01:请求读取零偏；
       0x02：请求写入零偏；
       0x03：请求写入零偏，并保存到外置存储器，永久使用*/
    uint8_t cmd;
    /*倍率 1<<16*/
    int32_t bias[3];
    /*倍率 1<<24*/
    int32_t     scall[3][3];
    uint16_t    crc16;
}AccelInfo_UART_PC2ND;
```

1.2.2 接收端返回数据帧格式

数据帧格式
<地址 0x01>+<0x04>+<1Byte：status>+<3x4Bytes：零偏>+<3x3x4Bytes：比例因子>+<CRC16>

C/C++结构体参考

```
typedef struct
{
    uint8_t addr;
    uint8_t code;
```

```
    /*
       1、若上位机发送 0x01：
        （1）返回 0x01 代表读取零偏成功。
        （2）返回 0xFF 代表读取零偏失败。
       2、若上位机发送 0x02：
        （1）返回 0x01 写入零偏成功。
        （2）返回 0xFF 代表写入零偏失败。
       3、若上位机发送 0x03：
        （1）返回 0x01 写入零偏，并保存到外置存储器成功。
        （2）返回 0xFF 写入零偏失败。
     */
    uint8_t status;
    /*倍率 1<<16*/
    int32_t bias[3];
    /*倍率 1<<24*/
    int32_t     scall[3][3];
    uint16_t    crc16;
}AccelInfo_UART_ND2PC;
```

1.3 查询/设置陀螺仪计参数

命令字	指令	描述
0x05（Write）	GYROINFO	查询陀螺仪参数
0x06（Read）		

1.3.1 发送端数据帧格式

数据帧格式
<地址 0x01>+<0x05>+<CRC16>
C/C++结构体参考

```
typedef struct {
    uint8_t addr;
    uint8_t code;
    /*0x01:请求读取零偏；
       0x02：请求写入零偏；
       0x03：请求写入零偏，并保存到外置存储器，永久使用*/
    uint8_t cmd;
```

```
    /*倍率 1<<16*/
    int32_t bias[3];
    /*倍率 1<<24*/
    int32_t     scall[3][3];
    uint16_t    crc16;
}GyroInfo_UART_PC2ND;
```

1.3.2 接收端返回数据帧格式

数据帧格式
<地址 0x01>+<0x06>+<1Byte：status>+<3x4Bytes：零偏>+<3x3x4Bytes：比例因子>+<CRC16>
C/C++结构体参考

```
typedef struct
{
    uint8_t addr;
    uint8_t code;
    /*
       1、若上位机发送 0x01：
        （1）返回 0x01 代表读取零偏成功。
        （2）返回 0xFF 代表读取零偏失败。
       2、若上位机发送 0x02：
        （1）返回 0x01 写入零偏成功。
        （2）返回 0xFF 代表写入零偏失败。
       3、若上位机发送 0x03：
        （1）返回 0x01 写入零偏，并保存到外置存储器成功。
        （2）返回 0xFF 写入零偏失败。
     */
    uint8_t status;
    /*倍率 1<<16*/
    int32_t bias[3];
    /*倍率 1<<24*/
    int32_t     scall[3][3];
    uint16_t    crc16;
}GyroInfo_UART_ND2PC;
```

1.4 查询/设置磁力计参数

命令字	指令	描述

0x07（Write）	MAGINFO	查询磁力计参数
0x08（Read）		

1.4.1 发送端数据帧格式

数据帧格式
<地址 0x01>+<0x07>+<CRC16>

C/C++结构体参考

```
typedef struct {
    uint8_t addr;
    uint8_t code;
    /*0x01:请求读取零偏；
       0x02：请求写入零偏；
       0x03：请求写入零偏，并保存到外置存储器，永久使用*/
    uint8_t cmd;
    /*倍率 1<<16*/
    int32_t bias[3];
    /*倍率 1<<24*/
    int32_t     scall[3][3];
    uint16_t    crc16;
}MagInfo_UART_PC2ND;
```

1.4.2 接收端返回数据帧格式

数据帧格式
<地址 0x01>+<0x08>+<1Byte：status>+<3x4Bytes：零偏>+<3x3x4Bytes：比例因子>+<CRC16>

C/C++结构体参考

```
typedef struct
{
    uint8_t addr;
    uint8_t code;
    /*
       1、若上位机发送 0x01：
        （1）返回 0x01 代表读取零偏成功。
        （2）返回 0xFF 代表读取零偏失败。
       2、若上位机发送 0x02：
        （1）返回 0x01 写入零偏成功。
        （2）返回 0xFF 代表写入零偏失败。
```

```
        3、若上位机发送 0x03：
        （1）返回 0x01 写入零偏，并保存到外置存储器成功。
        （2）返回 0xFF 写入零偏失败。
     */
    uint8_t status;
    /*倍率 1<<16*/
    int32_t bias[3];
    /*倍率 1<<24*/
    int32_t     scall[3][3];
    uint16_t    crc16;
}MagInfo_UART_ND2PC;
```

1.5 重新启动/复位

此条命令控制模块重启，模块收到复位指令后，首先返回数据包，然后自动重启，重启后模块返回上一次开机时的运行状态。

命令字	指令	描述
0x09（Write）	REBOOT	模块重启
0x0A（Read）		

1.5.1 发送端数据帧格式

数据帧格式
<地址 0x01>+<0x09>+<CRC16>
C/C++结构体参考

```
typedef struct
{
    uint8_t     addr;
    uint8_t     code;
    uint16_t    crc16;
}Reboot_UART_PC2ND;
```

1.5.2 接收端返回数据帧格式

数据帧格式
<地址 0x01>+<0x0A>+<CRC16>
C/C++结构体参考

```
typedef struct
```

```
{
    uint8_t     addr;
    uint8_t     code;
    uint16_t        crc16;
}Reboot_UART_ND2PC;
```

1.6 置零当前旋转

无论当前旋转值为多少，以当前值为 0 旋转，传输求差值之后的旋转量，即在当前旋转值的基础上旋转了多少。

命令字	指令	描述
0x0B（Write）	SETZERO	置零当前旋转
0x0C（Read）		

1.6.1 发送端数据帧格式

数据帧格式

<地址 0x01>+<0x0B>+<1Byte:Command>+<CRC16>

Command	含义
0x01	开启置零当前旋转
0x02	退出置零当前旋转（开机默认为此状态）

C/C++结构体参考

```
typedef struct
{
    uint8_t addr;
    uint8_t code;
    /*cmd :
       0x01:set current quat to 1
       0x02:reset current quat
     */
    uint8_t     cmd;
    uint16_t    crc16;
}SetZero_UART_PC2ND;
```

1.6.2 接收端返回数据帧格式

数据帧格式

<地址 0x01>+<0x0C>+<1Byte：Status>+<CRC16>

Status	说明
0x01	执行成功
0xFF	执行失败

C/C++结构体参考

```
typedef struct
{
    uint8_t addr;
    uint8_t code;
    /*status:
       0x01:execute success.
     */
    uint8_t     status;
    uint16_t    crc16;
}SetZero_UART_ND2PC;
```

1.7 发送原始数据

请求是否发送原始数据，

设置加速度计、陀螺仪、磁力计是否输出未经校准的原始值。

6Dof 及 9Dof 融合产生姿态四元数，使用的是经过校准的原始数据，不受此条指令的影响。

什么是原始值：芯片的原始输出，未经过校准的数据，一般偏移较大。

命令字	指令	描述
0x0D（Write）	RAWDATA	置零当前旋转
0x0E（Read）		

1.7.1 发送端数据帧格式

数据帧格式

<地址 0x01>+<1Byte：Code>+<1Byte：Command>+<CRC16>

Command	含义
0x01	设置为发送未经校准的原始数据
0x02	设置为发送经过校准的数据（开机默认为此状态）

C/C++结构体参考

```
typedef struct
{
    uint8_t     addr;
    uint8_t     code;
    uint8_t     cmd;
    uint16_t    crc16;
}RawData_UART_PC2ND;
```

用例

1.7.2 接收端返回数据帧格式

数据帧格式

<地址 0x01>+<0x0E>+<1Byte：Status>+<CRC16>

Status	说明
0x01	执行成功
0xFF	执行失败

C/C++结构体参考

```
typedef struct
{
    uint8_t     addr;
    uint8_t     code;
    uint8_t     status;
    uint16_t        crc16;
}RawData_UART_ND2PC;
```

1.8 恢复出厂设置

恢复模组的出厂设置，默认出厂设置如下：

波特率：115200bps

发送数据频率：200Hz

发送数据类型：四元数+欧拉角+标定后的加速度值+标定后的角速度值+标定后的磁力计值。

命令字	指令	描述
0x0F（Write）	FACTORYRESTORE	恢复模块的出厂设置
0x10（Read）		

1.8.1 发送端数据帧格式

数据帧格式

<地址 0x01>+<1Byte：Code>+<1Byte：Command>+<CRC16>

Command	含义
0x01	设置恢复出厂设置，Q1 硬件回复命令后，自动重启后生效。
0x02	设置恢复出厂设置，Q1 硬件回复命令后，暂不重启，在手动重启或断电重启后生效。

C/C++结构体参考

```
typedef struct
{
    uint8_t addr;
    uint8_t code;
    uint8_t     cmd;
    uint16_t    crc16;
}FactoryRestore_UART_PC2ND;
```

1.8.2 接收端返回数据帧格式

数据帧格式

<地址 0x01>+<0x0E>+<1Byte：Status>+<CRC16>

Status	说明
0x01	执行成功
0xFF	执行失败

C/C++结构体参考

```
typedef struct
{
    uint8_t addr;
    uint8_t code;

    /*status:
```

```
7.          0x01:命令成功执行。
8.          0xFF：未知错误，
9.        */
10.     uint8_t      status;
11.     uint16_t     crc16;
12. }FactoryRestore_UART_ND2PC;
```

二、传输命令

2.1 设置传输参数

要求节点以指定频率实时上传姿态等数据，节点确认传输条件是否具备。

若设置参数均正确，模组回复指令的时间大约需要 80ms。

若有错误参数，模组在 5ms 内返回错误信息。

命令字	指令	描述
0x21（Write）	RTTRANS	设置数据传输参数
0x22（Read）		

2.1.1 发送端数据帧格式

数据帧格式

<地址 0x01>+<0x21>+<2Bytes：dataSendFlag 数据发送标志>+<2Bytes：freq 发送频率>+<4Bytes：baudRate 波特率>+<CRC16>

2Bytes：dataSendFlag

用位表示是否包含数据项，全部发送则设置为 0x001F。

Bit15-Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Reserved	1：上传磁力值 0：不上传磁力值	1：上传角速度 0：不上传角速度	1：上传加速度 0：不上传加速度	1：上传欧拉角 0：不上传欧拉角	1：上传四元数 0：不上传四元数

2Bytes：freq（单位 Hz）

上传数据频率，实时生效，手动重启或者断电重启后保持新设置值。

0 代表停止发送。任何不在上表中的值均无法设置。

可以设置的值（单位 Hz）如下：

设置值（十进制表示）	设置值（十六进制表示）

200（默认）	C8
100	64
50	32
25	19
20	14
10	0A
5	05
4	04
2	02
1	01
0	00

4Bytes：baudRate 波特率（单位 bps）

波特率，手动重启或者断电重启后生效。

关于波特率的说明：波特率的最小值为 115200bps。

取值如下（单位 bps）：

设置值（十进制表示）	设置值（十六进制表示）
115200	0x01C200
128000	0x01F400
230400	0x038400
460800	0x070800
921600（默认）	0x0E1000
1500000	0x16E360
2000000	0x1E8480
3000000	0x2DC6C0

C/C++结构体参考

```
1. typedef struct
2. {
```

```
    uint8_t addr;
    uint8_t code;

    /*用位表示是否包含数据项，全部发送则设置为 0x001F*/
    union {
        struct {
            uint16_t    quatFlag : 1;   /*Bit:0*/
            uint16_t    eulerFlag : 1;  /*Bit:1*/
            uint16_t    accelFlag : 1;  /*Bit:2*/
            uint16_t    gyroFlag : 1;   /*Bit:3*/
            uint16_t    magFlag : 1;    /*Bit:4*/
            uint16_t    resvFlag : 11;  /*Bit:5~11*/
        };
        uint16_t dataSendFlag;
    };

    /*
    上传数据频率，实时生效，手动重启或者断电重启后保持新设置值。
    可以设置的值（单位 Hz）如下：
    ----------------------------------------
    DEC|200|100|50|25|20|10|5 |4 |2 |1 |0 |
    HEX|C8 |64 |32|19|14|0A|05|04|02|01|00|
    ----------------------------------------
    0 代表停止发送。任何不在上表中的值均无法设置。
     */
    uint16_t freq;
    /*
    波特率，手动重启或者断电重启后生效。
    取值如下（单位 bps）：
    ------------------------------------------------------------------
    DEC|115200|128000|230400|460800|921600|1500000|2000000|3000000|
    HEX|01C200|01F400|038400|070800|0E1000|16E360 |1E8480 |2DC6C0 |
    ------------------------------------------------------------------
    */
    uint32_t    baudRate;
    uint16_t    crc16;
}RtTrans_UART_PC2ND;
```

用例

发送/回复指令	说明
发送指令：01 21 1F 00 64 00 00 10 0E	上传四元数+欧拉角+加速度+角速度+

00 FF FF 回复指令：01 22 01 F9 60	磁力值； 100Hz 上传； 921600bps 波特率；
发送指令：01 21 1F 00 01 00 00 10 0E 00 FF FF 回复指令：01 22 01 F9 60	上传四元数+欧拉角+加速度+角速度+ 磁力值； 1Hz 上传； 921600bps 波特率；
发送指令：01 21 1F 00 00 00 00 10 0E 00 FF FF 回复指令：01 22 01 F9 60	上传四元数+欧拉角+加速度+角速度+ 磁力值； 0Hz 上传（停止上传）； 921600bps 波特率；
发送指令：01 21 01 00 C8 00 00 10 0E 00 FF FF 回复指令：01 22 01 F9 60	上传四元数； 200Hz 上传； 921600bps 波特率；
01 21 1F 00 C8 00 00 C2 01 00 FF FF	上传四元数+欧拉角+加速度+角速度+ 磁力值； 200Hz 上传； 115200bps 波特率；

2.1.2 接收端返回数据帧格式

数据帧格式

<地址 0x01>+<0x22>+<1Byte：Status>+<CRC16>

Status	说明
0x01	执行成功
0xFF	执行失败

C/C++结构体参考

```
1.  typedef struct
2.  {
```

```
    uint8_t addr;
    uint8_t code;
    /*status:
       0x01：设置成功.
       0xFF：设置中有一个或多个参数不正确。.
     */
    uint8_t     status;
    uint16_t    crc16;
}RtTrans_UART_ND2PC;
```

2.2 开始实时传输

节点收到 2.1 节 RTTRANS 指令后，按照要求的频率上传姿态及其他数据，其中波特率的设置需要重启生效，其它的设置为实时生效。

命令字	指令	描述
0xXX（No Write）	UDIMU	上传 IMU 数据
0x24（Read）		

2.2.1 发送端数据帧格式

无需向模组发送指令，模组根据 2.1 节的设置自动上传数据帧。

2.2.2 接收端返回数据帧格式

数据帧格式
<地址 0x01>+<0x22>+<2Bytes：dataSendFlag 数据发送标志>+[8Bytes：四元数]+[6Bytes：欧拉角]+[6Bytes：加速度]+ [6Bytes：角速度]+ [6Bytes：磁力值]+<CRC16>

2Bytes：dataSendFlag 根据 dataSendFlag 的不同可至多发送 16 种不同数据。					
Bit15-Bit5	**Bit4**	**Bit3**	**Bit2**	**Bit1**	**Bit0**
Reserved	1：包含磁力值 0：不包含磁力值	1：包含角速度 0：包含传角速度	1：包含加速度 0：不包含加速度	1：包含欧拉角 0：不包含欧拉角	1：包含四元数 0：不包含四元数

8Bytes：四元数							
Byte7	Byte6	Byte5	Byte4	Byte3	Byte2	Byte1	Byte0
z 高字节	z 低字节	y 高字节	y 低字节	x 高字节	x 低字节	w 高字节	w 低字节
两字节合为一个 int16_t 类型数据		两字节合为一个 int16_t 类型数据		两字节合为一个 int16_t 类型数据		两字节合为一个 int16_t 类型数据	
将上述所得数据除以（1<<14）获得浮点数值表示的 z		将上述所得数据除以（1<<14）获得浮点数值表示的 y		将上述所得数据除以（1<<14）获得浮点数值表示的 x		将上述所得数据除以（1<<14）获得浮点数值表示的 w	
取值范围为[-1.0,+1.0]		取值范围为[-1.0,+1.0]		取值范围为[-1.0,+1.0]		取值范围为[-1.0,+1.0]	

6Bytes：欧拉角					
Byte5	Byte4	Byte3	Byte2	Byte1	Byte0
z 高字节	z 低字节	y 高字节	y 低字节	x 高字节	x 低字节
两字节合为一个 int16_t 类型数据		两字节合为一个 int16_t 类型数据		两字节合为一个 int16_t 类型数据	
将上述所得数据除以（1<<7）获得浮点数值表示的 z		将上述所得数据除以（1<<7)获得浮点数值表示的 y		将上述所得数据除以（1<<7)获得浮点数值表示的 x	
取值范围为[0,360.0]		取值范围为[-180.0,+180.0]		取值范围为[-180.0,+180.0]	

6Bytes：加速度					
Byte5	Byte4	Byte3	Byte2	Byte1	Byte0
z 高字节	z 低字节	y 高字节	y 低字节	x 高字节	x 低字节

两字节合为一个 int16_t 类型数据	两字节合为一个 int16_t 类型数据	两字节合为一个 int16_t 类型数据
将上述所得数据除以 2048.0 获得浮点数值表示的 z	将上述所得数据除以 2048.0 获得浮点数值表示的 y	将上述所得数据除以 2048.0 获得浮点数值表示的 x
取值范围为[-16.0,+16.0]	取值范围为[-16.0,+16.0]	取值范围为[-16.0,+16.0]

6Bytes：角速度					
Byte5	Byte4	Byte3	Byte2	Byte1	Byte0
z 高字节	z 低字节	y 高字节	y 低字节	x 高字节	x 低字节
两字节合为一个 int16_t 类型数据		两字节合为一个 int16_t 类型数据		两字节合为一个 int16_t 类型数据	
将上述所得数据除以 16.4 获得浮点数值表示的 z		将上述所得数据除以 16.4 获得浮点数值表示的 y		将上述所得数据除以 16.4 获得浮点数值表示的 x	
取值范围为[-2000.0,+2000.0]		取值范围为[-2000.0,+2000.0]		取值范围为[-2000.0,+2000.0]	

6Bytes：磁力值					
Byte5	Byte4	Byte3	Byte2	Byte1	Byte0
z 高字节	z 低字节	y 高字节	y 低字节	x 高字节	x 低字节
两字节合为一个 int16_t 类型数据		两字节合为一个 int16_t 类型数据		两字节合为一个 int16_t 类型数据	
将上述所得数据除以 1.0 获得浮点数值表示的 z		将上述所得数据除以 1.0 获得浮点数值表示的 y		将上述所得数据除以 1.0 获得浮点数值表示的 x	

C/C++结构体参考

```
1.  typedef struct
2.  {
3.      uint8_t addr;
```

```
    uint8_t code;
    /*[0, 4294967295]之间增加，达到最大值后从 0 开始重新计数，
       注意若以 200Hz 上传数据，此值约 248 天溢出一次*/
    uint32_t frameNum;

    /*用位表示是否包含数据项*/
    union {
        struct {
            uint16_t    quatFlag : 1;   /*Bit:0*/
            uint16_t    accelFlag : 1;  /*Bit:1*/
            uint16_t    gyroFlag : 1;   /*Bit:2*/
            uint16_t    magFlag : 1;    /*Bit:3*/
            uint16_t    eulerFlag : 1;  /*Bit:4*/
            uint16_t    resvFlag : 11;  /*Bit:5~11*/
        };
        uint8_t dataSendFlag;
    };

    int16_t     quat[4];
    int16_t     euler[3];
    int16_t     accel[3];
    int16_t     gyro[3];
    int16_t     mag[3];
    uint16_t    crc16;
}UDIMU_UART_ND2PC;
```

三、硬件特性

3.1 陀螺仪参数

Full-Scale Range	±2000º/s
Cross-Axis Sensitivity	±2 %
Nonlinearity	±0.1%
Noise Power Spectral Density	0.01º/s/√Hz
Low pass filter	250Hz

3.2 加速度计参数

Full-Scale Range	±16g
Cross-Axis Sensitivity	±2 %
Nonlinearity	±0.5%
Noise Power Spectral Density	300μg/√Hz
Low pass filter	460Hz

3.3 磁力计参数

Full-Scale Range	±2Gauss
Cross-Axis Sensitivity	0.1%/G(Cross field = 1 Gauss, Happlied = ±2 Gauss)
Linearity (Best fit linear curve)	0.1%FS(Field Range = ±2G)
X-Y-Z Orthogonality	90±1degree

3.4 姿态解算精度

Pitch/Roll	0.5 º
Yaw	1 º

3.5 电气特性

电流	25mA@5V
功耗	125mW
供电电源	5V/1A DC

四、硬件接口

4.1 硬件接口图

图 4.1 数据接口 Front View/前视图

表 4.1 引脚功能

Pin Number	Function
1	USB DP
2	5V DC
3	GND
4	TTL TX
5	TTL RX
6	USB DM

4.2 USB 连接方式

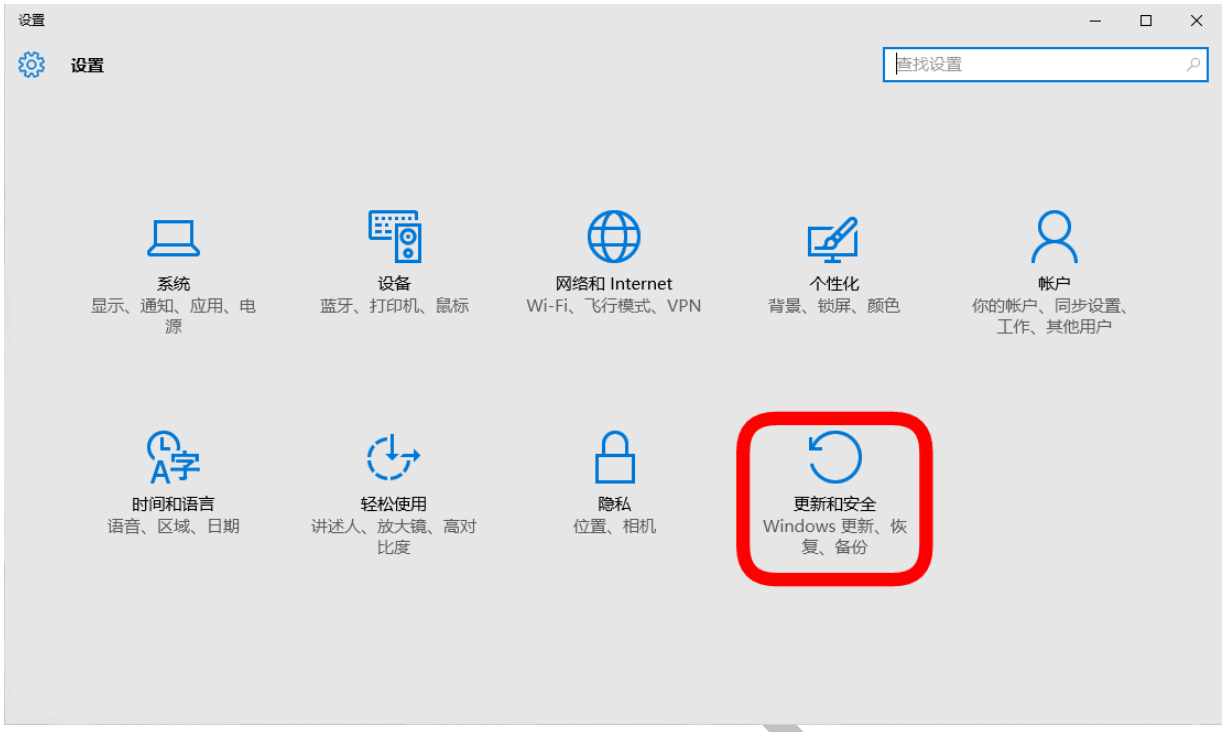
使用 USB 连接方式评估模组的性能。

使用 USB 航空插头线，连接模组与 PC 机，并按照视频教程安装所需硬件驱动。

4.2.1 Win10 系统驱动安装

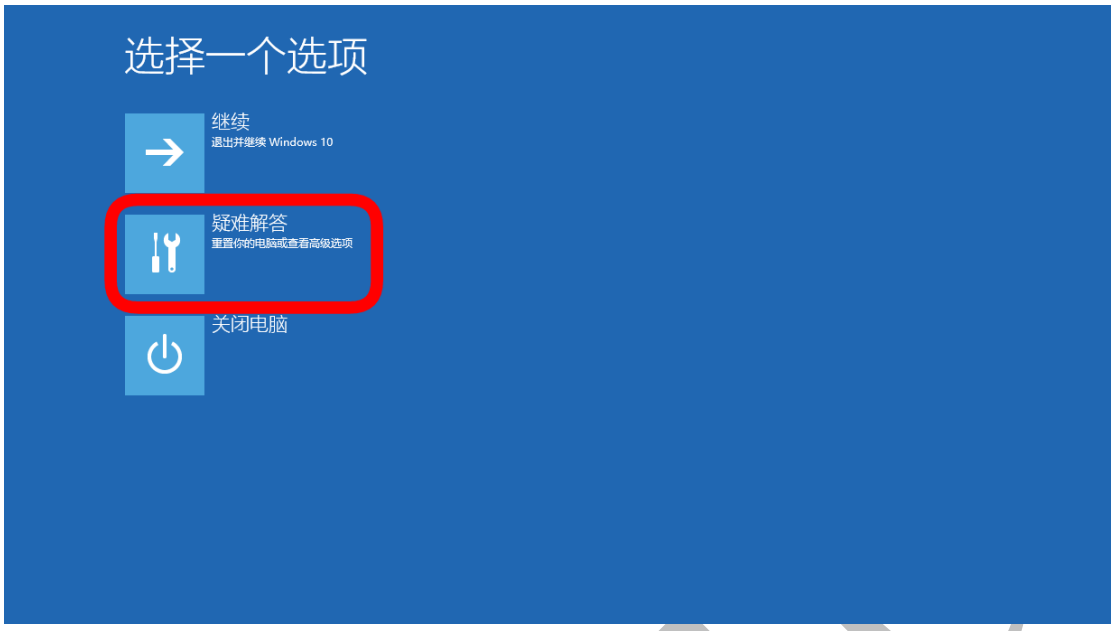
（1）打开所有设置

选择更新和安全。



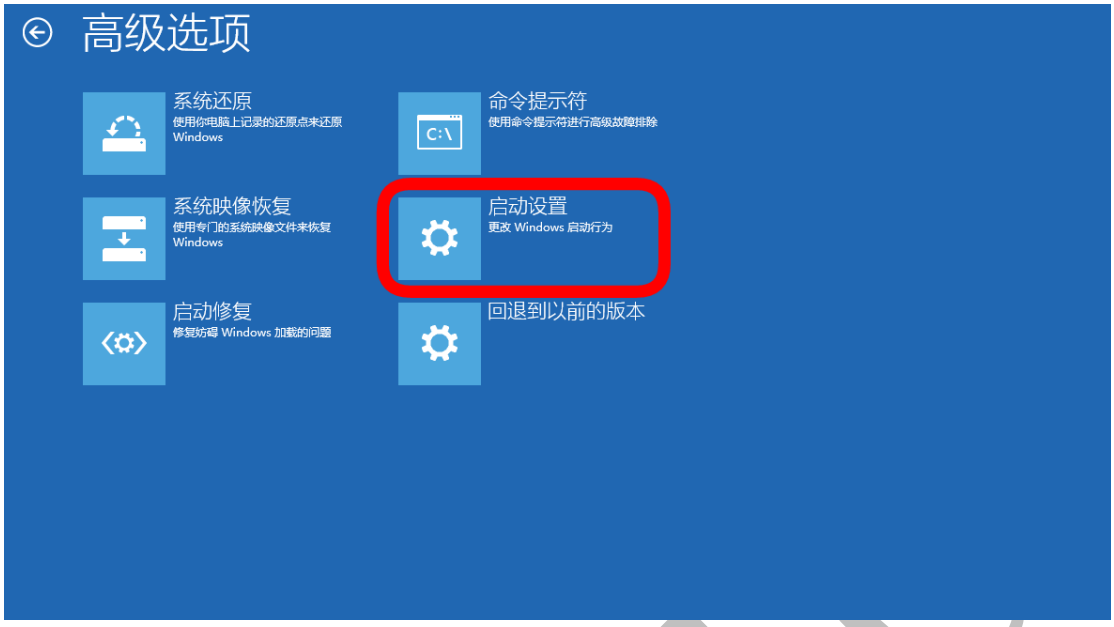
（2）选择高级启动

（3）选择疑难解答

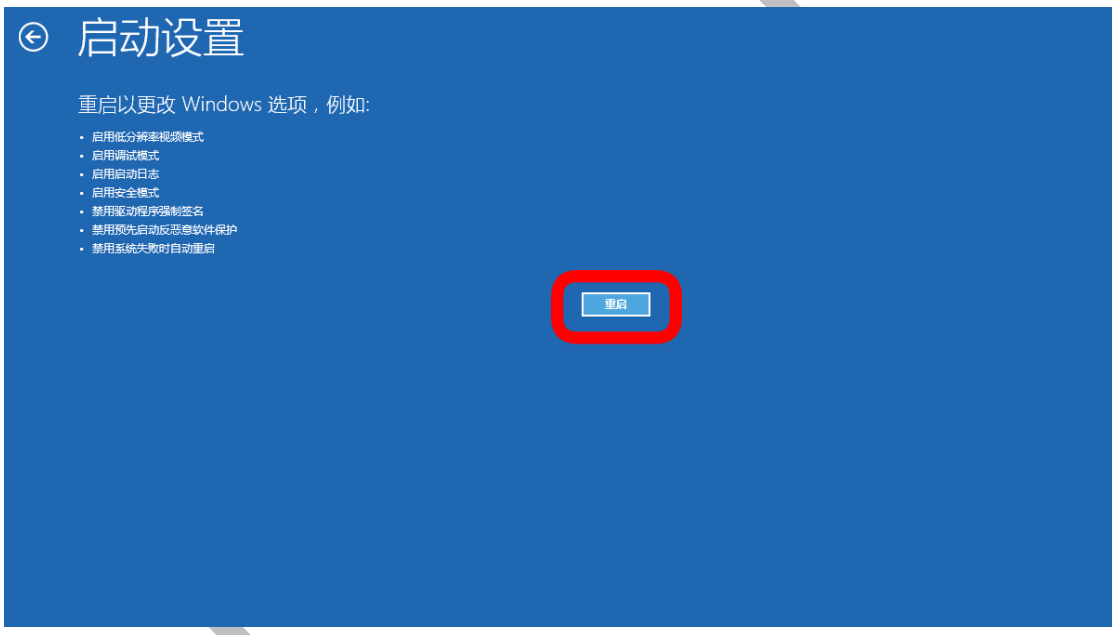


（4）选择高级选项

（5）选择启动设置

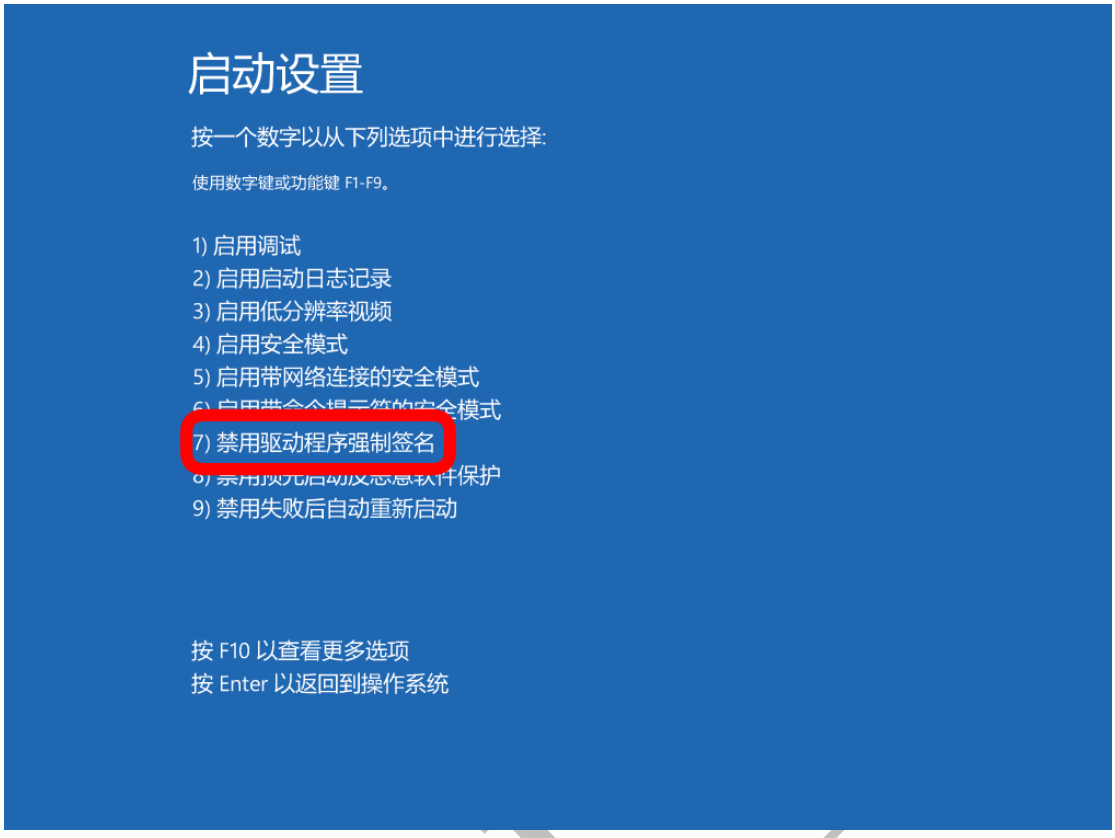


（6）选择重启



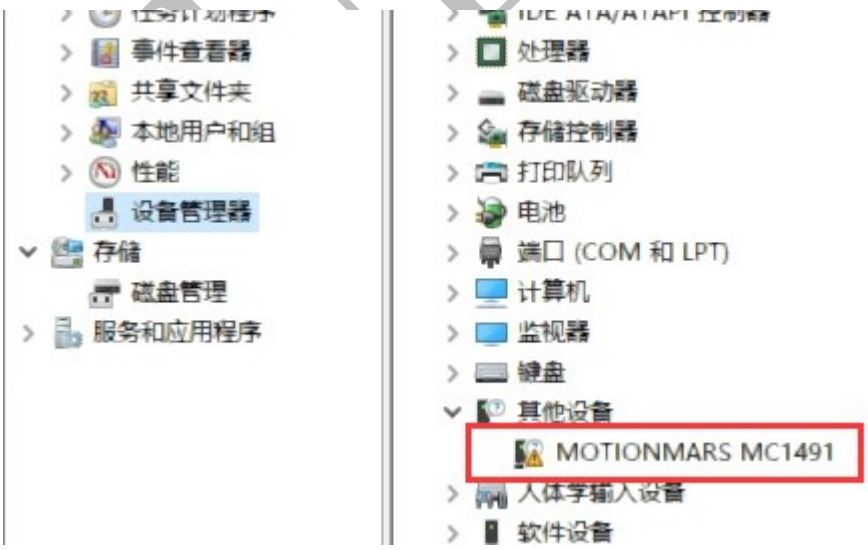
重启之后，进入启动设置界面。

（7）按键盘 7 选择禁用驱动程序强制签名



（8）连接模组

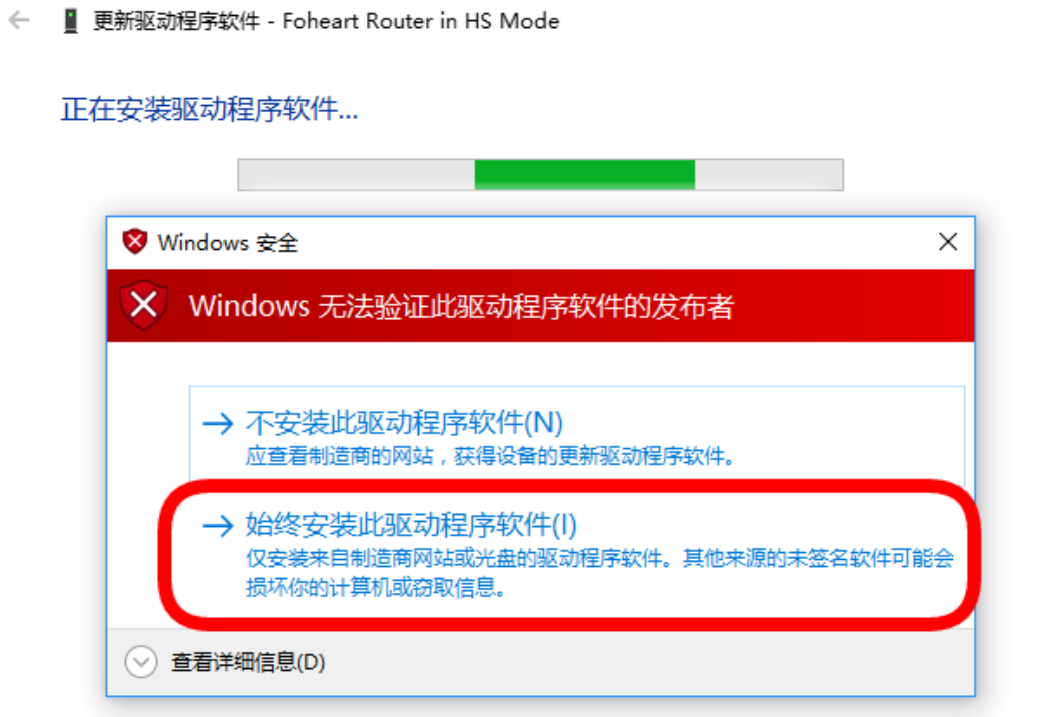
连接 MOTIONMARS Q1 设备到电脑，打开计算机管理，出现 MOTIONMARS MC1491 设备。



右键更新驱动程序软件，选择浏览计算机以查找驱动程序软件。

选择 MotionMars 软件安装目录中的 **MOTIONMARS MC1491 Driver** 文件夹。

单击下一步，选择始终安装驱动



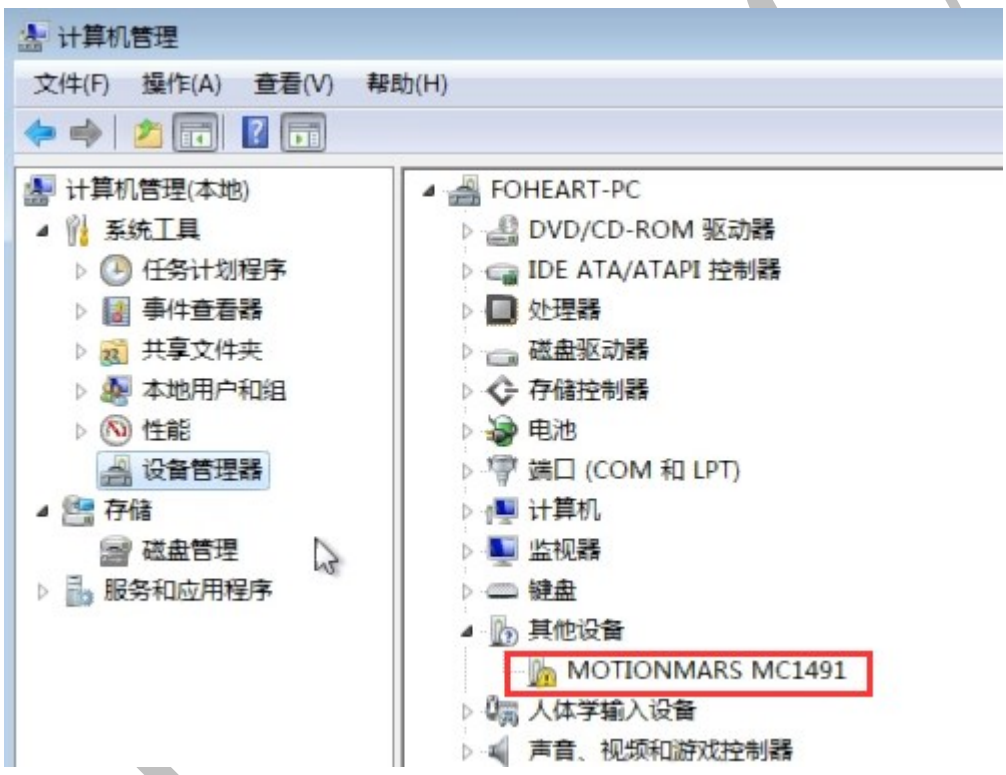
（9）断电重启模组，驱动安装完成

在设备管理器中出现 MOTIONMARS MC1491 设备，驱动成功安装。

4.2.2 Win7 系统驱动安装

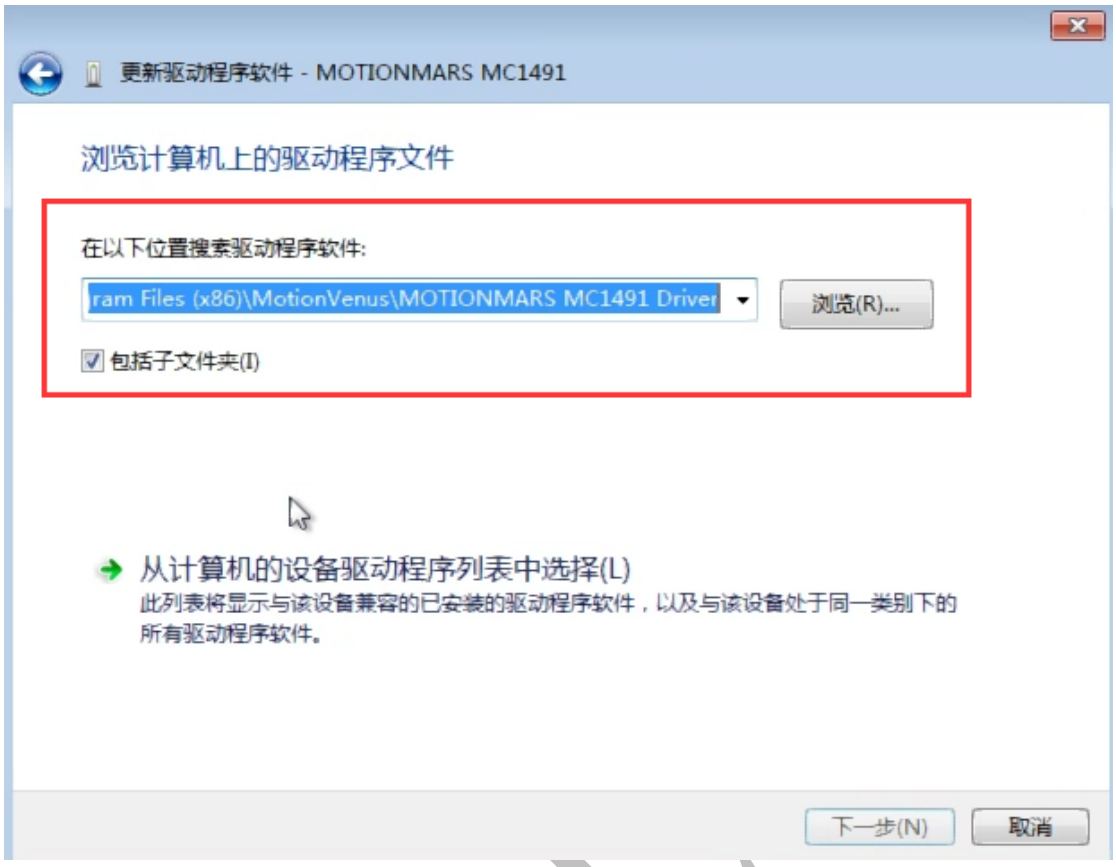
（1）连接模组

连接 MOTIONMARS Q1 设备到电脑，打开计算机管理，出现 MOTIONMARS MC1491 设备。



右键更新驱动程序软件，选择浏览计算机以查找驱动程序软件。

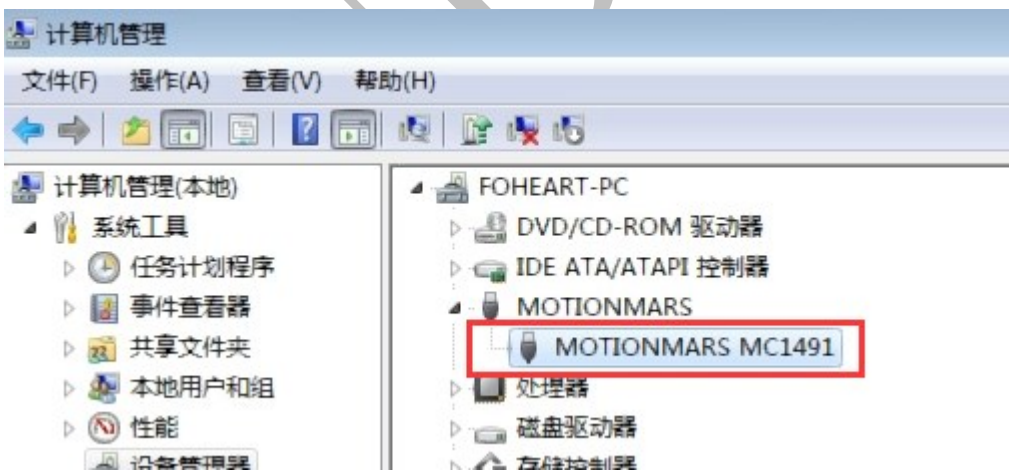
选择 MotionMars 软件安装目录中的 **MOTIONMARS MC1491 Driver** 文件夹。



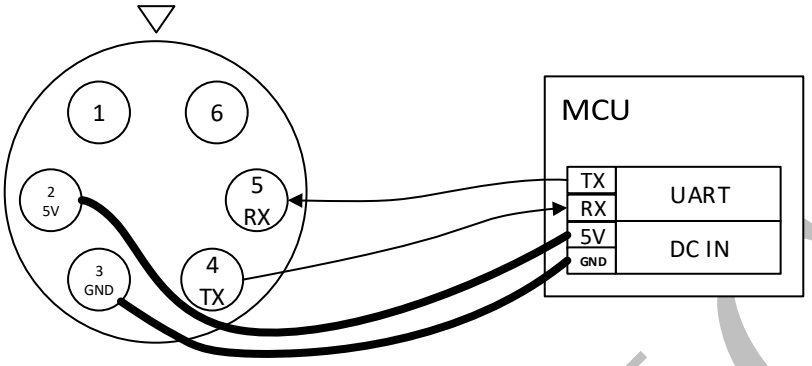
单击下一步，选择始终安装驱动。

（9）断电重启模组，驱动安装完成

在设备管理器中出现 MOTIONMARS MC1491 设备，驱动成功安装。



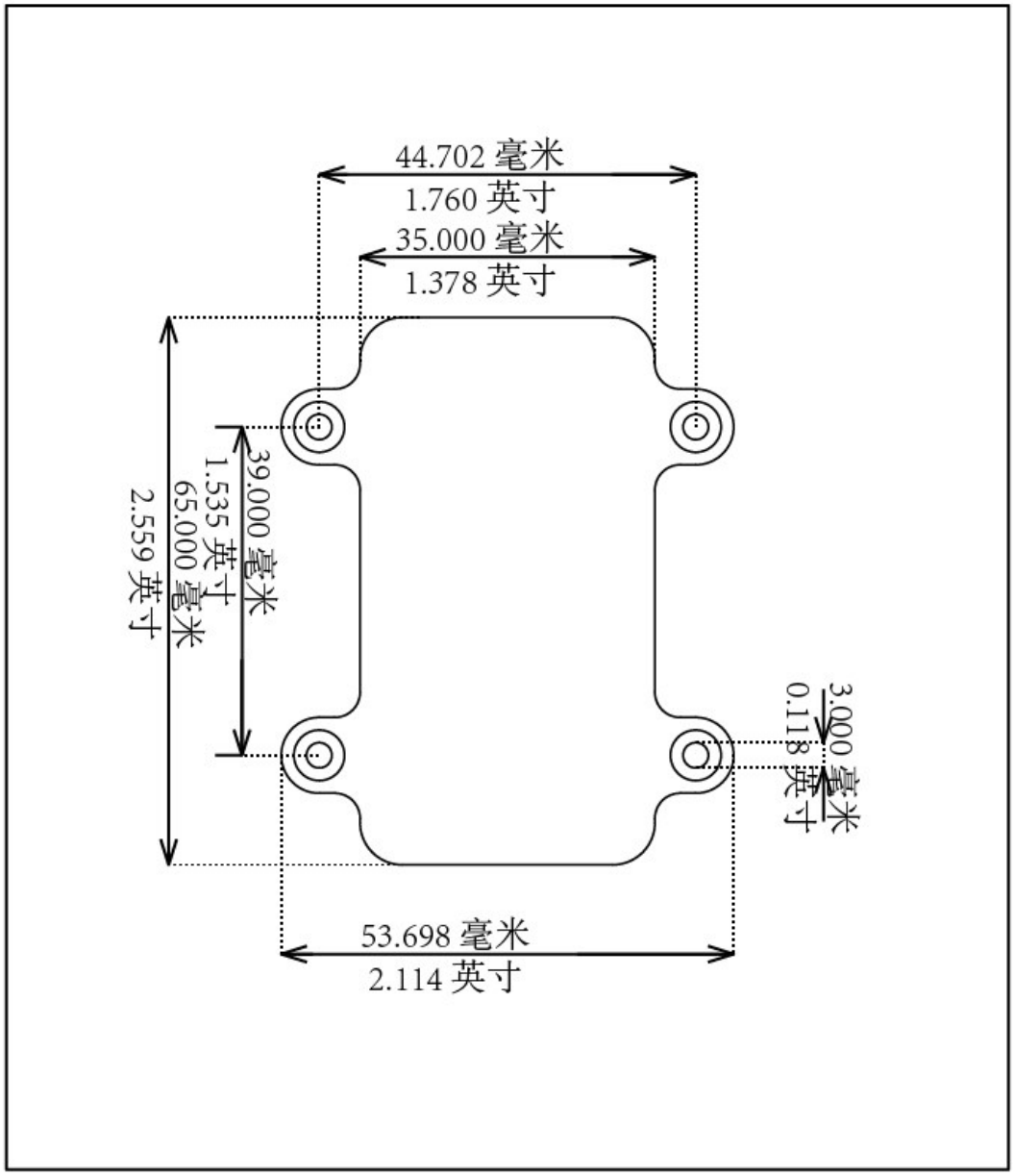
4.3 TTL 连接方式



TTL 方式通过交叉线与单片机端口连接，通过单片机发送指令与模组进行通信。

五、安装尺寸图

5.1 外观尺寸



1 更新信息: 1 2 更新信息: 2 3 更新信息: 3 4 更新信息: 4	（1）壳体高度18毫米; （2）安装螺丝Φ3x0.5mm螺距。	标题: Q1外观尺寸图 日期: Jan 5, 2019 绘制者: HMX 页数: 1 比例: 1

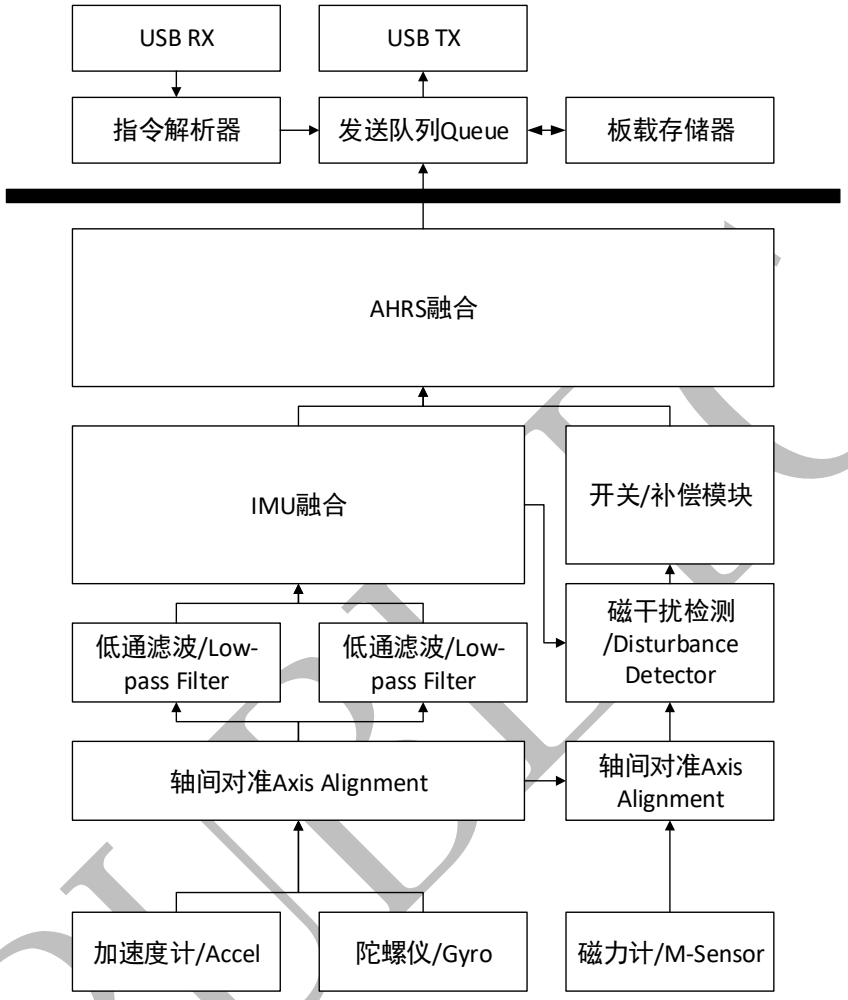
图 5.1 安装尺寸图

六、订货信息

型号	功能	品牌
MC1491-C	IMU：输出加速度、角速度、磁力计原始数据/校准后的数据（可通过串口指令切换）。	MOTIONMARS
MC1491-D	VRU：包含 MC1491-C 的所有功能。 可以输出经过内置姿态融合算法计算的俯仰、翻滚、横滚角。	MOTIONMARS

七、工作流程图

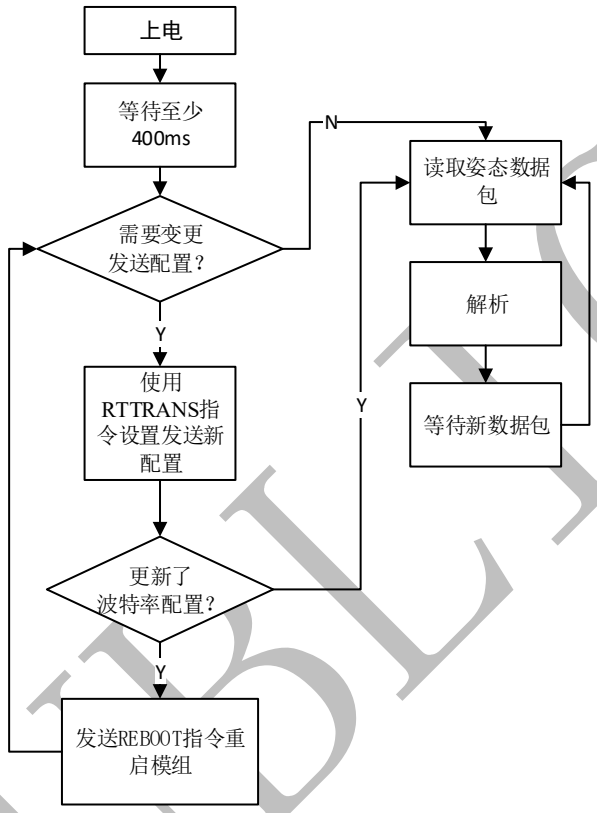
本产品可实现最大 200Hz 的 6-DoF 及 9-DoF 追踪，其工作流程如下：



八、使用流程

8.1 配置流程

以 TTL 方式连接单片机，在单片机端按照以下流程进行模块操作：



8.2 姿态数据包解析流程

根据 RTTRANS 指令设置的不同，模组上传到数据包内容也不相同，即模组的数据包长度可变。

收到一帧完整数据后，首先需计算 CRC16 是否正确（源码见附录 2），再截取数据包头部信息，根据头部信息顺序向下截取四元数、欧拉角、加速度、角速度、磁力值等信息。

头部信息的结构体如下：

```
#pragma pack(1)
typedef struct {
    uint8_t addr;
    uint8_t code;
```

```
5.
6.
7.      /*[0, 4294967295]之间增加，达到最大值后从 0 开始重新计数，
8.         注意若以 200Hz 上传数据，此值约 248 天溢出一次*/
9.      uint32_t frameNum;
10.
11.     /*用位表示是否包含数据项*/
12.     union {
13.         struct {
14.             uint16_t    quatFlag : 1;   /*Bit:0*/
15.             uint16_t    eulerFlag : 1;  /*Bit:1*/
16.             uint16_t    accelFlag : 1;  /*Bit:2*/
17.             uint16_t    gyroFlag : 1;   /*Bit:3*/
18.             uint16_t    magFlag : 1;    /*Bit:4*/
19.             uint16_t    resvFlag : 11;  /*Bit:5~11*/
20.         };
21.         uint16_t dataSendFlag;
22.     };
23. }UDIMU_UART_HDR_ND2PC;
24. #pragma pack()
```

数据解析参考代码如下：

```
1.  /************************************************************
2.  * Function: ParsePckage
3.  * Description: 姿态数据包解析参考函数
4.  * Input: buff:一帧完整数据包的首字节地址 size:一帧完整数据包长度
5.  * Output:NULL
6.  * Return:
7.  * Others:如果 crc 校验失败会中途退出
8.  ************************************************************/
9.  void ParsePckage( uint8_t* buff, uint16_t size )
10. {
11.     uint16_t    crc16          = buff[size - 2] | ( buff[size - 1] << 8 );
12.     uint16_t    crc16Expect    = CRC16( buff, size - 2 );
13.
14.     if( crc16 != crc16Expect )
15.     {
16.         /*CRC16 校验错误，不执行后续代码*/
17.         return;
18.     }
19.
```

```
    int                     frontptr = 0;
    UDIMU_UART_HDR_ND2PC    hdr;
    for( int i = 0; i < sizeof( hdr ); i++ )
    {
        *( (uint8_t *)( &hdr ) + i ) = buff[i];
    }
    /*打印帧序号，如不需要可去掉*/
    printf( "[%d]Frame Num:%d\r\n", HAL_GetTick( ), hdr.frameNum );
    frontptr += sizeof( hdr );

    /*! 最终数据!
       定义浮点数数据，由整型格式转换出浮点数表示的数据*/
    float   quat_f[4]  = { 0.0f, 0.0f, 0.0f, 0.0f };/*w,x,y,z*/
    float   euler_f[3] = { 0.0f, 0.0f, 0.0f };
    float   accel_f[3] = { 0.0f, 0.0f, 0.0f };
    float   gyro_f[3]  = { 0.0f, 0.0f, 0.0f };
    float   mag_f[3]   = { 0.0f, 0.0f, 0.0f };

    if( hdr.quatFlag )
    {
        int16_t quat[4] = { 0, 0, 0, 0 };
        memcpy( quat, &buff[frontptr], sizeof( quat ) );
        frontptr  += sizeof( quat );
        quat_f[0]  = (float)quat[0] / ( 1 << 14 );
        quat_f[1]  = (float)quat[1] / ( 1 << 14 );
        quat_f[2]  = (float)quat[2] / ( 1 << 14 );
        quat_f[3]  = (float)quat[3] / ( 1 << 14 );
        /*打印四元数，如不需要可去掉*/
        printf( "Quat:%f %f %f %f\r\n",
                quat_f[0],
                quat_f[1],
                quat_f[2],
                quat_f[3]
                );
    }

    if( hdr.eulerFlag )
    {
        int16_t euler[3] = { 0, 0, 0 };
        memcpy( euler, &buff[frontptr], sizeof( euler ) );
        frontptr  += sizeof( euler );
        euler_f[0] = (float)euler[0] / ( 1 << 7 );
```

```
62.         euler_f[1] = (float)euler[1] / ( 1 << 7 );
63.         euler_f[2] = (float)euler[2] / ( 1 << 7 );
64.         /*打印欧拉角，如不需要可去掉*/
65.         printf( "Euler:%f %f %f\r\n",
66.                 euler_f[0],
67.                 euler_f[1],
68.                 euler_f[2]
69.                 );
70.     }
71.
72.     if( hdr.accelFlag )
73.     {
74.         int16_t accel[3] = { 0, 0, 0 };
75.         memcpy( accel, &buff[frontptr], sizeof( accel ) );
76.         frontptr  += sizeof( accel );
77.         accel_f[0] = (float)accel[0] / 2048.0f;
78.         accel_f[1] = (float)accel[1] / 2048.0f;
79.         accel_f[2] = (float)accel[2] / 2048.0f;
80.         /*打印加速度，如不需要可去掉*/
81.         printf( "Accel:%f %f %f\r\n",
82.                 accel_f[0],
83.                 accel_f[1],
84.                 accel_f[2]
85.                 );
86.     }
87.
88.     if( hdr.gyroFlag )
89.     {
90.         int16_t gyro[3] = { 0, 0, 0 };
91.         memcpy( gyro, &buff[frontptr], sizeof( gyro ) );
92.         frontptr  += sizeof( gyro );
93.         gyro_f[0]  = (float)gyro[0] / 16.4f;
94.         gyro_f[1]  = (float)gyro[1] / 16.4f;
95.         gyro_f[2]  = (float)gyro[2] / 16.4f;
96.         /*打印角速度，如不需要可去掉*/
97.         printf( "Gyro:%f %f %f\r\n",
98.                 gyro_f[0],
99.                 gyro_f[1],
100.                 gyro_f[2]
101.                 );
102.     }
103.
```

```
104.     if( hdr.magFlag )
105.     {
106.         int16_t mag[3] = { 0, 0, 0 };
107.         memcpy( mag, &buff[frontptr], sizeof( mag ) );
108.         frontptr  += sizeof( mag );
109.         mag_f[0]   = (float)mag[0] / 120.0f;
110.         mag_f[1]   = (float)mag[1] / 120.0f;
111.         mag_f[2]   = (float)mag[2] / 120.0f;
112.         /*打印磁力值，如不需要可去掉*/
113.         printf( "Mag:%f %f %f\r\n",
114.                 mag_f[0],
115.                 mag_f[1],
116.                 mag_f[2]
117.                 );
118.     }
119.     /*在此可插入回调函数，将解析完成的单项或多项数据分发出去。*/
120. }
```

样例数据	
接收数据（16 进制）	解析后打印数据
01 24 02 00 00 00 1F 00 FE 3F 53 00 56 00 8C 00 4B 00 4C 00 7E 00 15 00 0B 00 80 07 FD FF 02 00 00 00 A3 FB 9F 15 E5 E8 0A 0B	[1396]Frame Num:2 Quat:0.999878 0.005066 0.005249 0.008545 Euler:0.585938 0.593750 0.984375 Accel:0.010254 0.005371 0.937500 Gyro:-0.182927 0.121951 0.000000 Mag:-9.308333 46.125000 -49.291668
01 24 03 00 00 00 1F 00 FF 3F 4E 00 05 00 8D 00 46 00 04 00 7E 00 14 00 1B 00 91 07 00 00 FB FF 00 00 BC FB A4 15 FE E8 A0 7E	[2396]Frame Num:3 Quat:0.999939 0.004761 0.000305 0.008606 Euler:0.546875 0.031250 0.984375 Accel:0.009766 0.013184 0.945801 Gyro:0.000000 -0.304878 0.000000 Mag:-9.100000 46.166668 -49.083332
01 24 04 00 00 00 1F 00 FF 3F 4E 00 ED	[3396]Frame Num:4

FF 8C 00 45 00 EE FF 7D 00 0C 00 18 00 8C 07 FE FF 03 00 FF FF 87 FB D3 15 E0 E8 89 61	Quat:0.999939 0.004761 -0.001160 0.008545 Euler:0.539063 -0.140625 0.976563 Accel:0.005859 0.011719 0.943359 Gyro:-0.121951 0.182927 -0.060976 Mag:-9.541667 46.558334 -49.333332
01 24 05 00 00 00 1F 00 FF 3F 4D 00 E1 FF 8D 00 44 00 E4 FF 7E 00 15 00 0F 00 85 07 FD FF 01 00 01 00 9B FB BF 15 EA E8 61 B7	[4396]Frame Num:5 Quat:0.999939 0.004700 -0.001892 0.008606 Euler:0.531250 -0.218750 0.984375 Accel:0.010254 0.007324 0.939941 Gyro:-0.182927 0.060976 0.060976 Mag:-9.375000 46.391666 -49.250000
01 24 06 00 00 00 1F 00 FF 3F 4D 00 DA FF 8D 00 44 00 DE FF 7E 00 0A 00 15 00 89 07 FD FF 00 00 00 00 B7 FB BF 15 A7 E8 BA C5	[5396]Frame Num:6 Quat:0.999939 0.004700 -0.002319 0.008606 Euler:0.531250 -0.265625 0.984375 Accel:0.004883 0.010254 0.941895 Gyro:-0.182927 0.000000 0.000000 Mag:-9.141666 46.391666 -49.808334

附录 1 帧格式

帧格式说明：

< >：表示必须包含的部分。

[ ]： 表示可选的部分。

一般格式为

<帧头>+<命令字>+<>+[ ].....[ ]+<CRC16>

其中：

（1）帧头：为主/从设备在 Modbus 通信局域网中的标号，在这里由于结构为单主单从，帧头设置为 0x01 即可。

（2）命令字：标识当前包的功能。

（3）CRC16：对整个帧的 CRC16 校验码，低八位在前，高八位在后。

附录 2 CRC16 校验源码

由于数据传输需要，所以数据校验必不可少；在这里约定一种统一的 CRC16 校验方法，可以移植到任何支持 ANSIC 标准编译器的平台使用。

```
#include <stdint.h>
static const uint8_t    CRCHi[] = {
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
```

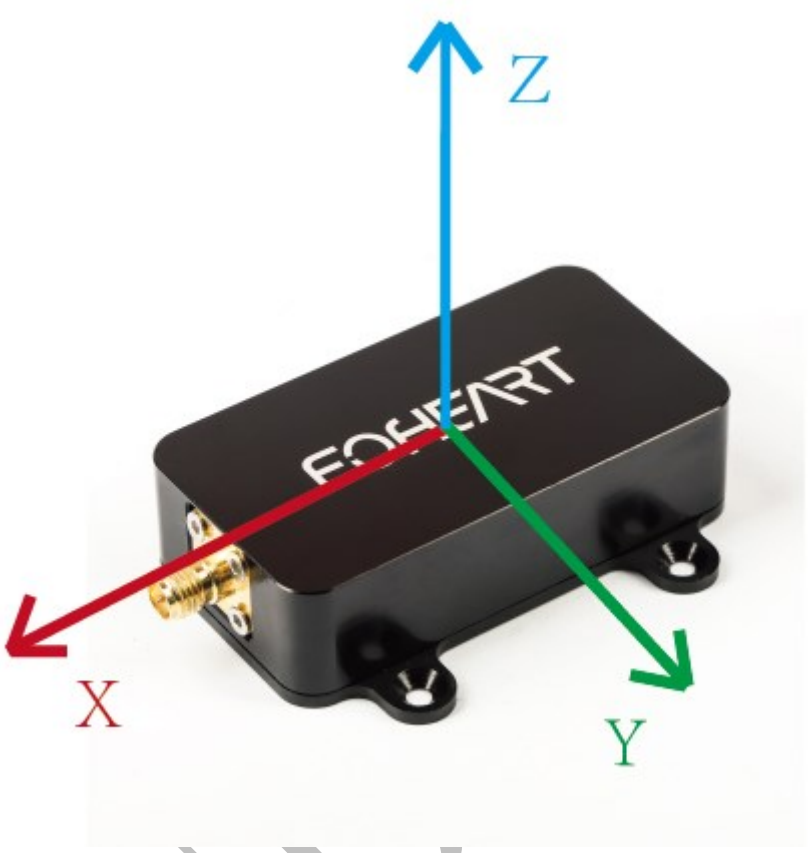
```
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40
};


static const uint8_t     CRCLo[] = {
    0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07, 0xC7,
    0x05, 0xC5, 0xC4, 0x04, 0xCC, 0x0C, 0x0D, 0xCD, 0x0F, 0xCF, 0xCE, 0x0E,
    0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09, 0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9,
    0x1B, 0xDB, 0xDA, 0x1A, 0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC,
    0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
    0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32,
    0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4, 0x3C, 0xFC, 0xFD, 0x3D,
    0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A, 0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38,
    0x28, 0xE8, 0xE9, 0x29, 0xEB, 0x2B,     x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF,
    0x2D, 0xED, 0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
    0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60, 0x61, 0xA1,
    0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64, 0xA4,
    0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F, 0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB,
    0x69, 0xA9, 0xA8, 0x68, 0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA,
```

```
    0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
    0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0,
    0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92, 0x96, 0x56, 0x57, 0x97,
    0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C, 0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E,
    0x5A, 0x9A, 0x9B, 0x5B, 0x99, 0x59, 0x58, 0x98, 0x88, 0x48, 0x49, 0x89,
    0x4B, 0x8B, 0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
    0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83,
    0x41, 0x81, 0x80, 0x40
};

/*********************************************************
* Function:             CRC16
* Description:      Return the CRC16 result of data stream
* Input:                  pucFrame:Data stream repre by hex16
* Input:                  usLen: Total length of data stream
* Output:
* Return:               CRC16 Results
* Others:               Other Description.
*********************************************************/
uint16_t CRC16( uint8_t * pucFrame, uint16_t usLen )
{
    uint8_t ucCRCHi         = 0xFF;
    uint8_t ucCRCLo         = 0xFF;
    int     iIndex;

    while( usLen-- )
    {
        iIndex      = ucCRCLo ^ *( pucFrame++ );
```

```
        ucCRCLo         = ( uint8_t )( ucCRCHi ^ CRCHi[iIndex] );
        ucCRCHi         = CRCLo[iIndex];
    }
    return ( uint16_t )( ucCRCHi << 8 | ucCRCLo );
}
```

附录 3 坐标系定义

惯性单元（IMU Sensor），其坐标系定义如下：



符合该坐标系的输出值：

（1） 加速度值

（2） 角速度值

（3） 磁力值

（4） 六轴融合下的四元数欧拉角。

九轴融合后的四元数、欧拉角坐标系符合东(X)北(Y)天(Z)坐标系定义。

(+86)010-56106165
contact@foheart.com
www.foheart.com
北京市海淀区黑山扈路红山口8号D2-南-3号