FORSENSE
原极科技

IMU61X 接口协议和使用说明

目录

目录.....0
图例.....2
表例.....3
1. 串口通信协议.....4
1.1 串口接口参数.....4
1.2 数据包格式.....4
1.3 数据流帧 – AHRS 数据.....5
1.4 命令模式 GET 输出 – 系统状态.....5
1.5 命令模式 GET 输出 – 安装误差校准状态..6
1.6 命令模式 GET 输出 – 读取参数.....7
1.7 命令模式 SET 指令.....8
1.8 命令模式输出 – 用户命令响应.....10
2. SPI 通信协议.....11
2.1 SPI 接口参数.....11
2.2 SPI 连线示意图.....11
2.3 SPI 通信位序.....12
2.4 SPI 寄存器.....12
2.4.1 SPI BURST_CTRL 寄存器.....13
2.4.2 SPI BURST 寄存器.....13
2.4.3 SPI FILTER_CTRL 寄存器.....15
2.4.4 SPI ID 寄存器.....16
2.4.5 SPI WIN_CTRL 寄存器.....17
3. I2C 通信协议.....18
3.1 I2C 接口参数.....18
3.2 I2C 连接方法.....18
3.3 I2C 寄存器.....19
3.3.1 I2C BURST 寄存器.....19
3.3.2 I2C FILTER_CTRL 寄存器.....20
3.3.3 I2C ID 寄存器.....20
4. CAN 通信协议.....21
4.1 通讯参数.....21
4.2 标准帧格式.....21
5. 校准.....22
5.1 陀螺仪零偏校准.....22
5.2 安装误差校准.....22
6. 驱动和上位机.....23
6.1 驱动安装.....23
6.2 测试上位机功能介绍.....24
6.3 升级上位机功能介绍.....26
7. CRC32 查表法计算.....27

图例

图 1 SPI 连线示意图.....11
图 2 SPI 通信位序示意图.....12
图 3 SPI BURST 连续读取示意图.....13
图 4 SPI 32 位数据还原示意图.....14
图 5 I2C 连接方法.....18
图 6 I2C 连续读取模式.....19
图 7 I2C FILTER_CTRL 寄存器写入方法.....20
图 8 工厂校准、陀螺偏置校准和安装误差角校准的关系.....22

表例

表 1 串口接口参数.....4
表 2 IMU 输出和用户输入数据结构.....4
表 3 串口 AHRS 数据格式.....5
表 4 串口 A1 负载数据格式.....5
表 5 串口系统状态数据格式.....5
表 6 串口 S1 负载数据格式.....6
表 7 安装误差标定数据格式.....6
表 8 B1 负载数据格式.....6
表 9 串口参数输出数据格式.....7
表 10 串口 P1 负载数据格式.....7
表 11 串口 P1 负载参数索引表.....7
表 12 串口输入命令格式.....8
表 13 串口 R1 负载数据格式.....8
表 14 串口 R1 负载参数索引表.....9
表 15 串口用户命令响应数据格式.....10
表 16 串口 K1 负载数据格式.....10
表 17 SPI 接口参数.....11
表 18 SPI 寄存器列表.....12
表 19 SPI BURST_CTRL 寄存器格式.....13
表 20 SPI BURST 寄存器格式.....13
表 21 SPI BURST 连续读取基本格式.....14
表 22 SPI BURST 连续读取扩展格式.....14
表 23 标准帧 SPI 32 位数据转换公式.....15
表 24 SPI FILTER_CTRL 寄存器格式.....15
表 25 SPI 加速度计和陀螺仪滤器波配置.....16
表 26 SPI ID 寄存器格式.....16
表 27 SPI WIN_CTRL 寄存器格式.....17
表 28 SPI 寄存器 WIN_CTRL.WINDOW_ID 编码 17
表 29 I2C 接口参数.....18
表 30 I2C 寄存器列表.....19
表 31 I2C 连续读取数据格式.....20
表 32 I2C ID 寄存器读取模式.....20
表 33 CAN 标准帧格式 101.....21
表 34 CAN 标准帧格式 102.....21
表 35 CAN 标准帧格式 103.....21
表 36 CAN 标准帧格式 104.....21
表 37 CAN 标准帧格式 105.....21

1. 串口通信协议

基于 QT 的串口协议示例：

https://github.com/forsense/qt_serial_protocol_demo

串口通信具有两种模式：数据流模式(Stream Mode)和命令模式(Command Mode)，IMU 在上电初始化完成后，根据参数配置的模式值进入对应模式。

数据流模式：以固定频率周期性输出 AHRS 数据；

命令模式：在此模式下，停止周期性输出，用户通过发送命令与 IMU 进行通信，可通过 GET 指令获取传感器数据、状态、参数等，也可配置 IMU 的参数。

1.1 串口接口参数

表 1 串口接口参数

传输速率范围	**115200bps ~ 1.5Mbps**
默认传输速率	115200bps
开始位	1 bit
数据位	8 bits
停止位	1 bit
奇偶校验	无

1.2 数据包格式

IMU 输出和用户输入的数据包结构组成如下：

表 2 IMU 输出和用户输入数据结构

偏移量	数据类型	名称	描述
0	uint8	帧头 1	IMU 输出帧头：0xAA，0x55 用户输入帧头：0x55，0xAA
1	uint8	帧头 2	
2	uint16	ID 低位	串口通信帧 ID 的低位字节
3		ID 高位	串口通信帧 ID 的高位字节
4	uint16	数据长度低位	串口通信帧长度的低位字节，length 为 payload 所占字节数，即为 n
5		数据长度高位	串口通信帧长度的高位字节， length 为 payload 所占字节数，即为 n
6	uint8	Payload （n 个字节）	数据负载
6+n	uint32	CRC_CEHCK（32 位数据低字节）	CRC 校验
7+n		CRC_CEHCK（32 位数据中低字节）	
8+n		CRC_CEHCK（32 位数据中高字节）	
9+n		CRC_CEHCK（32 位数据高字节）	

注[1]数据以小端格式传输，低字节在前，高字节在后

注[2]crc32 的初值为 1，CRC 计算不包括本身的本帧所有数据，查表计算法见第 7 章

1.3 数据流帧 – AHRS 数据

表 3 串口 AHRS 数据格式

	帧头	帧头	ID	length	payload	帧尾
数据类型	uint8	uint8	uint16	uint16	A1	uint32
编码	0xAA	0x55	0x0002	0x002C		crc32

注1 最大输出更新率不大于 200Hz@115200bps

表 4 串口 A1 负载数据格式

offset	名称	数据类型	单位	描述
0	timer	uint32	μs	时间标
4	pitch	float	°	俯仰角
8	roll	float	°	横滚角
12	yaw	float	°	航向角
16	ax	float	g	X 轴加速度
20	ay	float	g	Y 轴加速度
24	az	float	g	Z 轴加速度
28	gx	float	°/s	X 轴角速度
32	gy	float	°/s	Y 轴角速度
36	gz	float	°/s	Z 轴角速度
40	temp	float	℃	IMU 芯片温度

1.4 命令模式 GET 输出 – 系统状态

表 5 串口系统状态数据格式

	帧头	帧头	ID	length	payload	帧尾
数据类型	uint8	uint8	uint16	uint16	S1	uint32
编码	0xAA	0x55	0x00FF	0x0042		crc32

表 6 串口 S1 负载数据格式

offset	名称	数据类型	描述
0	Software_ver	uint32	软件版本号
4	Hardware_ver	uint32	硬件版本号
8	Board_ver	uint16	IMU 型号
10	sn0	uint32	第一 SN 号
14	sn1	uint32	第二 SN 号
18	sn2	uint32	第三 SN 号
22	system_status	uint32	系统状态
26	cpu_load	uint16	CPU 占用率
28	drop_rate_com	uint16	通信误码计数
30	rev[9]	uint32	保留字节

1.5 命令模式 GET 输出 – 安装误差校准状态

表 7 安装误差标定数据格式

	帧头	帧头	ID	length	payload	帧尾
数据类型	uint8	uint8	uint8	uint16	——	uint32
编码	0xAA	0x55	0x00B2	0x0058	B1	crc32

表 8 B1 负载数据格式

offset	域名	数据类型	单位	描述
0	timer	uint32	ms	时间标
4	rev[24]	uint8	–	6 个保留字节
28	Install1	uint16	–	安装误差位置 1 计数
30	Install2	uint16	–	安装误差位置 2 计数
32	Install3	uint16	–	安装误差位置 3 计数
34	Install4	uint16	-	安装误差位置 4 计数
36	Install5	uint16	-	安装误差位置 5 计数
38	Install6	uint16	-	安装误差位置 6 计数
40	rev[48]	uint8	-	48 个保留字节

1.6 命令模式 GET 输出 – 读取参数

表 9 串口参数输出数据格式

	帧头	帧头	ID	length	payload	帧尾
数据类型	uint8	uint8	uint16	uint16	P1	uint32
编码	0xAA	0x55	0x0006	0x0018		crc32

表 10 串口 P1 负载数据格式

offset	名称	数据类型	描述
0	Param1	float	设置的参数
4	Param2	float	保留，默认为 0
8	Param3	uint32	设置的参数索引
12	Param4	uint32	保留，默认为 0
16	Param5	Int32	保留，默认为 0
20	Param6	Int32	保留，默认为 0

表 11 串口 P1 负载参数索引表

Param3	Param1	单位
3	串口输出波特率，支持以下波特率: 115200 230400 460800 921600 1500000	bps
8	X 轴陀螺零偏标定结果，GYRO_X_OFF	°/s
9	Y 轴陀螺零偏标定结果，GYRO_Y_OFF	°/s
10	Z 轴陀螺零偏标定结果，GYRO_Z_OFF	°/s
16	X 轴安装误差角，INSTALL_X_OFF	°
17	Y 轴安装误差角，INSTALL_Y_OFF	°
18	Z 轴安装误差角，INSTALL_Z_OFF	°
21	AHRS 输出频率，默认 100Hz	Hz
27	SPI 或 I2C 模式 0：不支持 SPI 或 I2C，只支持串口 1：支持串口和 SPI 2：支持串口和 I2C 3：通过外部 IO 口设置模式，悬空或拉低引脚选择 SPI 模式，拉高引脚选择 I2C 模式	
30	I2C 地址，默认为 0x18	
31	内部滤波器配置，定义同 SPI 的 FILTER_CTRL 对照表	

1.7 命令模式 SET 指令

表 12 串口输入命令格式

	帧头	帧头	ID	length	payload	帧尾
数据类型	uint8	uint8	uint16	uint16	R1	uint32
编码	0x55	0xAA	CMD	0x0018		crc32

注1 CMD 与 R1 关系，详见 R1 负载参数索引表

表 13 串口 R1 负载数据格式

offset	名称	数据类型	描述
0	Param1	float	设置的参数
4	Param2	float	保留，默认为 0
8	Param3	uint32	设置的参数索引
12	Param4	uint32	保留，默认为 0
16	Param5	Int32	保留，默认为 0
20	Param6	Int32	保留，默认为 0

表 14 串口 R1 负载参数索引表

CMD	Param1	Param3	描述
1	0	0	触发获取一次系统状态数据
2	0	0	触发获取一次 AHRS 数据
3	<mode>	0	设置输出模式： Mode=1，数据流输出 AHRS Mode=100，禁止数据流模式，进入 COMMAD 模式
4	<heading>	0	重置 AHRS 的航向角： heading 输入范围为 0 ~ 360，单位为度
5	0	0	保存当前参数到 FLASH
6	0	<value>	读取参数，value 为要读取的参数索引，即 P1.index，详见 串口应答性输出-参数读取 例如需读取 AHRS 输出频率（ODR），则设置 value=21
7	0	0	开始安装误差标定
8	0	0	触发获取一次安装误差标定状态
9	0	0	执行软件重启
10	0	0	执行静态陀螺仪零偏校准
14	<value>	3	设置串口输出波特率，单位 bps value 的有效值为： `115200, 230400, 460800, 921600, 1500000` value 为其他值时，默认采用 115200bps
14	<value>	21	设置周期性 AHRS 数据输出频率，单位 Hz value 的有效值为： `1, 10, 50, 100, 200, 400` value 为其他值时，默认采用 100Hz
14	<value>	27	设置 SPI 或 I2C 模式 value=0，不支持 SPI 或 I2C，只支持串口 value=1，支持串口和 SPI value=2，支持串口和 I2C Value=3，表示根据外部 IO 口配置 SPI 或 I2C 模式 默认 value=1，即支持串口和 SPI
14	<value>	30	设置 I2C 地址，value 默认值为 0x18，其他有效值为： `0x18, 0x19, 0x1A, 0x1B`
14	<value>	31	内部滤波器配置，定义同 SPI 加速度计和陀螺仪滤波器配置，默认 0xBB，即 47Hz
15	0	0	恢复出厂设置参数

注 1 请注意本表中数值均为十进制

注 2 可使用上位机命令生成器功能生成对应命令发送，使用方法见本手册上位机使用部分

1.8 命令模式输出 – 用户命令响应

表 15 串口用户命令响应数据格式

	帧头	帧头	ID	length	payload	帧尾
数据类型	uint8	uint8	uint16	uint16	K1	uint32
编码	0xAA	0x55	0x0064	0x0007		crc32

表 16 串口 K1 负载数据格式

offset	名称	数据类型	描述
0	command	uint16	待响应的命令 ID
2	result	uint16	结果

2. SPI 通信协议

基于 STM32 的 SPI 主机读取驱动示例：

https://github.com/forsense/IMU61x_SPI_I2C_CAN_Drivers

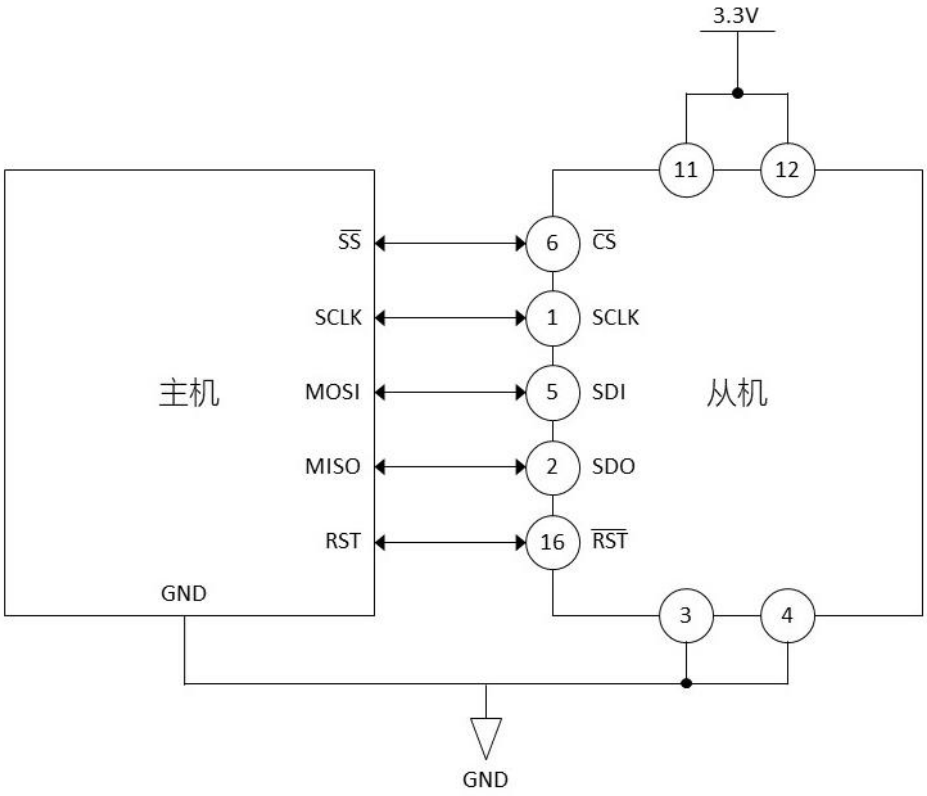
2.1 SPI 接口参数

表 17 SPI 接口参数

SPI 主机	本产品作为从机
SPI 速率	0.2 ~ 2MHz
SPI 字长	16 bit
相位	上升沿触发
极性	空闲为低电平
位序	MSB 优先

2.2 SPI 连线示意图

图 1 SPI 连线示意图

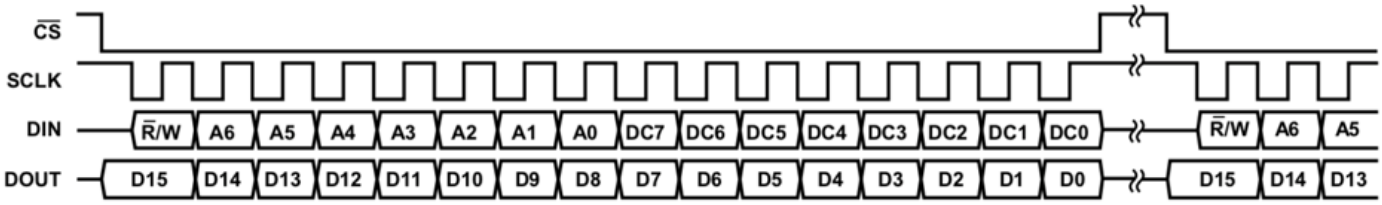


注初始化读取前，需将 IMU 复位并等待 3s，使得 IMU 进入正常工作状态。

2.3 SPI 通信位序

SPI 接口支持全双工串行通信（同时执行发送和接收），采用下图所示的位序。

图 2 SPI 通信位序示意图



其中，DIN 最高位表示读/写操作，[A6:A0]表示寄存器地址，[DC7:DC0]表示写入的数据（写操作）或 DUMMY 数据（读操作）。

当 $\overline{R}/W = 1$时，此 SPI 周期的 DOUT 数据无意义。当 $\overline{R}/W = 0$时，此 SPI 周期的 DOUT 数据表示上两个周期的寄存器输出数据，具体见 BURST 读取示例。

2.4 SPI 寄存器

表 18 SPI 寄存器列表

名称	地址	读/写	默认值	窗 ID	描述
BURST	0x00	RW		0	连续读取
BURST_CTRL	0x0D,0x0C	RW	0x0000	1	连续读取配置
FILTER_CTRL	0x07,0x06	RW	0x00BB	1	滤波器选择
PROD_ID1	0x6A	R	0x0000	1	ID 号 1
PROD_ID2	0x6C	R	0x494d	1	ID 号 2
PROD_ID3	0x6E	R	0x5536	1	ID 号 3
PROD_ID4	0x70	R	0x3132	1	ID 号 4 (IMU612)
			0x3134	1	ID 号 4 (IMU614)
			0x3138	1	ID 号 4 (IMU618)
			0x3141	1	ID 号 4 (IMU6132A)
			0x3142	1	ID 号 4 (IMU6132B)
WIN_CTRL	0x7F,0x7E	RW	0x0000	0，1	窗 ID 选择

2.4.1 SPI BURST_CTRL 寄存器

BURST_CTRL 寄存器用于控制 BURST 寄存器的输出格式

- 当 ATT 为 0 时（默认值），BURST 寄存器输出为基本格式
- 当 ATT 为 1 时，BURST 寄存器输出为扩展格式

表 19 SPI BURST_CTRL 寄存器格式

地址	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	读/写
0x0D								ATT	RW
地址	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	读/写
0x0C									RW

2.4.2 SPI BURST 寄存器

BURST 为连续读取寄存器，在一个数据流中读取所有数据，各 16 位段之间无停转。

表 20 SPI BURST 寄存器格式

地址	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	读/写
0x01									RW
地址	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	读/写
0x00	BURST_CMD								RW

BURST 读取方法是：读取前发送 0x8000 表示设置 BURST 并开始读取，然后一直发送 0x0000 并接收数据，输出寄存器内容比读取指令发送偏移 2 个 SPI 周期，读取期间一直持续片选低电平。

图 3 SPI BURST 连续读取示意图

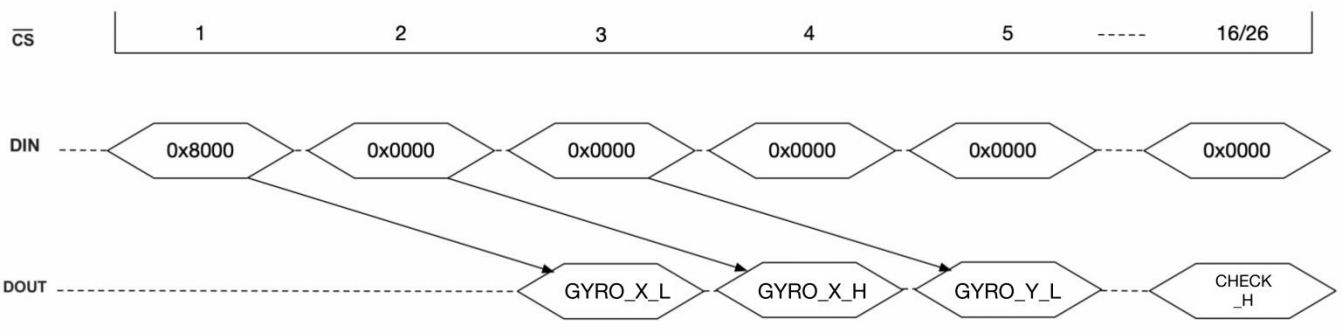


表 21 SPI BURST 连续读取基本格式

发送顺序	1	2	3	4	5	6
发送内容	GYRO_X_L	GYRO_X_H	GYRO_Y_L	GYRO_Y_H	GYRO_Z_L	GYRO_Z_H
	7	8	9	10	11	12
	ACCL_X_L	ACCL _X_H	ACCL _Y_L	ACCL _Y_H	ACCL _Z_L	ACCL _Z_H
	13	14				
	CHKSM_L	CHKSM_H				

注[1] 所有数据均为 16-bit 宽度

注[2] 陀螺、加速度计数据拼接后格式表示为 int32

注[3] 所 CHKSM 即 CHECKSUM，用于确认数据完整性。计算方法为将 CHECKSUM 之前的所有数据累加求和

表 22 SPI BURST 连续读取扩展格式

发送顺序	1	2	3	4	5	6
发送内容	GYRO_X_L	GYRO_X_H	GYRO_Y_L	GYRO_Y_H	GYRO_Z_L	GYRO_Z_H
	7	8	9	10	11	12
	ACCL_X_L	ACCL _X_H	ACCL _Y_L	ACCL _Y_H	ACCL _Z_L	ACCL _Z_H
	13	14				
	TEMP_L	TEMP_H				
	15	16	17	18	19	20
	ATT_X_L	ATT_X_H	ATT_Y_L	ATT_Y_H	ATT_Z_L	ATT_Z_H
	21	22	23	24		
	STATUS_L	STATUS_H	CRC32_L	CRC32_H		

注[1] 所有数据均为 16-bit 宽度

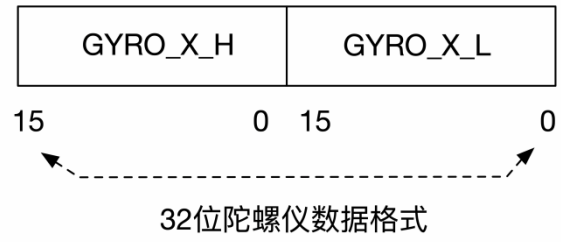
注[2] 陀螺、加速度计、温度、姿态数据拼接后格式表示为 float，状态数据拼接后表示为 uint32

注[3] TEMP 单位为℃，陀螺仪输出单位为 $°/s$，加速度计输出单位为 g，姿态输出单位为度

注[4] crc32 的初值为 1，CRC 计算不包括本身的本帧所有数据，查表计算法见附录 1

在 BURST 连续读取过程中，32 位的完整数据被拆分成高 16 位和低 16 位分别输出，输出时采用小端模式，即低字节先输出。用户需要将这两部分 16 位数据首尾拼接，还原出完整的 32 位数据。

图 4 SPI 32 位数据还原示意图



得到完整的 32 位数据后，标准帧用户可根据以下公式将其转换为角速度、加速度、温度和姿态角信息。

表 23 标准帧 SPI 32 位数据转换公式

名称	单位	公式	条件/备注
角速度	°/s	$G = \frac{SF}{65536} \times \mathrm{GYRO}$	▪ GYRO 为上表中 X/Y/Z 轴的 GYRO 数据 ▪ 陀螺刻度因子$SF = 0.016$
加速度	mg	$A = \frac{SF}{65536} \times \mathrm{ACCL}$	▪ ACCL 为上表中 X/Y/Z 轴的 ACCL 数据 ▪ 加速度计刻度因子$SF = 0.2$
温度	℃	$T = \frac{SF}{65536} \times (\mathrm{TEMP} - 172621824) + 25$	▪ TEMP 为上表中的 TEMP 数据 ▪ 温度刻度因子$SF = -1/263.4$
姿态角	°	$D = \frac{SF}{65536} \times \mathrm{ATT}$	▪ ATT 为上表中 ATT 数据 ▪ 姿态刻度因子$SF = 0.00699411$

2.4.3 SPI FILTER_CTRL 寄存器

FILTER_CTRL 寄存器为用户提供对数字低通滤波器的控制。 此寄存器为可读/写寄存器，写命令为发送 0x86XX，且当前 SPI 周期设置有效；读命令发送 0x0600，输出寄存器内容比读取指令发送偏移 2 个 SPI 周期。

表 24 SPI FILTER_CTRL 寄存器格式

地址	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	读/写
0x07									RW
地址	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	读/写
0x06	加速度计滤波器配置				陀螺仪滤波器配置				RW

表 25 SPI 加速度计和陀螺仪滤器波配置

	编码	描述
加速度计/陀螺仪滤波器配置	4'b 0000	IIR filter fc=1.0Hz
	4'b 0011	IIR filter fc=5.0 Hz
	4'b 0100	IIR filter fc=10.0 Hz
	4'b 1001	IIR filter fc=15 Hz
	4'b 1010	IIR filter fc=20 Hz
	4'b 0111	IIR filter fc=25 Hz
	4'b 1011	IIR filter fc=31 Hz for gyroscope, 40hz for accelerometer
	4'b 1100	MOVING AVERAGE FILTER TAP=4
	4'b 1101	MOVING AVERAGE FILTER TAP=16
	4'b 1110	MOVING AVERAGE FILTER TAP=32
	4'b 1111	MOVING AVERAGE FILTER TAP=64

2.4.4 SPI ID 寄存器

ID 寄存器为只读寄存器，数据内容为 ASCII 编码形式的字符 "IMU61x", 读取方法类似 BURST 数据读取：读取时发送 0x6A00~0x7000，并接收数据。输出寄存器内容比读取指令发送偏移 2 个周期。

将 4 个 16 位 ID 数据拼接后转为 ASICII 码，可获得产品的完整 ID。拼接方法同 BURST 连续读取数据的拼接，PROD_ID1 在高位，PROD_ID4 在低位。

表 26 SPI ID 寄存器格式

地址	bit15 ~ bit0	编码	读/写
0x6A	PROD_ID1	0x0000	R
0x6C	PROD_ID2	0x494D	R
0X6E	PROD_ID3	0x5536	R
0x70	PROD_ID4 编码内容代表产品 ID	0x3132(IMU612)	R
		0x3134(IMU614)	
		0x3138(IMU618)	
		0x3141(IMU6132A)	
		0x3142(IMU6132B)	

2.4.5 SPI WIN_CTRL 寄存器

此寄存器用于控制切换窗口 ID，可读可写。窗口默认为 0，写入 0xFE01，则切换为 1。

表 27 SPI WIN_CTRL 寄存器格式

地址	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	读/写
0x7F									RW
地址	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	读/写
0x7E	WINDOW_ID								RW

表 28 SPI 寄存器 WIN_CTRL.WINDOW_ID 编码

名称	编码	描述
WINDOW_ID	0x00	window0，开始读取数据
	0x01	window1，进入配置

3. I2C 通信协议

基于 STM32 的 I2C 主机读取驱动示例：

https://github.com/forsense/IMU61x_SPI_I2C_CAN_Drivers

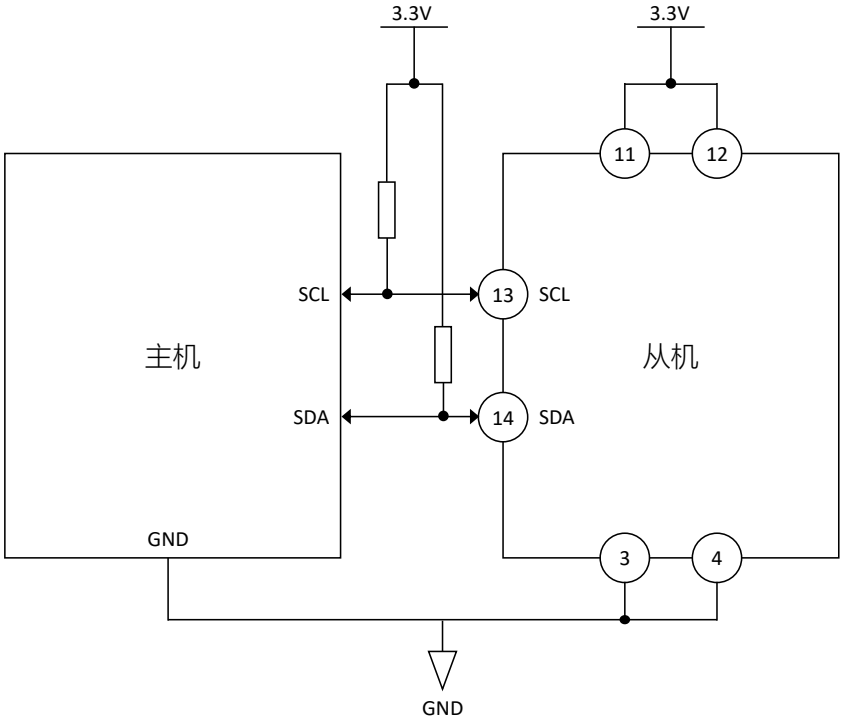
3.1 I2C 接口参数

表 29 I2C 接口参数

I2C 速率	400KHz
I2C 从机地址（7 位）	0x18

3.2 I2C 连接方法

图 5 I2C 连接方法



备注：上拉电阻阻值为 4.7KΩ

3.3 I2C 寄存器

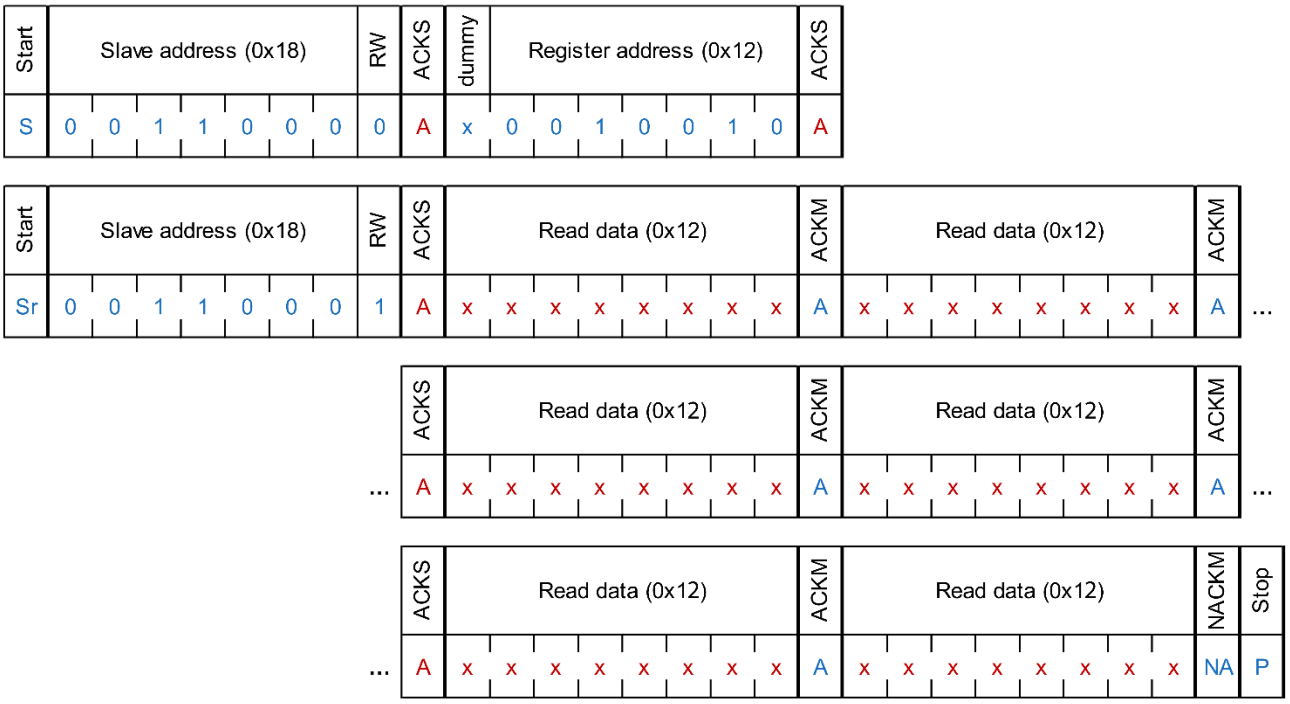
表 30 I2C 寄存器列表

名称	地址	读/写	默认值	描述
BURST	0x00	R		连续读取寄存器
FILTER_CTRL	0x06	RW	0xBB	滤波器选择
PROD_ID	0x6A	R		产品名称

3.3.1 I2C BURST 寄存器

本 I2C 协议支持连续读取，连续读寄存器地址 0x12，从机自动累加地址，以 8bit 模式连续输出 32 个字节，读取过程如下：

图 6 I2C 连续读取模式



帧定义如下：

表 31 I2C 连续读取数据格式

发送顺序	1	2	3	4	5	6
数据格式	float	float	float	float	float	float
发送内容	ACCL_X	ACCL_Y	ACCL_Z	GYRO_X	GYRO_Y	GYRO_Z
	7	8				
	float	uint32				
	TEMP	CRC32				

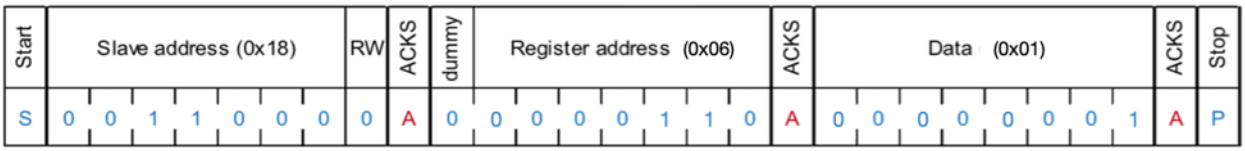
注[1] TEMP 单位为℃，陀螺仪输出单位为 $°/s$，加速度计输出单位为 g，姿态输出单位为度

注[2] crc32 的初值为 1，CRC 计算不包括本身的本帧所有数据，查表计算法见附录 1

3.3.2 I2C FILTER_CTRL 寄存器

FILTER_CTRL 寄存器地址为 0x06，滤波器配置对照表同 SPI 加速度计和陀螺仪滤器波配置。寄存器读取过程同 I2C BURST 读取方法，写寄存器过程如下图所示。

图 7 I2C FILTER_CTRL 寄存器写入方法



3.3.3 I2C ID 寄存器

ID 寄存器地址为 0x6A，数据内容为 ASCII 编码形式的字符 “IMU61B”，读取过程同 I2C BURST，如下表所示。

表 32 I2C ID 寄存器读取模式

发送顺序	1	2	3	4
发送内容	0x00	0x00	0x49	0x4D
	5	6	7	8
	0x55	0x36	0x31	0x*

注[1] 所有数据均为 8-bit 宽度

注[2] 0x*表示的内容为产品 ID，0x32 代表 IMU612，0x34 代表 IMU614，0x38 代表 IMU618，0x41 代表 IMU6132A，0x42 代表 IMU6132B

4. CAN 通信协议

基于 STM32 的 CAN 主机读取驱动示例：

https://github.com/forsense/IMU61x_SPI_I2C_CAN_Drivers

4.1 通讯参数

- 接口形式：CAN，标准帧
- CAN 速率：1Mbps

4.2 标准帧格式

表 33 CAN 标准帧格式 101

标准帧 ID	1	2	3	4	5	6	7	8
101	ROLL				PITCH			

表 34 CAN 标准帧格式 102

标准帧 ID	1	2	3	4	5	6	7	8
102	YAW				Gx			

表 35 CAN 标准帧格式 103

标准帧 ID	1	2	3	4	5	6	7	8
103	Gy				Gz			

表 36 CAN 标准帧格式 104

标准帧 ID	1	2	3	4	5	6	7	8
104	Ax				Ay			

表 37 CAN 标准帧格式 105

标准帧 ID	1	2	3	4	5	6	7	8
105	Az				TEMP		INDEX	

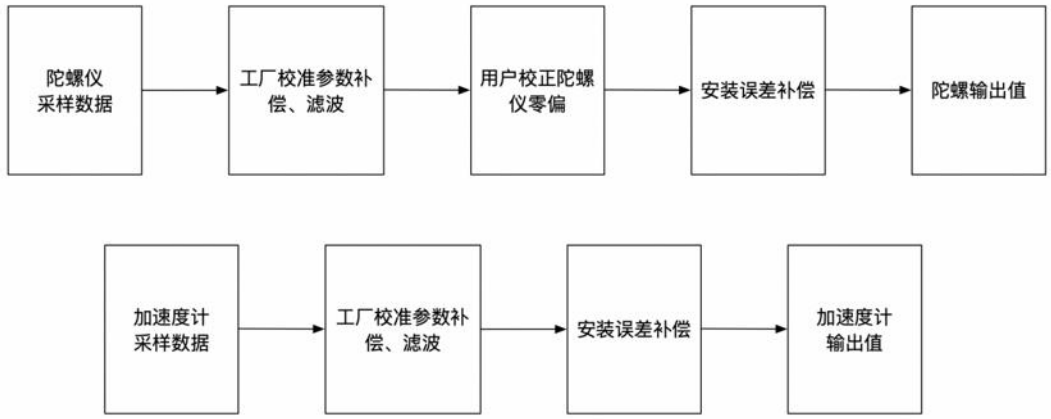
注[1]姿态角、陀螺、加速度计数据表示为 float，温度、计数值数据表示为 int16

注[2]TEMP 单位为 100*℃，陀螺仪输出单位为 $°/s$，加速度计输出单位为 g，姿态输出单位为度

5. 校准

IA 系列产品在出厂前会经过全温域的（-40℃至 85℃）精密温度补偿，各轴陀螺仪和加速度计的灵敏度、零偏、非正交误差也已经过转台标定，输出参数都已经过补偿公式校正，已经具有出色的性能和稳定性。
同时为了方便用户深层开发，本产品提供标准接口，可对静态陀螺仪零偏校准和系统安装误差标定进行标定，本操作目前仅支持通过串口。

图 8 工厂校准、陀螺偏置校准和安装误差角校准的关系



5.1 陀螺仪零偏校准

陀螺零偏校准指令通过上文中串口 R1 负载参数索引表表给出，校准过程需要静止 20 秒，IMU 会收集陀螺仪数据并求取平均值，校准完毕后，通过串口 P1 负载参数索引表中的 GYRO_X_OFF、GYRO_Y_OFF 和 GYRO_Z_OFF 查看。校准过程中，产品附近的振动、电源波动、热梯度等因素，都有可能影响校准精度。

- 请注意，校准的陀螺偏置参数不需要用户补偿，本 IMU 系统输出时已补偿过。

5.2 安装误差校准

本 IMU 使用时应尽量将 IMU 坐标轴与载体坐标轴对准，消除安装误差。如果不能保证对齐，且载体可以整体放置到转台上，可以利用转台的重力加速度矢量提供基准对安装误差进行标定。
本 IMU 提供了标定流程，并给出标定过程和结果，具体操作流程见测试上位机的安装误差标定界面。标定完毕后，系统自动计算安装误差校正矩阵，用户可通过串口 P1 负载参数索引表中 INSTALL_X_OFF，INSTALL_Y_OFF 和 INSTALL_Z_OFF 查看三轴安装误差角。

- 请注意，校准后的安装误差角不需要用户补偿，本 IMU 的输出已经过安装误差系数矩阵并的计算补偿。
- 请注意，本产品只支持对 3 度以内的安装偏差角标定。因此在进行安装偏差校准前，请通过安装方式保持结构上的轴对齐，然后执行安装偏差校准。

6. 驱动和上位机

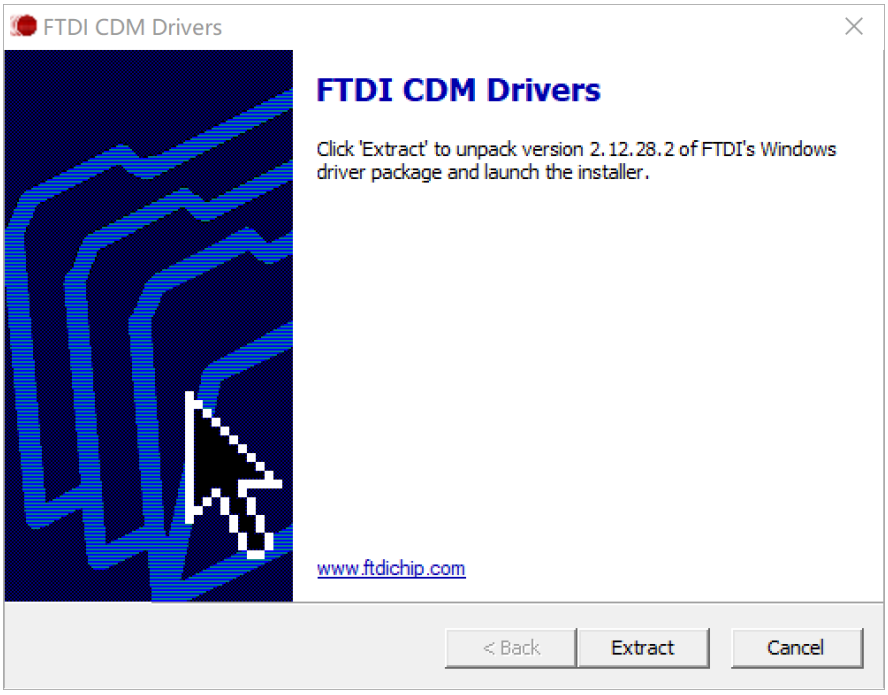
驱动、测试和升级上位机可在 github 下载最新版：

https://github.com/forsense/IMU61x_PC_Tools

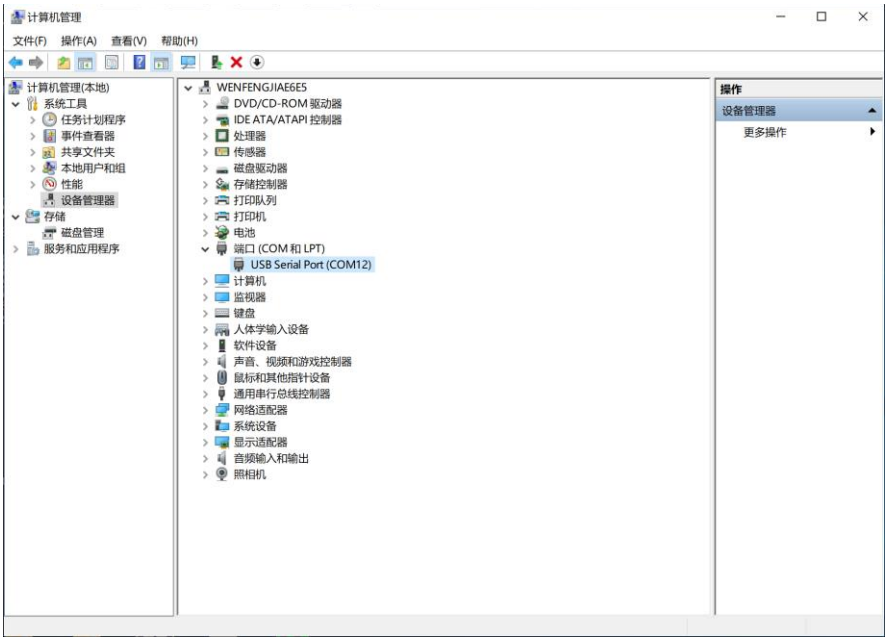
6.1 驱动安装

IMU61x 通过 TTL 转 USB 接口连接到 PC 机上，PC 机要安装对应的驱动，本产品使用 FTDI 公司的 FT232 产品，其驱动安装过程如下：

1. 打开 CDM21228_Setup.exe 文件，按照提示安装



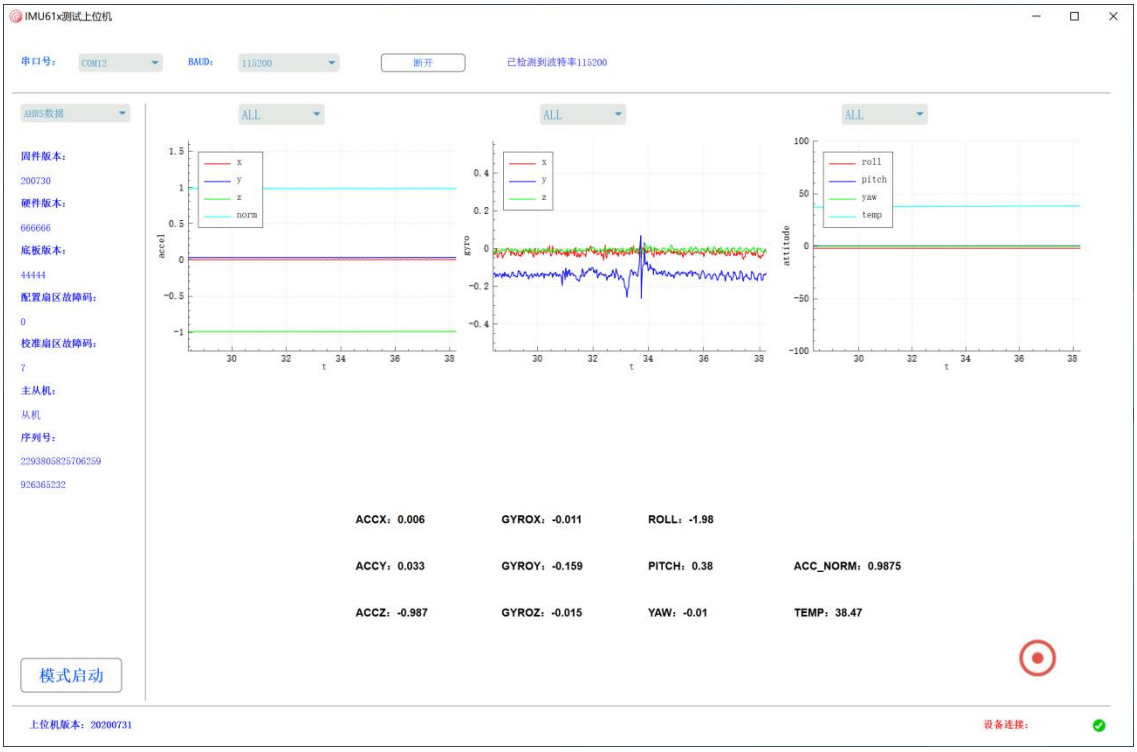
2. 安装后，可在设备管理器中看到对应端口号



6.2 测试上位机功能介绍

选择对应的端口号和波特率（波特率可以选AUTO自动识别），打开串口，上位机会自动获取版本号信息，左侧选项卡可以选择功能是：AHRS 数据，仪表盘，参数设置，命令生成器，安装误差校准等。

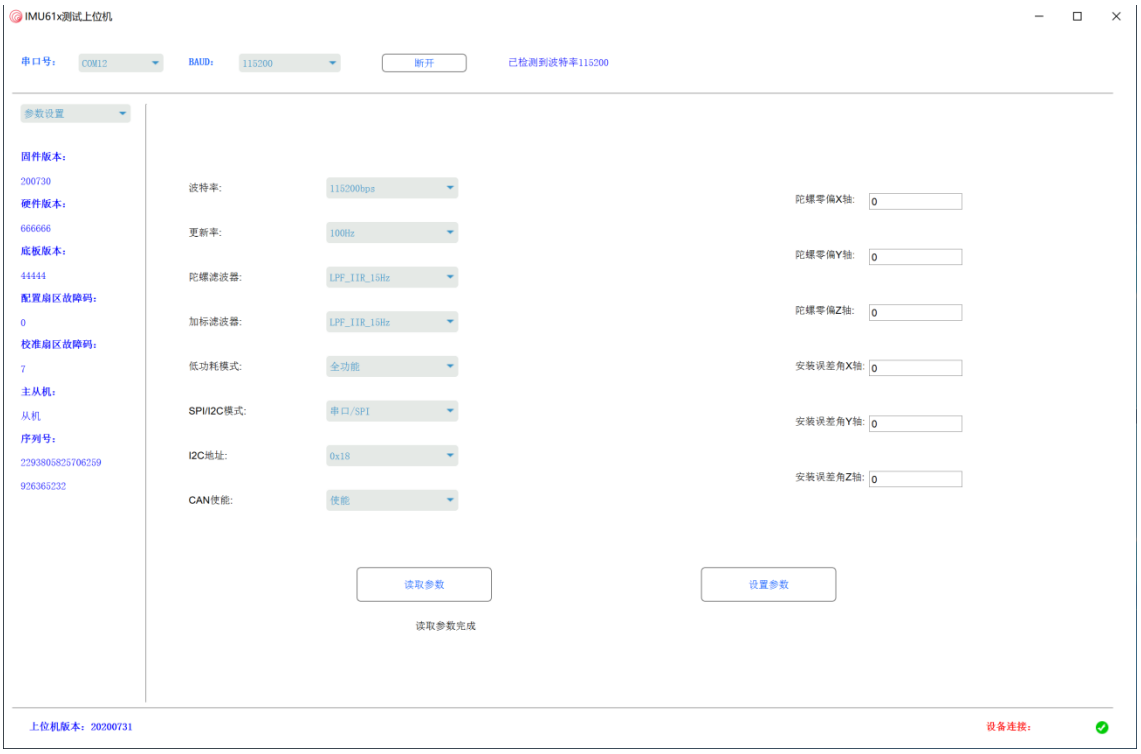
1. AHRS 数据主界面如下：



点击模式启动，IMU61x 发送 AHRS 数据流，界面显示曲线和数据。

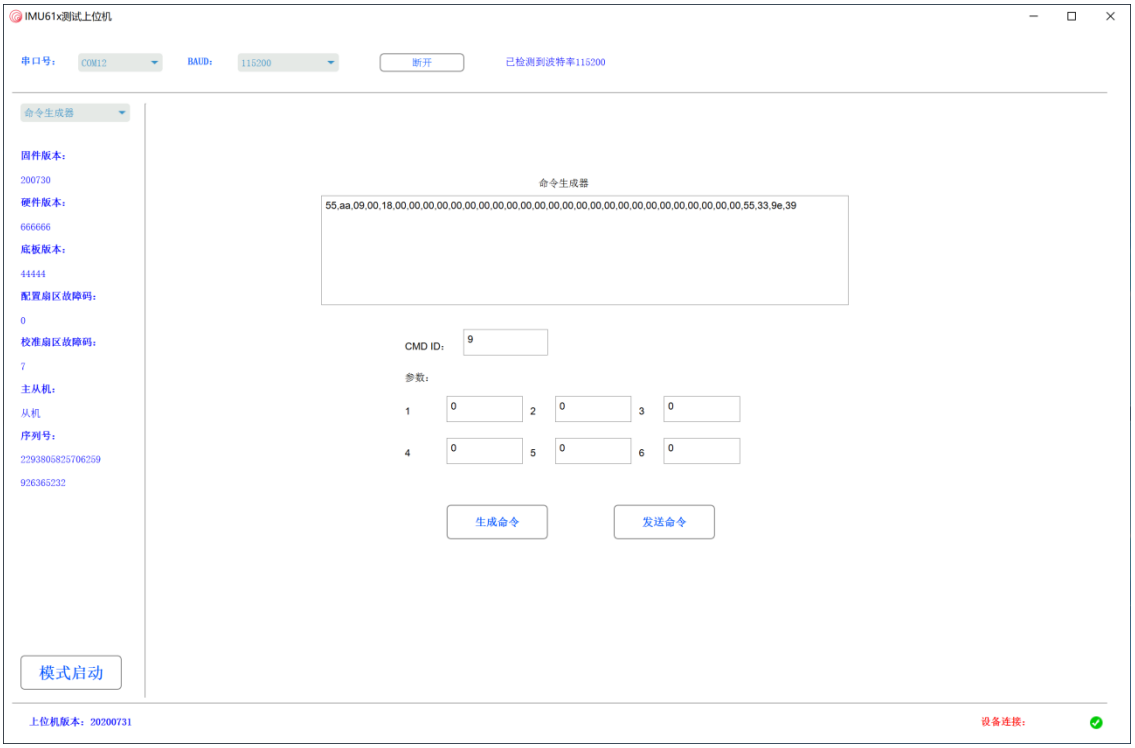
右侧图标可以选择保存和停止保存数据，数据路径在底部显示。

2. 参数设置界面如下：



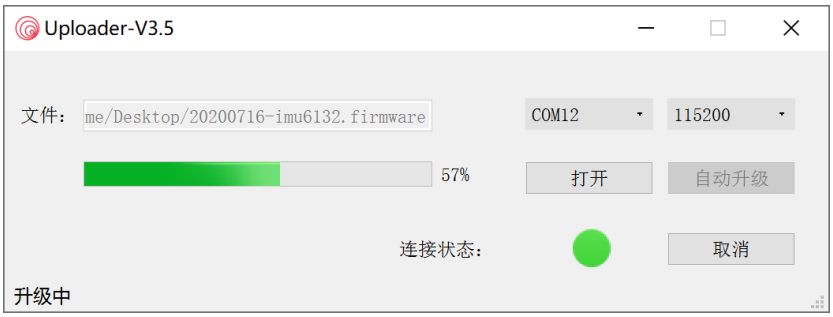
可选参数包括波特率，数据更新率，滤波器，低功耗模式，SPI/I2C 模式，I2C 地址，CAN 使能，陀螺零偏校准结果，安装误差角校准结果等。

3. 命令生成器界面如下：



参照手册的串口通信的命令模式 SET 指令部分，可以填入需要的 ID 和参数，生成完整的一条命令。数组可以复制并直接通过串口调试助手发送，也可以点击发送命令按钮直接发送。示意图上的命令是执行软件重启。

6.3 升级上位机功能介绍



升级上位机可以加载.firmware 格式固件，选择当前使用的波特率（AUTO 可以自动识别波特率），进行离线升级。

升级时，请注意右下角圆圈变成绿色表示当前有串口已连接，否则表示串口被占用或没有安装驱动。升级后，IMU 会自动重启，升级软件关闭串口连接。

请升级后用测试上位机确认新的固件版本号及功能是否正确。

7. CRC32 查表法计算

```
static const uint32_t crc32_tab[] = {
    0x00000000, 0x77073096, 0xee0e612c, 0x990951ba, 0x076dc419, 0x706af48f,
    0xe963a535, 0x9e6495a3, 0x0edb8832, 0x79dcb8a4, 0xe0d5e91e, 0x97d2d988,
    0x09b64c2b, 0x7eb17cbd, 0xe7b82d07, 0x90bf1d91, 0x1db71064, 0x6ab020f2,
    0xf3b97148, 0x84be41de, 0x1adad47d, 0x6ddde4eb, 0xf4d4b551, 0x83d385c7,
    0x136c9856, 0x646ba8c0, 0xfd62f97a, 0x8a65c9ec, 0x14015c4f, 0x63066cd9,
    0xfa0f3d63, 0x8d080df5, 0x3b6e20c8, 0x4c69105e, 0xd56041e4, 0xa2677172,
    0x3c03e4d1, 0x4b04d447, 0xd20d85fd, 0xa50ab56b, 0x35b5a8fa, 0x42b2986c,
    0xdbbbc9d6, 0xacbcf940, 0x32d86ce3, 0x45df5c75, 0xdcd60dcf, 0xabd13d59,
    0x26d930ac, 0x51de003a, 0xc8d75180, 0xbfd06116, 0x21b4f4b5, 0x56b3c423,
    0xcfba9599, 0xb8bda50f, 0x2802b89e, 0x5f058808, 0xc60cd9b2, 0xb10be924,
    0x2f6f7c87, 0x58684c11, 0xc1611dab, 0xb6662d3d, 0x76dc4190, 0x01db7106,
    0x98d220bc, 0xefd5102a, 0x71b18589, 0x06b6b51f, 0x9fbfe4a5, 0xe8b8d433,
    0x7807c9a2, 0x0f00f934, 0x9609a88e, 0xe10e9818, 0x7f6a0dbb, 0x086d3d2d,
    0x91646c97, 0xe6635c01, 0x6b6b51f4, 0x1c6c6162, 0x856530d8, 0xf262004e,
    0x6c0695ed, 0x1b01a57b, 0x8208f4c1, 0xf50fc457, 0x65b0d9c6, 0x12b7e950,
    0x8bbeb8ea, 0xfcb9887c, 0x62dd1ddf, 0x15da2d49, 0x8cd37cf3, 0xfbd44c65,
    0x4db26158, 0x3ab551ce, 0xa3bc0074, 0xd4bb30e2, 0x4adfa541, 0x3dd895d7,
    0xa4d1c46d, 0xd3d6f4fb, 0x4369e96a, 0x346ed9fc, 0xad678846, 0xda60b8d0,
    0x44042d73, 0x33031de5, 0xaa0a4c5f, 0xdd0d7cc9, 0x5005713c, 0x270241aa,
    0xbe0b1010, 0xc90c2086, 0x5768b525, 0x206f85b3, 0xb966d409, 0xce61e49f,
    0x5edef90e, 0x29d9c998, 0xb0d09822, 0xc7d7a8b4, 0x59b33d17, 0x2eb40d81,
    0xb7bd5c3b, 0xc0ba6cad, 0xedb88320, 0x9abfb3b6, 0x03b6e20c, 0x74b1d29a,
    0xead54739, 0x9dd277af, 0x04db2615, 0x73dc1683, 0xe3630b12, 0x94643b84,
    0x0d6d6a3e, 0x7a6a5aa8, 0xe40ecf0b, 0x9309ff9d, 0x0a00ae27, 0x7d079eb1,
    0xf00f9344, 0x8708a3d2, 0x1e01f268, 0x6906c2fe, 0xf762575d, 0x806567cb,
    0x196c3671, 0x6e6b06e7, 0xfed41b76, 0x89d32be0, 0x10da7a5a, 0x67dd4acc,
    0xf9b9df6f, 0x8ebeeff9, 0x17b7be43, 0x60b08ed5, 0xd6d6a3e8, 0xa1d1937e,
    0x38d8c2c4, 0x4fdff252, 0xd1bb67f1, 0xa6bc5767, 0x3fb506dd, 0x48b2364b,
    0xd80d2bda, 0xaf0a1b4c, 0x36034af6, 0x41047a60, 0xdf60efc3, 0xa867df55,
    0x316e8eef, 0x4669be79, 0xcb61b38c, 0xbc66831a, 0x256fd2a0, 0x5268e236,
    0xcc0c7795, 0xbb0b4703, 0x220216b9, 0x5505262f, 0xc5ba3bbe, 0xb2bd0b28,
    0x2bb45a92, 0x5cb36a04, 0xc2d7ffa7, 0xb5d0cf31, 0x2cd99e8b, 0x5bdeae1d,
    0x9b64c2b0, 0xec63f226, 0x756aa39c, 0x026d930a, 0x9c0906a9, 0xeb0e363f,
    0x72076785, 0x05005713, 0x95bf4a82, 0xe2b87a14, 0x7bb12bae, 0x0cb61b38,
    0x92d28e9b, 0xe5d5be0d, 0x7cdcefb7, 0x0bdbdf21, 0x86d3d2d4, 0xf1d4e242,
    0x68ddb3f8, 0x1fda836e, 0x81be16cd, 0xf6b9265b, 0x6fb077e1, 0x18b74777,
    0x88085ae6, 0xff0f6a70, 0x66063bca, 0x11010b5c, 0x8f659eff, 0xf862ae69,
    0x616bffd3, 0x166ccf45, 0xa00ae278, 0xd70dd2ee, 0x4e048354, 0x3903b3c2,
    0xa7672661, 0xd06016f7, 0x4969474d, 0x3e6e77db, 0xaed16a4a, 0xd9d65adc,
    0x40df0b66, 0x37d83bf0, 0xa9bcae53, 0xdebb9ec5, 0x47b2cf7f, 0x30b5ffe9,
    0xbdbdf21c, 0xcabac28a, 0x53b39330, 0x24b4a3a6, 0xbad03605, 0xcdd70693,
    0x54de5729, 0x23d967bf, 0xb3667a2e, 0xc4614ab8, 0x5d681b02, 0x2a6f2b94,
    0xb40bbe37, 0xc30c8ea1, 0x5a05df1b, 0x2d02ef8d
};

uint32_t crc_crc32(uint32_t crc, const uint8_t *buf, uint32_t size)
{
    for (uint32_t i=0; i<size; i++) {
        crc = crc32_tab[(crc ^ buf[i]) & 0xff] ^ (crc >> 8);
    }

    return crc;
}
```