NPU 系统通信协议

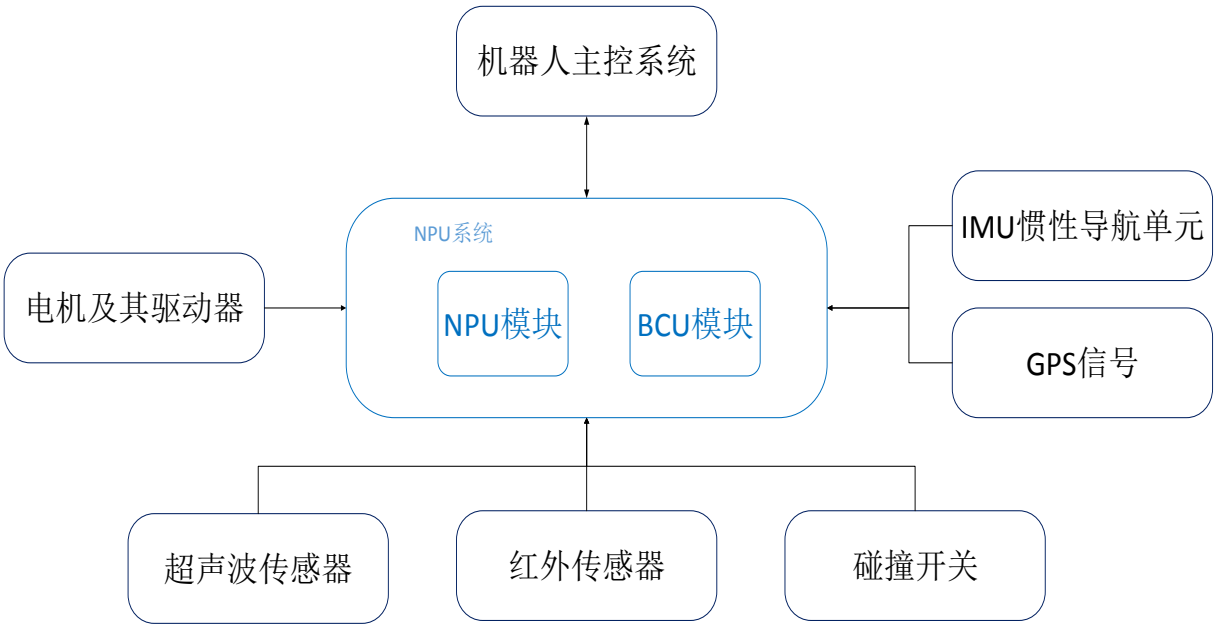
简介

NPU系统包含NPU核心导航模块和BCU底层控制模块，NPU系统可通过通信协议与底盘控制单元进行通信，实现方式为RS232接口，波特率为115200bps；该通信协议仅适用于NPU为master模式下，由用户开发底层控制板与NPU系统实现通信。

通过通信协议，NPU系统可获得底盘控制单元所采集到的速度、传感器等信息，同时底盘控制单元可获得NPU的运动控制指令及参数配置信息，以实现机器人的建图、导航功能。

内部模块框图和使用方法

下图描述了 NPU 系统和外部系统之间的通信连接框图。



通信协议命令

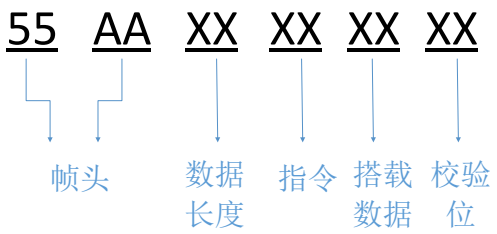
通信协议指令表

指令	描述
0x0A	获取版本信息
0xFB	设置控制参数
0x0B	读取控制参数
0xFC	设置 PID 参数
0x0C	读取 PID 参数
0xFD	设置传感器参数
0x0D	读取传感器参数
0xF0	设置电机使能
0xF8	设置电机刹车
0xF1	设置电机转速
0x01	读取电机转速
0xA1	设置并读取电机转速
0x02	读取电机编码器
0xA2	清零电机编码器
0x03	读取超声波传感器数据
0x04	读取红外传感器数据
0x05	读取碰撞开关数据

0x06	读取全部 IMU 数据
0xA6	读取 IMU-GYR 数据
0xB6	读取 IMU-ACC 数据
0xC6	读取 IMU-MAG 数据
0x07	读取 GPS 数据

通信协议结构

通信协议结构如下图所示：



其中，校验位数值为除该位外，其余所有字节相加求和后取最后一个字节为校验位。

参考：

55 AA 01 0A 0A

其中，55 AA 为帧头，01 为数据长度，0A 为指令，校验位为（55+AA+01+0A）求和后取最后一个字节。

通信协议请求命令

- 读取版本信息请求（GET_VER_ID）

NPU 系统初始化后需要读取版本信息确认无误后方可正常工作。

请求数据包				
帧头		数据长度	指令	校验位
0x55	0xAA	0xDataLen	0x0A	0xChkSum

应答数据包					
帧头		数据长度	指令	版本号	校验位
0x55	0xAA	0xDataLen	0x0A	0xVersionID	0xChkSum

版本号 VersionID=*(unit8)* (100*a+10*b+c)，转换为版本号“a.b”。

● 控制参数设置与读取

控制参数设置与读取包含机器人底盘电机数量，电机方向，电机最大转速，电机减速比，编码器脉冲数，电机 PWM 频率，电机制动方式以及电平有效位的参数设置与读取。

➢ 设置控制参数请求（SET_CTL_PARA）

请求数据包									
帧头		数据长度	指令	电机数量	电机方向	电机最大转速		电机减速比	
0x55	0xAA	0xDataLen	0xFB	0xNum	0xDir	0xD1	0xD2	0xD1	0xD2
编码器脉冲数		电机 PWM 频率		电机制动方式		电平有效位		校验位	
0xD1	0xD2	0xD1	0xD2	0xMotorBrkType		0xVol		0xChkSum	

应答数据包				
帧头		数据长度	指令	校验位
0x55	0xAA	0xDataLen	0xFB	0xChkSum

➢ 读取控制参数请求（GET_CTL_PARA）

请求数据包				
帧头		数据长度	指令	校验位
0x55	0xAA	0xDataLen	0x0B	0xChkSum

应答数据包									
帧头		数据长度	指令	电机数量	电机方向	电机最大转速		电机减速比	
0x55	0xAA	0xDataLen	0x0B	0xNum	0xDir	0xD1	0xD2	0xD1	0xD2
编码器脉冲数		电机 PWM 频率		电机制动方式		电平有效位		校验位	
0xD1	0xD2	0xD1	0xD2	0xMotorBrkType		0xVol		0xChkSum	

电机数量 MotorNum=*(unit8)*, max=4;

电机方向 MotorDir=0b(bDir1_bDir2_bDir3_bDir4_0_0_0_0);

电机最大转速 MotorMaxSpdRpm = (uint16)0x(D1_D2)[RPM], max=65536；

电机减速比 MotorRdcRatio = (uint16)0x(D1_D2), prcs=0.01, max=655.36；

编码器脉冲数 MotorEncResPpr = (uint16)0x(D1_D2)[PPR], max=65536；

电机 PWM 频率 MotorPwmFrqHz = (uint16)0x(D1_D2)[hz], max=65536；

电机制动方式 MotorBrkType(1=HARD_BRK,0=SOFT_BRK)；

电平有效位 Vol=0b(bEnbVol_bDirVoll_bPwmVol_bBrkVol_0_0_0_0)；

电机方向转换为八位二进制后，该位为 1 则代表相应的电机方向为正方向，该位为 0 则代表相应的电机方向为负方向；

电平有效位转换为八位二进制后，该位为 1 则代表相应的管脚高电平有效，该位为 0 则代表相应的管脚低电平有效。

● PID 参数设置与读取

PID 参数设置与读取包含 Kp+,Kp-,Ki+,Ki-,Kd+,Kd-的参数设置与读取。

➢ 设置 PID 参数请求（SET_PID_PARA）

请求数据包									
帧头		数据长度	指令	PID 使能位	KpAcc		KpDec		
0x55	0xAA	0xDataLen	0xFC	0xEnbPID	0xD1	0xD2	0xD1	0xD2	
KiAcc		KiDec		KdAcc		KdDec		校验位	
0xD1	0xD2	0xD1	0xD2	0xD1	0xD2	0xD1	0xD2	0xChkSum	

应答数据包				
帧头		数据长度	指令	校验位
0x55	0xAA	0xDataLen	0xFC	0xChkSum

➢ 读取 PID 参数请求（GET_PID_PARA）

请求数据包				
帧头		数据长度	指令	校验位
0x55	0xAA	0xDataLen	0x0C	0xChkSum

应答数据包								
帧头		数据长度	指令	PID 使能位	KpAcc		KpDec	
0x55	0xAA	0xDataLen	0x0C	0xEnbPID	0xD1	0xD2	0xD1	0xD2
KiAcc		KiDec		KdAcc		KdDec		校验位
0xD1	0xD2	0xD1	0xD2	0xD1	0xD2	0xD1	0xD2	0xChkSum

PID 使能开关 EnbPID=(1:On, 0:OFF) ;

Kp+数据 KpAcc = (uint16)0x(D1_D2)*0.1 ;

Kp-数据 KpDec = (uint16)0x(D1_D2)*0.1 ;

Ki+数据 KiAcc = (uint16)0x(D1_D2)*0.1 ;

Ki-数据 KiDec = (uint16)0x(D1_D2)*0.1 ;

Kd+数据 KdAcc = (uint16)0x(D1_D2)*0.1 ;

Kd-数据 KdDec = (uint16)0x(D1_D2)*0.1

● 传感器参数设置与读取

传感器参数设置与读取包含超声波传感器的数量，红外传感器的数量，碰撞开关的数量的设置与读取。

➢ 设置传感器参数请求（SET_SEN_PARA）

请求数据包								
帧头		数据长度	指令	使能位	超声波数量	红外数量	碰撞开关数量	校验位
0x55	0xAA	0xDataLen	0xFD	0xEnb	0xSnrNum	0xIrdNum	0xBprNum	0xChkSum

应答数据包				
帧头		数据长度	指令	校验位
0x55	0xAA	0xDataLen	0xFD	0xChkSum

➢ 读取传感器参数请求（GET_SEN_PARA）

请求数据包				
帧头		数据长度	指令	校验位
0x55	0xAA	0xDataLen	0x0D	0xChkSum

应答数据包								
帧头		数据长度	指令	使能位	超声波数量	红外数量	碰撞开关数量	校验位
0x55	0xAA	0xDataLen	0x0D	0xEnb	0xSnrNum	0xIrdNum	0xBprNum	0xChkSum

传感器使能位 Enb=0b(bEnbSnr_bEnbIrd_bEnbBmp_bEnbGyr_bEnbAcc_bEnbMag_bEnbGps_0)；

使能位转换为八位二进制后，该位为 1 则代表对应的传感器使能有效，该位为 0 则代表对应的传感器使能无效；

超声波传感器数量 SonarNum = *(uint8)*, max=8；

红外传感器数量 IrdNum = *(uint8)*, max=8；

碰撞开关数量 BumperNum = *(uint8)*, max=8。

● 电机使能及刹车设置

电机使能及刹车设置主要包含电机数量，电机使能开关以及电机刹车开关的设置。

➢ 设置电机使能（SET_MTR_ENB）

请求数据包						
帧头		数据长度	指令	电机数量	电机使能	校验位
0x55	0xAA	0xDataLen	0xF0	0xMotorNum	0xEnb	0xChkSum

电机数量 MotorNum=*(unit8)*, max=4;

电机使能开关 Enb (1:ON, 0:OFF)

应答数据包				
帧头		数据长度	指令	校验位
0x55	0xAA	0xDataLen	0xF0	0xChkSum

➢ 设置电机刹车（SET_MTR_BRK）

请求数据包						
帧头		数据长度	指令	电机数量	电机使能	校验位
0x55	0xAA	0xDataLen	0xF8	0xMotorNum	0xBrk	0xChkSum

电机数量 MotorNum=*(unit8)*, max=4;

电机使能开关 Brk (1:ON, 0:OFF)

应答数据包				
帧头		数据长度	指令	校验位
0x55	0xAA	0xDataLen	0xF8	0xChkSum

● 电机转速设置与读取

电机转速设置与读取包含电机数量和各电机转速的设置及读取。

➢ 设置电机转速（SET_MTR_SPD）

请求数据包									
帧头		数据长度	指令	电机数量	电机 1 转速				
0x55	0xAA	0xDataLen	0xF1	0xMotorNum	0xD1	0xD2	0xD3		
电机 2 转速			电机 3 转速			电机 4 转速			校验位
0xD1	0xD2	0xD3	0xD1	0xD2	0xD3	0xD1	0xD2	0xD3	0xChkSum

应答数据包				
帧头		数据长度	指令	校验位
0x55	0xAA	0xDataLen	0xF1	0xChkSum

➢ 读取电机转速（GET_MTR_SPD）

请求数据包			
帧头	数据长度	指令	校验位

0x55	0xAA	0xDataLen	0x01	0xChkSum

应答数据包									
帧头		数据长度	指令	电机数量	电机 1 转速				
0x55	0xAA	0xDataLen	0x01	0xMotorNum	0xD1	0xD2	0xD3		
电机 2 转速			电机 3 转速			电机 4 转速			校验位
0xD1	0xD2	0xD3	0xD1	0xD2	0xD3	0xD1	0xD2	0xD3	0xChkSum

➢ 设置并读取电机转速（SWP_MTR_SPD）

请求数据包									
帧头		数据长度	指令	电机数量	电机 1 转速				
0x55	0xAA	0xDataLen	0xA1	0xMotorNum	0xD1	0xD2	0xD3		
电机 2 转速			电机 3 转速			电机 4 转速			校验位
0xD1	0xD2	0xD3	0xD1	0xD2	0xD3	0xD1	0xD2	0xD3	0xChkSum

应答数据包									
帧头		数据长度	指令	电机数量	电机 1 转速				
0x55	0xAA	0xDataLen	0xA1	0xMotorNum	0xD1	0xD2	0xD3		
电机 2 转速			电机 3 转速			电机 4 转速			校验位
0xD1	0xD2	0xD3	0xD1	0xD2	0xD3	0xD1	0xD2	0xD3	0xChkSum

电机数量 MotorNum=*(unit8)*, max=4;

电机 1 转速 Rpm1=(uint24)0x(D1_D2_D3)[0.01RPM]. MAX=1,677,72.15[RPM]；

电机 2 转速 Rpm2=(uint24)0x(D1_D2_D3)[0.01RPM]. MAX=1,677,72.15[RPM]；

电机 3 转速 Rpm3=(uint24)0x(D1_D2_D3)[0.01RPM]. MAX=1,677,72.15[RPM]；

电机 4 转速 Rpm4=(uint24)0x(D1_D2_D3)[0.01RPM]. MAX=1,677,72.15[RPM]。

● 电机编码器读取及清零

电机编码器读取及清零包含电机编码器脉冲数的读取及脉冲数清零操作。

➢ 电机编码器读取（GET_MTR_ENC）

请求数据包				
帧头		数据长度	指令	校验位
0x55	0xAA	0xDataLen	0x02	0xChkSum

应答数据包												
帧头		数据长度	指令	电机数量	编码器 1 脉冲数							
0x55	0xAA	0xDataLen	0x02	0xMotorNum	0xD1	0xD2	0xD3	0xD4				
编码器 2 脉冲数				编码器 3 脉冲数				编码器 4 脉冲数				校验位
0xD1	0xD2	0xD3	0xD4	0xD1	0xD2	0xD3	0xD4	0xD1	0xD2	0xD3	0xD4	0xChkSum

电机数量 MotorNum=*(unit8)*, max=4;

编码器 1 脉冲数 Enc1=(uint32)0x(D1_D2_D3_D4)[TICK], MAX=2,147,483,647[TICK]；

编码器 2 脉冲数 Enc2=(uint32)0x(D1_D2_D3_D4)[TICK], MAX=2,147,483,647[TICK]；

编码器 3 脉冲数 Enc3=(uint32)0x(D1_D2_D3_D4)[TICK], MAX=2,147,483,647[TICK]；

编码器 4 脉冲数 Enc4=(uint32)0x(D1_D2_D3_D4)[TICK], MAX=2,147,483,647[TICK] 。

➢ 电机编码器清零（CLR_MTR_ENC）

请求数据包				
帧头		数据长度	指令	校验位
0x55	0xAA	0xDataLen	0xA2	0xChkSum

应答数据包				
帧头		数据长度	指令	校验位
0x55	0xAA	0xDataLen	0xA2	0xChkSum

● 传感器数据读取

传感器数据读取包含超声波传感器数据，红外传感器数据以及碰撞开关数据的数据。

➢ 超声波传感器数据读取（GET_SNR_DATA）

请求数据包				
帧头		数据长度	指令	校验位
0x55	0xAA	0xDataLen	0x03	0xChkSum

应答数据包

帧头		数据长度	指令	超声波数量	超声波 1		超声波 2	
0x55	0xAA	0xDataLen	0x03	0xSnrNum	0xH	0xL	0xH	0xL

超声波 3		超声波 4		超声波 5		超声波 6		超声波 7		超声波 8		校验位
0xH	0xL	0xH	0xL	0xH	0xL	0xH	0xL	0xH	0xL	0xH	0xL	0xChkSum

超声波数量 SnrNum= *(unit8)* , max=8;

超声波数据为 16 进制字节（Data = *(uint16)* [cm]）

➢ 红外传感器数据读取（GET_IRD_DATA）

请求数据包				
帧头		数据长度	指令	校验位
0x55	0xAA	0xDataLen	0x04	0xChkSum

应答数据包								
帧头		数据长度	指令	红外数量	红外 1 数据		红外 2 数据	
0x55	0xAA	0xDataLen	0x04	0xIrdNum	0xH	0xL	0xH	0xL

红外 3 数据		红外 4 数据		红外 5 数据		红外 6 数据		红外 7 数据		红外 8 数据		校验位
0xH	0xL	0xH	0xL	0xH	0xL	0xH	0xL	0xH	0xL	0xH	0xL	0xChkSum

红外传感器数量 IrdNum= *(unit8)* , max=8;

红外传感器数据为 16 进制字节（Data = *(uint16)* [cm]）

➢ 碰撞开关数据读取（GET_BMP_DATA）

请求数据包				
帧头		数据长度	指令	校验位
0x55	0xAA	0xDataLen	0x05	0xChkSum

应答数据包						
帧头		数据长度	指令	碰撞开关数量	碰撞开关数据	校验位
0x55	0xAA	0xDataLen	0x05	0xBmpNum	0xBmpData	0xChkSum

碰撞开关数量 BmpNum=*（unit8）*, max=8;

碰撞开关数据 BmpData= 0b(b1_b2_b3_b4_b5_b6_b7_b8)

● IMU 数据读取

IMU 数据读取包含 IMU 全部数据读取 ,IMU-GYR 数据读取 ,IMU-ACC 数据读取以及 IMU-MAG 数据读取。

➢ IMU 全部数据读取（GET_IMU_ALL_DATA）

请求数据包				
帧头		数据长度	指令	校验位
0x55	0xAA	0xDataLen	0x06	0xChkSum

应答数据包

帧头		数据长度	指令	Gyr_X		Gyr_Y		Gyr_Z	
0x55	0xAA	0xDataLen	0x06	0xD1	0xD2	0xD1	0xD2	0xD1	0xD2

Acc_X		Acc_Y		Acc_Z		Mag_X		Mag_Y		Mag_Z		校验位
0xD1	0xD2	0xD1	0xD2	0xD1	0xD2	0xD1	0xD2	0xD1	0xD2	0xD1	0xD2	0xChkSum

➢ IMU-GYR 数据读取（GET_IMU_GYR_DATA）

请求数据包

帧头		数据长度	指令	校验位
0x55	0xAA	0xDataLen	0xA6	0xChkSum

应答数据包

帧头		数据长度	指令	Gyr_X		Gyr_Y		Gyr_Z		校验位
0x55	0xAA	0xDataLen	0xA6	0xD1	0xD2	0xD1	0xD2	0xD1	0xD2	0xChkSum

➢ IMU-ACC 数据读取（GET_IMU_ACC_DATA）

请求数据包

帧头		数据长度	指令	校验位
0x55	0xAA	0xDataLen	0xB6	0xChkSum

应答数据包

帧头		数据长度	指令	Acc_X		Acc_Y		Acc_Z		校验位
0x55	0xAA	0xDataLen	0xB6	0xD1	0xD2	0xD1	0xD2	0xD1	0xD2	0xChkSum

➢ IMU-MAG 数据读取（GET_IMU_MAG_DATA）

请求数据包				
帧头		数据长度	指令	校验位
0x55	0xAA	0xDataLen	0xC6	0xChkSum

应答数据包										
帧头		数据长度	指令	Mag_X		Mag_Y		Mag_Z		校验位
0x55	0xAA	0xDataLen	0xC6	0xD1	0xD2	0xD1	0xD2	0xD1	0xD2	0xChkSum

Gyr_X = (uint16)0x(D1_D2) ;

Gyr_Y = (uint16)0x(D1_D2) ;

Gyr_Z = (uint16)0x(D1_D2) ;

Acc_X = (uint16)0x(D1_D2) ;

Acc_Y = (uint16)0x(D1_D2) ;

Acc_Z = (uint16)0x(D1_D2) ;

Mag_X = (uint16)0x(D1_D2) ;

Mag_Y = (uint16)0x(D1_D2) ;

Mag_Z = (uint16)0x(D1_D2) ;

● GPS 数据读取（GET_GPS_DATA）

GPS 数据的读取包含 GPS 信号来源的经度，纬度以及海拔数据的读取。

请求数据包				
帧头		数据长度	指令	校验位
0x55	0xAA	0xDataLen	0x07	0xChkSum

应答数据包								
帧头		数据长度	指令	经度				
0x55	0xAA	0xDataLen	0x07	0xD1	0xD2	0xD3	0xD4	
纬度				海拔				校验位
0xD1	0xD2	0xD3	0xD4	0xD1	0xD2	0xD3	0xD4	0xChkSum

纬度 Latitude=(uint32)0x(D1_D2_D3_D4)；

经度 Longitude=(uint32)0x(D1_D2_D3_D4)；

海拔 Alitude=(uint32)0x(D1_D2_D3_D4)。