

上位机IMUPlayer使用方法

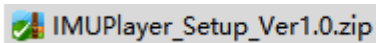
1. 适用范围

本款上位机IMUPlayer软件适用于极锐科技AGTR系列惯性测量单元。

2.IMUPlayer 使用说明

IMUPlayer是极锐科技为AGTR系列IMU开发的一款 windows 应用软件，其可以通过串口通讯查看IMU的各项数据，如加速度、角速度、姿态角度等。也可以保存 IMU 产品的数据，修改 IMU的配置参数。

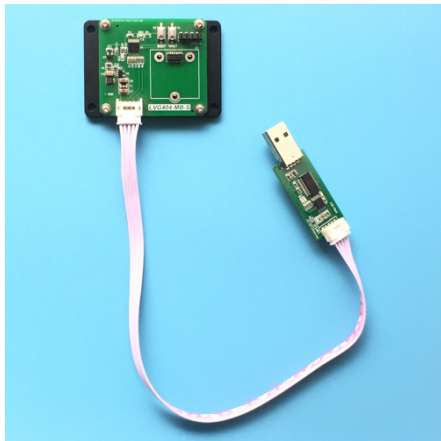
2.1 IMUPlayer 的安装



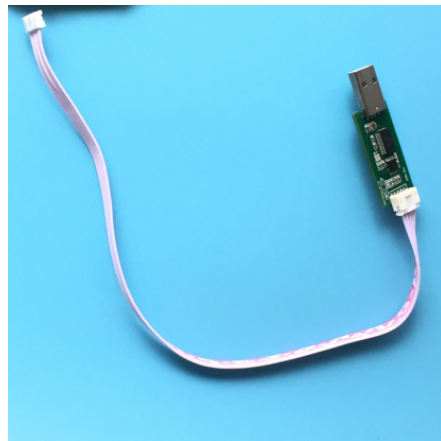
解压缩 IMUPlayer 安装程序压缩包，双击 setup.exe 文件，按照默认设置点击“下一步”进行安装，安装结束后，点击“完成”结束安装。

2.2 IMUPlayer 的启动

AGTR16460连接电脑可使用极锐科技的评估套件（即评估板+USB数据线，如下图一所示）；
AGTR350连接电脑可使用极锐科技的USB数据线，如下图二所示；
以上评估套件和USB数据线并非标配，可单独向极锐科技购买。



图一. 评估套件



图二. USB数据线



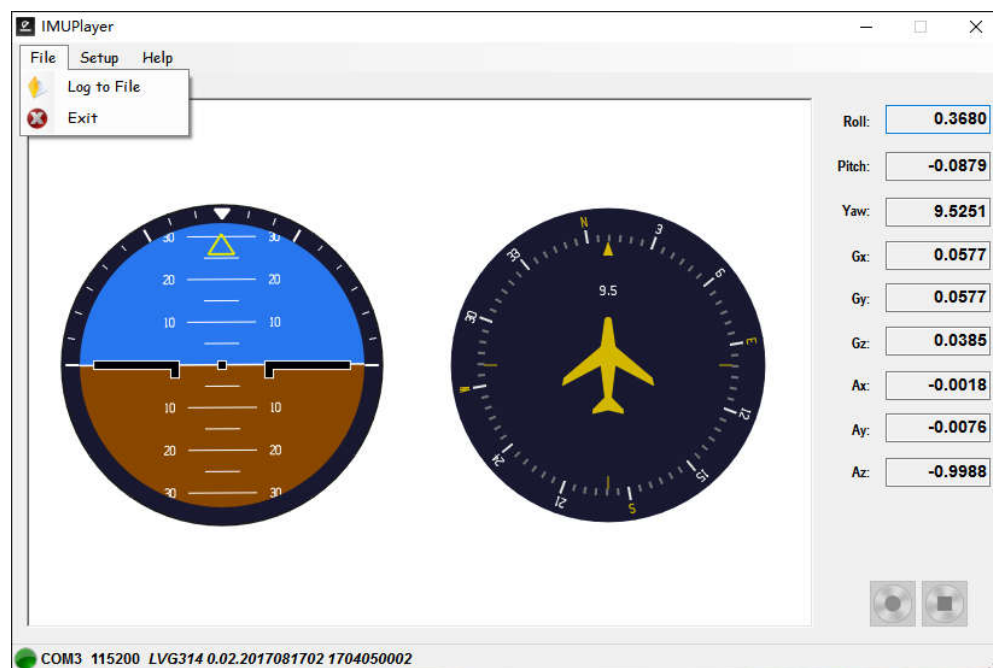
在 PC 机显示器上找到安装好的 IMUplayer 可运行程序，双击 IMUPlayer 图标启动程序，软件界面如下图：



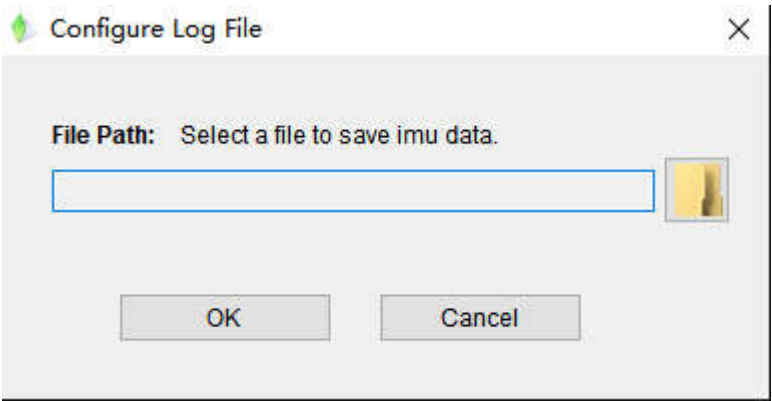
软件主界面分为菜单栏、姿态图形化显示区，数据显示区、操作按钮以及状态栏。


2.3 菜单栏

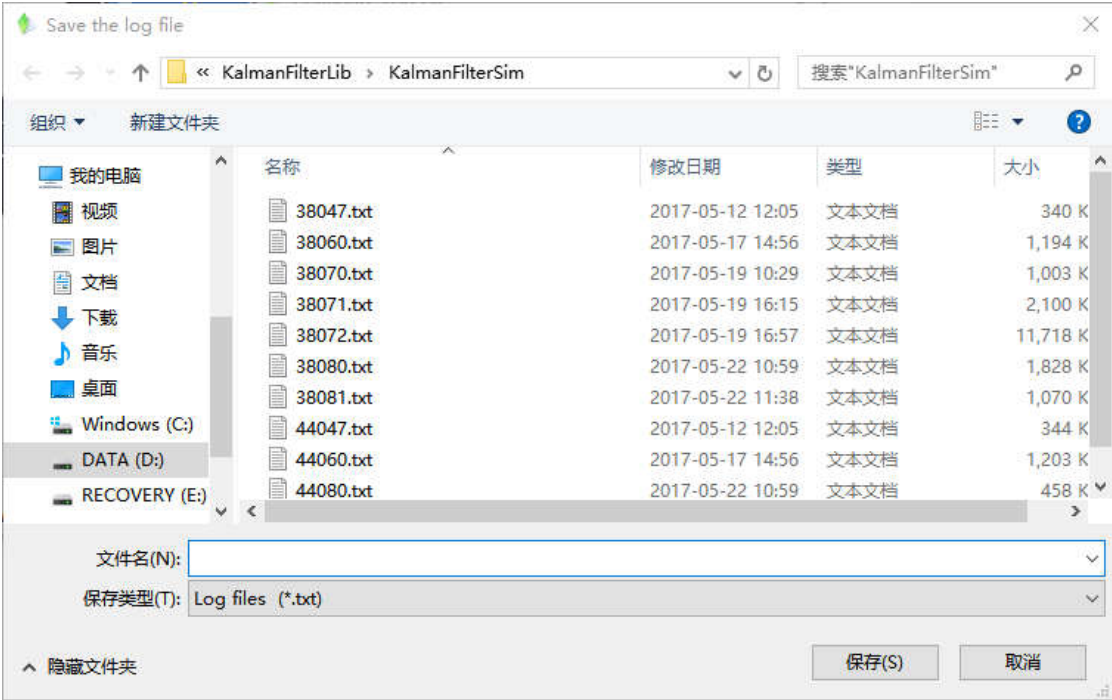
点击 File 选项，操作界面如下：



点击 Log to File 会弹出如下界面：

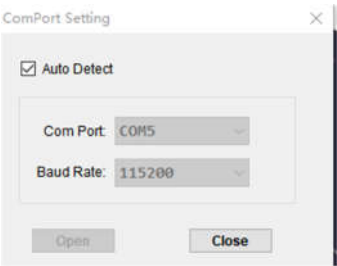


该界面主要用于设置惯性导航单元测量数据的文件保存路径。点击  按钮，会弹出文件路径选择对话框，在此选择数据保存路径。



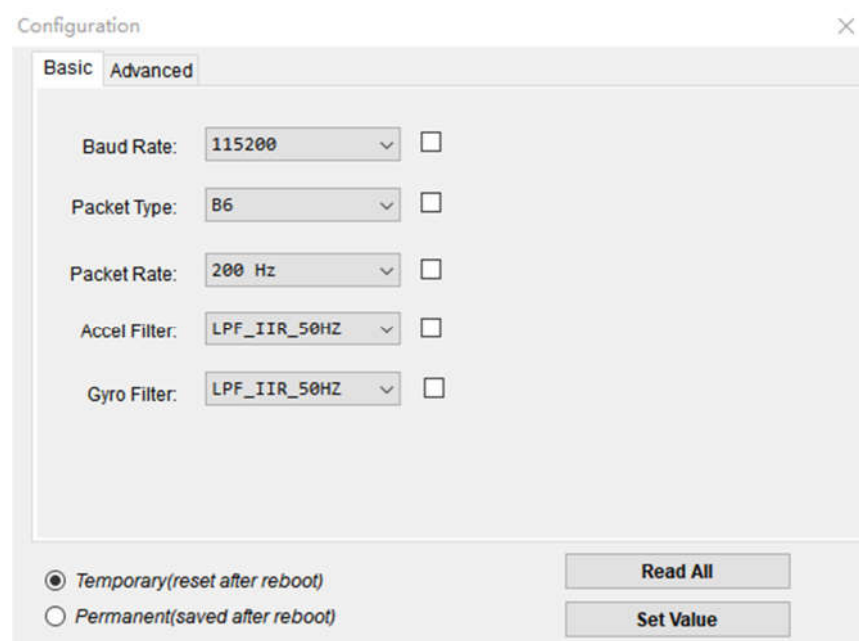
点击 Exit 选项，则关闭软件。

在初始界面中点击 Setup->ComPort 选项，可进入串口 RS232 参数设置界面，如下图：

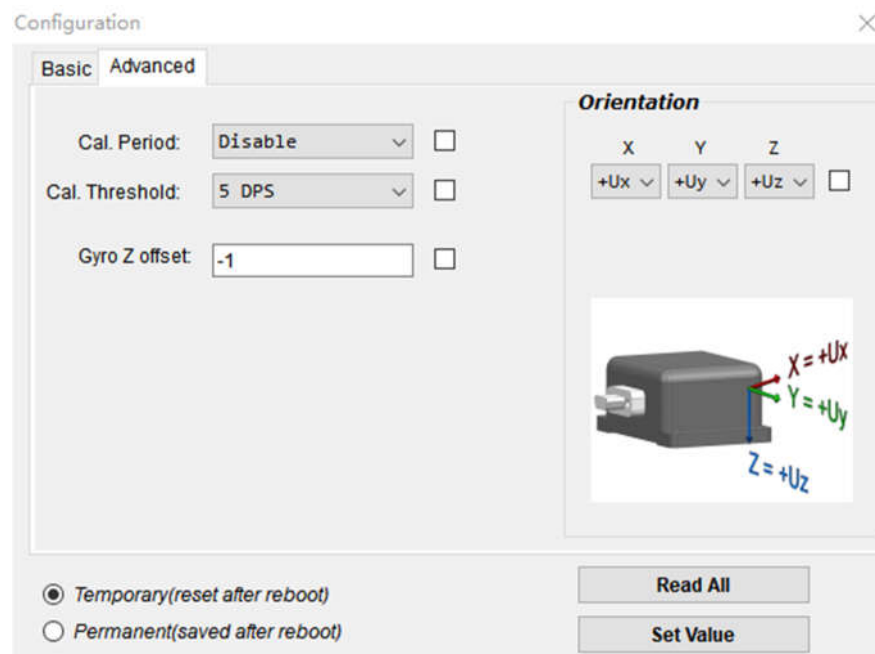


如果选择 **Auto Detect**，软件会自动遍历电脑上所有串口，自动切换波特率，自动匹配产品通讯参数。如果不选择 **Auto Detect**，可以手动选择串口号以及波特率。如果电脑只有 1-2 个少量串口号的时候，可以使用默认的 **Auto Detect** 方式，自动匹配，比较方便。如果电脑有 3 个及以上串口，可以手动选择串口号和波特率。

在初始界面中点击 **Setup->Unit Configuration** 选项，可进入产品参数设置界面，如下图：



点击 **Advanced**，界面如下：



IMU 相关参数解释如下表：

参数名	默认值	描述
Baud Rate	115200	IMU 串口通讯波特率，支持 115200，57600，38400，19200 等
Packet Type	B4	IMU 支持多种数据包，包括 A2,B3,B4,B5,B6 等
Packet Rate	200 Hz	IMU 数据传输速率可以支持 5Hz-200Hz，客户可以根据自身需要选择合适的传输速率
Accel Filter	50 Hz	IMU 内部采集传感器数据的时候，设置了低通滤波器，截止频率可以设置 5-50Hz
Gyro Filter	50 Hz	
Orientation	+Ux, +Uy,+Uz	IMU 默认方式是 NED 坐标系，连接器方向是-x，可以根据客户产品的安装方向需要，自身设置适合的方向
Cal.Period	Disable	算法内部参数，厂家内部使用，不建议客户修改
Cal.Threshold	5 DPS	算法内部参数，厂家内部使用，不建议客户修改
Gyro Z offset	-1	算法内部参数，厂家内部使用，不建议客户修改

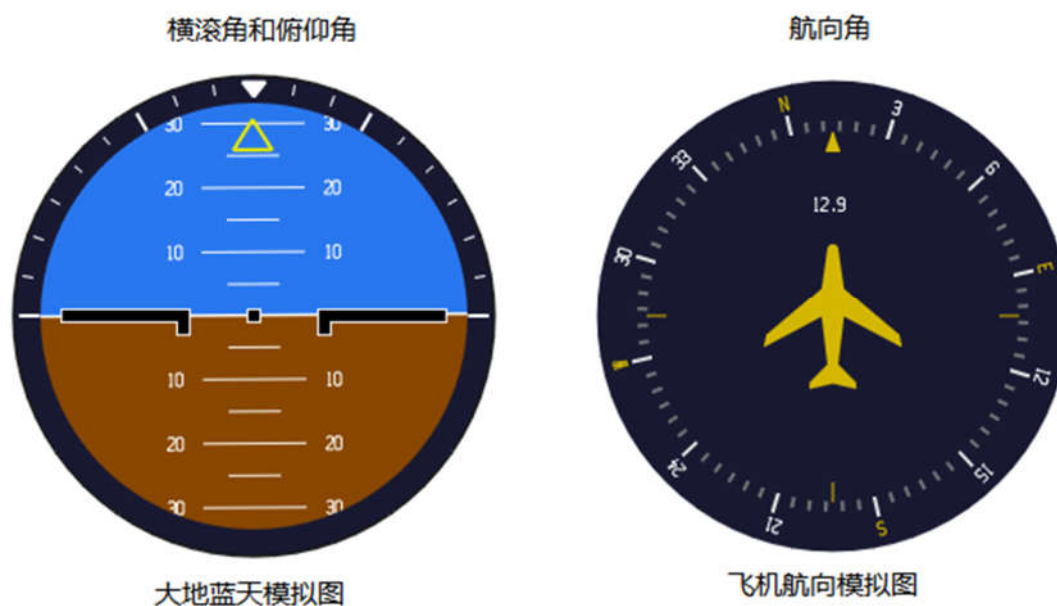
在参数设置界面，可以读取和改变惯导产品的相关参数。这些参数包括惯导产品通讯的波特率，通讯速率，数据发送类型，数字滤波器截止频率，惯导产品的坐标方向。点击 **Read All** 按钮，则会把产品所有参数读取并显示在界面上。如果需要更改某个参数，点击参数旁边相应的勾选框，然后点击 **Set Value** 按钮，进行设置。

在选择 **Temporary** 时，所有修改的参数立即起作用，惯导产品重新上电后参数将恢复到修改之前的参数。在 **Permanent** 选项下，修改完参数并点击 **set value** 按钮，将惯导产品重新上电后，修改的参数生效。

点击 **Help->About** 菜单，则会显示软件相关信息，如下图所示。



2.4 姿态显示



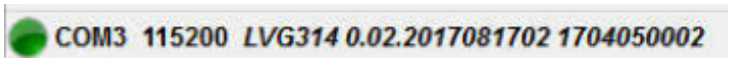
这个区域主要用于图形化直观显示惯导单元测量的横滚角（Roll）、俯仰角（Pitch）以及航向角（Yaw）。当惯导单元的姿态发生变化时，该界面可以实时显示出来相应的变化。

2.5 数据显示区域

Roll:	0.3735
Pitch:	-0.0879
Yaw:	12.8101
Gx:	-0.0385
Gy:	-0.0192
Gz:	0.0769
Ax:	-0.0034
Ay:	-0.0073
Az:	-0.9991

数据显示区域主要用于显示产品实时测量的数据，包括横滚角、俯仰角、航向角、X轴方向角速度、Y轴方向角速度、Z轴方向角速度、X轴方向加速度、Y轴方向加速度、Z轴方向加速度。其中横滚角、俯仰角以及航向角在图形区域也有显示。

2.6 状态栏



状态栏主要显示串口通讯的状态、波特率等信息。如果没有连接产品，则不会显示相关信息。

2.7 操作按钮



左边是开始记录按钮，右边是停止记录按钮。默认情况是灰色不可以点击状态，当在 File->Log to File 菜单里面选择好数据保存路径之后。开始记录按钮变成可以点击状态，如下图：



当点击开始记录按钮之后，开始记录按钮变成灰色，不可以继续点击。停止记录按钮变成可以点击状态，如下图：



此时，数据正在被记录到之前选择的数据文件里。当点击停止记录按钮之后，有恢复到原来的默认状态，如下图：



按照之前的保存文件路径找到我们保存的数据文件并打开，数据格式如下图：

AccelX	X 轴加速度值	单位是 g
AccelY	Y 轴加速度值	
AccelZ	Z 轴加速度值	
RateX	X 轴转速	单位是 dps, ° /s

RateY	Y 轴转速	
RateZ	Z 轴转速	
Roll	横滚角	单位是度
Pitch	俯仰角	
Yaw	航向角	

LVG314 0.02.2017081702 1704050002									
2017-09-14 10:32:36									
AccelX	AccelY	AccelZ	RateX	RateY	RateZ	Roll	Pitch	Yaw	
-0.0021	-0.0070	-0.9982	-0.0192	-0.0385	-0.0192	0.3735	-0.1044	15.3754	
-0.0021	-0.0073	-0.9982	-0.0769	-0.0385	-0.0192	0.3735	-0.1044	15.3754	
-0.0021	-0.0076	-0.9985	-0.0192	0.0192	-0.0385	0.3735	-0.1044	15.3754	
-0.0021	-0.0079	-0.9985	-0.0385	0.0385	0.0192	0.3735	-0.1044	15.3754	
-0.0018	-0.0073	-0.9988	-0.0385	0.0192	-0.0385	0.3735	-0.1044	15.3754	
-0.0018	-0.0070	-0.9988	0.0192	0.0192	0.0000	0.3735	-0.1044	15.3754	
-0.0018	-0.0070	-0.9988	-0.0577	0.0577	-0.0192	0.3735	-0.1044	15.3754	
-0.0021	-0.0070	-0.9988	-0.0769	0.0192	-0.0385	0.3735	-0.1044	15.3754	
-0.0021	-0.0070	-0.9988	-0.0385	0.0000	-0.0961	0.3735	-0.1044	15.3754	
-0.0024	-0.0067	-0.9988	0.0000	0.0192	-0.0385	0.3735	-0.1044	15.3754	
-0.0027	-0.0070	-0.9988	0.0000	0.0000	-0.0192	0.3735	-0.1044	15.3754	
-0.0027	-0.0073	-0.9988	-0.0192	-0.0192	0.0000	0.3735	-0.1044	15.3754	
-0.0024	-0.0073	-0.9985	-0.0961	-0.0385	0.0385	0.3735	-0.1044	15.3754	
-0.0021	-0.0073	-0.9985	-0.0769	0.0000	0.0000	0.3680	-0.1044	15.3754	
-0.0018	-0.0073	-0.9982	-0.0577	0.0192	-0.0192	0.3680	-0.1044	15.3754	
-0.0015	-0.0070	-0.9982	0.0192	0.0192	0.0000	0.3680	-0.1044	15.3754	
-0.0015	-0.0070	-0.9982	0.0192	0.0385	-0.0577	0.3735	-0.1044	15.3754	
-0.0018	-0.0070	-0.9982	0.0577	0.0769	0.0000	0.3735	-0.1044	15.3754	
-0.0021	-0.0067	-0.9982	-0.0192	0.0769	0.0192	0.3735	-0.1044	15.3754	

3. IMU 参数详细说明

Packet Type:目前产品主要支持 A2,B3,B4,B5,B6,B7,S1,F2 等协议。正常出厂默认是 B4，供常规客户使用，其他协议格式有给特殊客户定制协议，有工厂内部测试标定协议。具体协议格式和内容，请参考第 4 部分协议解析。

Accel Filter, Gyro Filter:传感器数据采集的低通滤波器，截止频率可以设置 5Hz~50Hz，客户可以根据现场的环境，来设置不同的截止频率。

Orientation: 产品坐标系方向设置。默认出厂轴设置为（Ux，Uy，Uz），由连接器指向-Ux 方向，底板指向+ Uz 方向。根据右手定则，可以设置为以下表格里的各种方向。

X Axis	Y Axis	Z Axis
+Ux	+Uy	+Uz
-Ux	-Uy	+Uz
-Uy	+Ux	+Uz
+Uy	-Ux	+Uz
-Ux	+Uy	-Uz

+Ux	-Uy	-Uz
+Uy	+Ux	-Uz
-Uy	-Ux	-Uz
-Uz	+Uy	+Ux
+Uz	-Uy	+Ux
+Uy	+Uz	+Ux
-Uy	-Uz	+Ux
+Uz	+Uy	-Ux
-Uz	-Uy	-Ux
-Uy	+Uz	-Ux
+Uy	-Uz	-Ux
-Ux	+Uz	+Uy
+Ux	-Uz	+Uy
+Uz	+Ux	+Uy
-Uz	-Ux	+Uy
+Ux	+Uz	-Uy
-Ux	-Uz	-Uy
-Uz	+Ux	-Uy
+Uz	-Ux	-Uy

4. IMU 通讯协议解析

IMU 主要支持的协议有 A2,B4,B6,S1,F2，其中 B4 是主要针对客户使用的，具体解析如下：

字节	内容	备注说明
1	同步字节 1	7FH（固定值）
2	同步字节 2	80H（固定值）
3	加速度（X 轴）低 8 位	$(20/65536) * N$ $N = \text{加速度高 8 位} * 256 + \text{加速度低 8 位}$
4	加速度（X 轴）高 8 位	
5	加速度（Y 轴）低 8 位	$(20/65536) * N$ $N = \text{加速度高 8 位} * 256 + \text{加速度低 8 位}$
6	加速度（Y 轴）高 8 位	
7	加速度（Z 轴）低 8 位	$(20/65536) * N$ $N = \text{加速度高 8 位} * 256 + \text{加速度低 8 位}$
8	加速度（Z 轴）高 8 位	
9	角速率（X 轴）低 8 位	$(1260/65536) * N$ $N = \text{角速率高 8 位} * 256 + \text{角速率低 8 位}$
10	角速率（X 轴）高 8 位	
11	角速率（Y 轴）低 8 位	$(1260/65536) * N$ $N = \text{角速率高 8 位} * 256 + \text{角速率低 8 位}$
12	角速率（Y 轴）高 8 位	
13	角速率（Z 轴）低 8 位	$(1260/65536) * N$ $N = \text{角速率高 8 位} * 256 + \text{角速率低 8 位}$
14	角速率（Z 轴）高 8 位	

15	横滚角（X 轴）低 8 位	$(360/65536) * N$
16	横滚角（X 轴）高 8 位	$N = \text{角度高 8 位} * 256 + \text{角度低 8 位}$
17	俯仰角（Y 轴）低 8 位	$(360/65536) * N$
18	俯仰角（Y 轴）高 8 位	$N = \text{角度高 8 位} * 256 + \text{角度低 8 位}$
19	航向角（Z 轴）低 8 位	$(360/65536) * N$
20	航向角（Z 轴）高 8 位	$N = \text{角度高 8 位} * 256 + \text{角度低 8 位}$
21	传感器温度低 8 位	$(200/65536) * N$
22	传感器温度高 8 位	$N = \text{温度高 8 位} * 256 + \text{温度低 8 位}$
23	校验和	除同步字外其余字节（3~22 字节）累加和取反

注：

1. 加速度值单位是 g，角速度值单位是度/秒，姿态角度单位是度。
2. 串口配置是 1bit 起始位，8bit 数据，无校验位，1bit 停止位。
3. Accel_Scale = 20. Rate_Scale = 1260. Angle_Scale = 360. Sensor_Scale = 65536.

附录：关于 B4 协议解析的示例代码，仅供客户参考

```
struct IMU_DATA
{
    double Ax; // Accel x axis
    double Ay; // Accel y axis
    double Az; // Accel z axis
    double Gx; // Gyro x axis
    double Gy; // Gyro y axis
    double Gz; // Gyro z axis
    double Roll; // Roll
    double Pitch; // Pitch
    double Yaw; // Yaw
    double Temp; // Temperature
};

bool ParseImuData(IMU_DATA* imuData, unsigned char* buffer, int length);
bool ParsePacketB4(IMU_DATA* imuData, unsigned char* buffer, int length);

// 参数说明
// imuData: 用于存储IMU数据的变量
// buffer: 串口接收IMU具体数据缓冲区
// length: 缓冲区有效数据的长度
bool ParseImuData(IMU_DATA* imuData, unsigned char* buffer, int length)
{
    static const int DATA_LENGTH = 23;

    if (imuData == NULL || buffer == NULL || length < DATA_LENGTH)
    {
        return false;
    }

    int index = 0;

    while (length >= DATA_LENGTH)
    {
        if (buffer[index++] == 0x7F && buffer[index++] == 0x80)
        {
            unsigned char sum = buffer[DATA_LENGTH - 1];
            unsigned char check = 0;

            for (int i = 2; i < DATA_LENGTH-1; i++)
            {
                check += buffer[i];
            }
        }
    }
}
```

```

    }

    check = ~check;

    if (check == sum)
    {
        ParsePacketB4(imuData, buffer + 2, DATA_LENGTH - 2);
    }

    index = 0;
    length = length - DATA_LENGTH;
    memcpy(buffer, buffer + DATA_LENGTH, length);
}
else
{
    index = 0;
    length = length - 1;
    memcpy(buffer, buffer + 1, length);
}
}

return true;
}

// 参数说明
// imuData: 用于存储IMU数据的变量
// buffer: IMU具体数据缓冲区
// length: 这次解析数据的长度
bool ParsePacketB4(IMU_DATA* imuData, unsigned char* buffer, int length)
{
    static const int Sensor_Scale = 65536;
    static const int Angle_Scale = 360;
    static const int Accel_Scale = 20;
    static const int Rate_Scale = 1260;
    static const int Temp_Scale = 200;
    static const int MinLength = 20;

    if (imuData == NULL || buffer == NULL || length < MinLength)
    {
        return false;
    }

    double sensors[10];

```

```

sensors[0] = (short)((byte)buffer[0] + ((byte)buffer[1] << 8));
sensors[1] = (short)((byte)buffer[2] + ((byte)buffer[3] << 8));
sensors[2] = (short)((byte)buffer[4] + ((byte)buffer[5] << 8));
sensors[3] = (short)((byte)buffer[6] + ((byte)buffer[7] << 8));
sensors[4] = (short)((byte)buffer[8] + ((byte)buffer[9] << 8));
sensors[5] = (short)((byte)buffer[10] + ((byte)buffer[11] << 8));
sensors[6] = (short)((byte)buffer[12] + ((byte)buffer[13] << 8));
sensors[7] = (short)((byte)buffer[14] + ((byte)buffer[15] << 8));
sensors[8] = (short)((byte)buffer[16] + ((byte)buffer[17] << 8));
sensors[9] = (short)((byte)buffer[18] + ((byte)buffer[19] << 8));

// Accel x, y, z
imuData->Ax = sensors[0] * Accel_Scale / Sensor_Scale;
imuData->Ay = sensors[1] * Accel_Scale / Sensor_Scale;
imuData->Az = sensors[2] * Accel_Scale / Sensor_Scale;

// Gyro x, y, z
imuData->Gx = sensors[3] * Rate_Scale / Sensor_Scale;
imuData->Gy = sensors[4] * Rate_Scale / Sensor_Scale;
imuData->Gz = sensors[5] * Rate_Scale / Sensor_Scale;

// Roll, Pitch, Yaw
imuData->Roll = sensors[6] * Angle_Scale / Sensor_Scale;
imuData->Pitch = sensors[7] * Angle_Scale / Sensor_Scale;
imuData->Yaw = sensors[8] * Angle_Scale / Sensor_Scale;

imuData->Temp = sensors[9] * Temp_Scale / Sensor_Scale;

return true;
}

```