高新兴机器人运动控制接口协议简述

文件状态： [√] 草稿 [ ] 正式发布 [ ] 正在修改	报告编号	20190306		
	当前版本	V1.0.7		
	编写人	李柏文	编写日期	2019 - 3 - 6
	审批人		审批日期	
	保密级别			

版本	日期	操作人	变更描述
V1.0.0	2017-12-25	莫存相	文档新建
V1.0.1	2018-01-01	莫存相	修改反馈数据为
V1.0.2	2018-01-05	莫存相	添加超声波测距数据
V1.0.3	2018-01-20	莫存相	分离各个内容，采用查询应答的方式
V1.0.4	2018-8-3	李柏文	-新增电池状态信息反馈 -对排版做了一些调整和美化
V1.0.5	2018-12-20	李柏文	增加了激光数据查询
V1.0.6	2018-12-22	李柏文	修正了2.2.10项的激光数据格式
V1.0.7	2019-03-06	李柏文	删除了充电部分和激光数据部分
V1.0.8	2019-12-24	李柏文	增加了IMU查询、GPS数据上报、电量信息上报

目录

一、网络参数

工控机与主控板之间通过 UDP 进行通信，网络配置为：

Gateway_IP:	10.7.5.1
Sub_Mask:	255.255.255.0
IP(主板): 端口: 端口:	10.7.5.224 4002 4001
IP(工控机): 端口: 端口:	10.7.5.220 4002 4001（主控发送到此端口的数据会被工控通过 TCP 转发到导航板）

工控机与导航板之间通过 TCP 进行通信，导航板作为客户端，如果导航板发送本文档中的命令给工控，工控会对其进行转发，网络配置：

Gateway_IP:	10.7.5.1
Sub_Mask:	255.255.255.0
IP(工控机): 端口:	10.7.5.220 4321
IP(导航板):	10.7.5.88

符号说明：

MCU(S0)表示 socket0 端口 4002

MCU(S1)表示 socket1 端口 4001

注：通信控制频率控制在 20HZ 至 50HZ 之间

二、帧结构

帧基本结构由两部分组成：帧头+数据(可选)

帧头(bytes)	数据(N bytes)
TNBHead	xxxx

2.1、帧头结构：

```
struct _TNBHead
{
    u16 u16MagicCode;       /* "GS" */
    u16 u16Size;            /* 总字节数，包括帧头与数据 */
    u16 u16CmdId;           /* 指令 ID */
    u16 u16CRC;             /* CRC 校验 */
};
```

2.1.1、成员说明：

u16 MagicCode	“GS”
u16 Size	帧头+数据总字节数
u16 CmdId	指令：例如 0x7426 代表运动控制帧
u16 CRC	帧数据的 CRC16 校验.见附 1

2.2、数据

数据部分并非必须，数据根据实际应用，定义其代表的意义。

三、导航与底盘交互帧

3.1 底盘导航间运动交互指令：

CMD		CMD_ID	数据结构	反馈 ID	反馈结构	备注
控制类	查询类					
运动控制		0x7426	Move_Ctrl	0x7426	Move_ST	
	里程、角度查询	0x7600	Dist_Get	0x7600	Dist_Rsp	
	超声数据查询	0x7601	Ultra_Get	0x7601	Ultra_Rsp	
	防撞条状态查询	0x7602	AnticollisionBar_Get	0x7602	AnticollisionBar_Rsp	
超声紧急制动开关控制		0x7603	UltraBreak_Set	0x7603	UltraBreak_Rsp	用于控制当超声近距障碍时本体是否主动停止。
防撞条紧急制动开关控制		0x7604	Anticollision_Set	0x7604	Anticollision_Rsp	
	驱动器异常反馈	0x7605	MotorDrvSt_Get	0x7605	MotorDrvSt_Rsp	
	通信异常反馈			0x7000	Comm_Err	
	激光数据	0x8600	LaserData_Get	0x8600	LaserData_Rsp	

注：通信异常无需查询，它可能是任何命令的反馈。

3.2、指令具体结构：

3.2.1 运动控制

```
/* PC->MCU(S0,S1): 运动控制命令 */
struct Move_Ctrl
{
    TNBHead head;
    u8 Cmd;              /* 运动控制命令 */
    u16 Spd_l;           /* 本体线速度 */
    u16 Spd_a;           /* 本体角速度 */
};
```

控制指令反馈：

```
/* MCU(s0,s1)->PC:  运动速度上报 */
struct Move_ST
{
    TNBHead head;
    u16 Spd_l;           /* 本体当前线速度 */
    u16 Spd_a;           /* 本体当前角速度 */
};
```

成员说明：

head	见 1.1 帧头结构
Cmd	见运动控制指令枚举
Spd_l	Bit15：符号位，1 代表负，0 代表正 Bit14~bit0：本体线速度的绝对值大小 单位：mm/s 本体向前运动为正，向后运动为负
Spd_a	Bit15：符号位，1 代表负，0 代表正 Bit14~bit0：本体角速度的绝对值大小 单位：0.1°/s 本体朝向逆时针偏转为正，顺时针为负。

运动控制指令枚举：

```
typedef enum
{
    BD_STOP = 0,      //停止，此命停止过程会递减速度最终停止下来，有一定缓冲。
    BD_RUN,           //运动，表示运动，方向和速度由 Spd_l，Spd_a 决定
    BD_BREAK,         //刹车，与 BS_STOP 相比，没有任何缓冲，直接停止。
}BD_CMD;
```

注：关于停止，有三种方式：

1，BD_STOP：　　此方式下，忽略速度参数，带缓冲滑行尽快的停止下来；

2，BD_RUN：　　此方式下，若速度参数给 0，由速度环将速度调节到 0；
3，BREAK：　　此方式，将忽略参数，忽略速度环调节，强行将速度直接置 0；

3.2.2 里程、角度查询

查询命令

```
/* PC->MCU(s1)：  位置，角度查询 */
struct Dist_Get
{
    TNBHead head;
};
```

反馈：

```
/* MCU(S1)->PC：  位置，角度反馈 */
struct Dist_Rsp
{
    TNBHead   head;
    s32       speedI;   /*  本体总运动里程 单位 mm  向前为正，向后为负  */
    s32       angleI;   /*本地总转动角度，单位 0.1 度，逆时针为正，顺时针为负
                        逆时针 0~1800  顺时针 0~ -1800*/
};
```

注:角度和里程都是从机器人上电开始为基准统计，关机后，下次开机将重新统计。

3.2.3 超声数据查询

查询命令

```
/* PC->MCU(s1)：  获取超声波数据 */
struct Ultra_Get
{
    TNBHead head;
};
```

反馈：

```
/* MCU(S1)->PC：  反馈超声波数据 */
struct Ultra_Rsp
{
    TNBHead head;
    /*8 个超声波数据，单位 mm，0—7 分别是：左，左前，前，右前，右，右后，后，左后。
    当数据时 0XFFFF 时，表示对应的超声波异常了 */
    u16  ultra_data[8];
};
```

3.2.4 防撞条状态查询

查询命令

```
/* PC->MCU(s1)：  获取超声波数据 */
struct AnticollisionBar_Get
{
    TNBHead head;
};
```

反馈：

```
/* MCU(S1)->PC：  反馈超声波数据 */
struct AnticollisionBar_Rsp
{
    TNBHead   head;
    U8        status;          /*防撞条状态 0：未撞击  1 撞击*/
};
```

3.2.5 超声紧急制动开关控制

设置命令

```
/* PC->MCU(s1)：  超声近距底层急停设置 */
struct UltraBreak_Set
{
    TNBHead   head;
    U8        cmd;             /*0-关闭，1-开启*/
};
```

反馈：

```
/* MCU(S1)->PC：  对设置命令的反馈 */
struct UltraBreak _Rsp
{
    TNBHead   head;
    U8        status;          /* 超声紧急自动制动功能开关状态，0-关闭，1-开启 */
};
```

3.2.6 防撞条紧急制动开关控制

设置命令

```
/* PC->MCU(s1)：  防撞条底层急停设置 */
struct Anticollision_Set
{
```

```
    TNBHead    head;
    U8         cmd;            /* 0-关闭，1-开启 */
};
```

反馈：

```
/* MCU(S1)->PC：  对设置命令的反馈 */
struct Anticollision_Rsp
{
    TNBHead    head;
    U8         status;         /* 防撞条紧急自动制动功能开关状态  0-关闭，1-开启 */
};
```

3.2.7 驱动器异常状态查询

查询命令

```
/* PC->MCU(s1)：  查询驱动器异常状态 */
struct MotorDrvSt_Get
{
    TNBHead    head;
    U8         Driver;         /* 0-左侧电机驱动器，1-右侧电机驱动器 */
};
```

反馈：

```
/* MCU(S1)->PC：  反馈驱动器异常状态(异常时主动上报) */
struct MotorDrvSt _Rsp
{
    TNBHead    head;
    U8         Driver;         /* 0-左侧电机驱动器，1-右侧电机驱动器 */
    U32        err;            /* 驱动器异常状态值 */
};
```

驱动器异常状态值表

Bit 位	异常内容
0	过流(对驱动器本身而言)
1	过压
2	编码器异常
3	欠压
4	过载(对电机而言)
5~31	预留

3.2.8 通信异常报告

```
/* PC->MCU(s0)：通信异常信息，工控告知 MCU，与导航失去通信连接；
   PC->NAV：    工控通知导航，与 MCU 失去通信连接；*/
```

```
struct Comm_Err
{
    TNBHead   head;
    U16       Err;
};
```

通信异常错误表：

错误码	内容
01	发送超时
02	目的设备未连接

3.2.9 IMU 数据查询

查询命令

```
/* PC->MCU(s1)：  查询 IMU 数据 */
struct _IMU_Get
{
    TNBHead head;
}__PACKED__;

typedef struct _IMU_Get IMU_Get;
```

反馈：

```
/* MCU(s1)->PC：  反馈 IMU 数据 */
struct _IMU_Rsp
{
      int16_t GyroX;      //单位：0.1° ，范围±2000° /s
      int16_t GyroY;
      int16_t GyroZ;
      int16_t AccelX;     //单位：0.000244，范围±8G
      int16_t AccelY;
      int16_t AccelZ;
      int16_t CompassX;  //单位：mT，范围±4800uT
      int16_t CompassY;
      int16_t CompassZ;
      int16_t Pitch;      //单位 0.1°
      int16_t Roll;
      int16_t Yaw;        //单位 0.1 度，逆时针为正(0~1800)，顺时针为负(0~-1800)
}__PACKED__;

typedef struct _IMU_Rsp IMU_Rsp;
```

四、工控与底盘交互帧

CMD		CMD_ID	数据结构	反馈 ID	反馈结构	备注
控制类	查询类					
		0x7303	TNBRobot_GPS_PushOn			主动上报的
		0x7425	Power_Manager			

4.1 GPS 信息上报

命令名称 GPS_PushOn 命令码 0x7303

```
/* MCU(s0)->PC: GPS 信息上报(主动上报)
*/
struct _TNBRobotTime
{
  unsigned short   Year;    // 1970 ~ 2100
  unsigned char   Month;   // 1 ~ 12
  unsigned char   Day;     // 1 ~ 31
  unsigned char   Hour;    // 0 ~ 23
  unsigned char   Minute;  // 0 ~ 59
  unsigned char   Second;  // 0 ~ 59
}__PACKED__;
typedef struct _TNBRobotTime TNBRobotTime;

struct _TNBRobot_GPS_PushOn
{
  TNBHead      head;
  unsigned char     byValid;      // 定位标志:0 无效,1 有效
  unsigned char     byEW;     // 1 西经，2 东经
  unsigned char    bySN;     // 1 北纬，2 南纬
  unsigned int      unLongitude;  // 经度 单位度，实际数据*1000000
  unsigned int      unLatitude;  // 纬度 单位度，实际数据*1000000
  int         nASL;    // 海拔  单位米，实际数据*100
  unsigned int      unSpeed;      // 相对位移速度 单位 km/h，实际数据*1000
  unsigned int      unDirection;  // 相对位移方向  单位度，实际数据*100
  TNBRobotTime       sTime;     // 时间
}__PACKED__;
typedef struct _TNBRobot_GPS_PushOn TNBRobot_GPS_PushOn;
```

4.2 电源信息上报

命令名称 PowerManager_CMD 命令码 0x7425

```
/* MCU(s0)->PC：电源管理信息上报（主动上报）*/
```

此结构上报与原来相同，但有效通道数量少了。

```
struct _STATE_{
  u16 STA_OUT_EN:1;     //放电使能
  u16 STA_CHA_EN:1;     //充电使能
  u16 STA_OUT:1;      //放电状态
  u16 STA_CHA:1;       //充电状态
  u16 STA_ADV_CHA:1;     //预充电状态
  u16 STA_CHAER_CONECT:1;   //适配器连接
  u16 STA_SWITCH:1;    //开关状态
  u16 bit7:1;         //预留
  u16 bit8:1;         //预留
  u16 bit9:1;        //预留
  u16 bit10:1;       //预留
  u16 bit11:1;       //预留
  u16 bit12:1;       //预留
  u16 bit13:1;       //预留
  u16 bit14:1;        //预留
  u16 bit15:1;       //预留
}__PACKED__;
struct _PROTECT_{
  u16 PRO_CHA_CUR:1;     //充电过流保护
  u16 PRO_OUT_CUA:1;     //放电过流保护
  u16  PRO_CHA_TEM:1;    //充电温度保护
  u16  PRO_OUT_TEM:1;    //放电温度保护
  u16 PRO_SHORT:1;      //短路保护
  u16  PRO_VOT_LOW:1;    //欠压保护
  u16  PRO_VOT_HIGH:1;    //过压保护
  u16 bit7:1;         //预留
  u16 bit8:1;         //预留
  u16 bit9:1;         //预留
  u16 bit10:1;       //预留
  u16 bit11:1;       //预留
  u16 bit12:1;        //预留
  u16 bit13:1;       //预留
  u16 bit14:1;       //预留
  u16 bit15:1;       //预留
}__PACKED__;
struct _ERROR_{
  u16 ERR_READ:1;      //读故障
```

```
  u16 ERR_CAP_UPDATE:1;  //容量更新故障
  u16 ERR_CELL_VOT:1;    //电芯电压故障
  u16 ERR_CHA_MOS:1;     //充电 MOS 故障
  u16 ERR_OUT_MOS:1;     //放电 MOS 故障
  u16 ERR_ADV_CHA:1;     //预充电故障
  u16 ERR_IC:1;       //ic 故障
  u16 bit7:1;         //预留
  u16 bit8:1;         //预留
  u16 bit9:1;         //预留
  u16 bit10:1;        //预留
  u16 bit11:1;        //预留
  u16 bit12:1;         //预留
  u16 bit13:1;        //预留
  u16 bit14:1;        //预留
  u16 bit15:1;         //预留
}__PACKED__;
typedef  struct{
  u16 Ver;          //版本号
  u16 Time;        //修订日期
  u16  Vot;          //实时电压(10mV)
  s16  Cur;          //实时电流(10mA)
  u8  SOC;          //电量百分比
  s8  Temp;        //温度(摄氏度)
  u16  CapNow;        //实时容量
  u16  CapFull;         //满电容量
  u16 ChgNum;         //循环次数
  u16  CellVotH;       //电芯最高电压
  u16 CellVotL;       //电芯最低电压
  s8  CellTempH;       //电芯最高温度
  s8  CellTempL;       //电芯最低温度
  struct _STATE_  State;  //状态标志
  struct _PROTECT_ Protect;  //保护信息
  struct _ERROR_  Error;  //故障信息
}__PACKED__ stdbat_info;
//
struct
{
    u32 RuningStatus:     1;           //运行状态  1 表示停止运行准备关机，0 表示正常开机运行 Bit0
    u32 flagChargeStatue: 1;           //充电状态              Bit1
    u32  auto_charge:     1;           //自动充电状态      Bit2
    u32  manualCharge:     1;         //手动充电状态      Bit3
    u32 flagAutoElectrodeConnect:  1;    //自动充电电极对接状态    Bit4
    u32 flagManualElectrodeConnect:  1;  //手动充电电极对接状态  Bit5
```

```
    u32  flagCpuButtonEnterStart:  1;    //机器人按键开机      Bit6
    u32  flagCpuPowerEnoughStart:  1;    //机器人电量充足开机    Bit7
    u32  flagCpuButtonUpShutdown:  1;    //机器人按键关机      Bit8
    u32  flagCpuPowerLowShutdown:  1;    //机器人电量低关机    Bit9
    u32  start_key:         1;    //机器人上电按钮      Bit10
    u32 flagBatComStatueErr:      1;      //电池通信出错状态        Bit11
    u32 flagMasterComStatue:    1;    //主控通信状态          Bit12
    u32 flagBatPowerEnoughStatue:  1;    //电量充足标志          Bit13
    u32 flagChargingEquipmentError:  1;  //充电桩没有输出，当通信成功，对接成
功，打开电池不能充电  Bit14
    u32 flagChargingBatOpen:    1;    //自动充电电池打开标志      Bit15
    u32 stop_key:         1;    //急停开关状态        Bit16
    u32 flagCpuNeedRestart:      1;    // 系统需要重启        Bit17
    u32 rev:                14;       //保留
} StaFlag;

//电源通路状态上报
struct _Power_Manager
{
  TNBHead    head;
  u16 Power_Status;  /*bit0~bit6: 5v 通道,12v 通道 1~5 最高位为软关机指令*/
  u16       Volt[8];  /* 电压采集数据 bit0~bit6: 5v 通道,12v 通道 1~5 / 10mV */
  u16       Curr[8];  /*电流采集数据 bit0~bit6: 5v 通道,12v 通道 1~5 / 1mA */
  stdbat_info   StdBat_Info;
    StaFlag      PowerManagerSysStatues;
}__PACKED__;
typedef struct _Power_Manager Power_Manager;
```

附 1：

```
u16 CRC16(const u8 *buffer, u32 len)
{
    u16 crc = 0;
    while (len--) crc = (crc >> 8) ^ crc16_table[(crc ^ (*buffer++)) & 0xff];
    return crc;
}
u16 const crc16_table[256] = {
    0x0000, 0xC0C1, 0xC181, 0x0140, 0xC301, 0x03C0, 0x0280, 0xC241,
    0xC601, 0x06C0, 0x0780, 0xC741, 0x0500, 0xC5C1, 0xC481, 0x0440,
    0xCC01, 0x0CC0, 0x0D80, 0xCD41, 0x0F00, 0xCFC1, 0xCE81, 0x0E40,
    0x0A00, 0xCAC1, 0xCB81, 0x0B40, 0xC901, 0x09C0, 0x0880, 0xC841,
    0xD801, 0x18C0, 0x1980, 0xD941, 0x1B00, 0xDBC1, 0xDA81, 0x1A40,
    0x1E00, 0xDEC1, 0xDF81, 0x1F40, 0xDD01, 0x1DC0, 0x1C80, 0xDC41,
    0x1400, 0xD4C1, 0xD581, 0x1540, 0xD701, 0x17C0, 0x1680, 0xD641,
    0xD201, 0x12C0, 0x1380, 0xD341, 0x1100, 0xD1C1, 0xD081, 0x1040,
    0xF001, 0x30C0, 0x3180, 0xF141, 0x3300, 0xF3C1, 0xF281, 0x3240,
    0x3600, 0xF6C1, 0xF781, 0x3740, 0xF501, 0x35C0, 0x3480, 0xF441,
    0x3C00, 0xFCC1, 0xFD81, 0x3D40, 0xFF01, 0x3FC0, 0x3E80, 0xFE41,
    0xFA01, 0x3AC0, 0x3B80, 0xFB41, 0x3900, 0xF9C1, 0xF881, 0x3840,
    0x2800, 0xE8C1, 0xE981, 0x2940, 0xEB01, 0x2BC0, 0x2A80, 0xEA41,
    0xEE01, 0x2EC0, 0x2F80, 0xEF41, 0x2D00, 0xEDC1, 0xEC81, 0x2C40,
    0xE401, 0x24C0, 0x2580, 0xE541, 0x2700, 0xE7C1, 0xE681, 0x2640,
    0x2200, 0xE2C1, 0xE381, 0x2340, 0xE101, 0x21C0, 0x2080, 0xE041,
    0xA001, 0x60C0, 0x6180, 0xA141, 0x6300, 0xA3C1, 0xA281, 0x6240,
    0x6600, 0xA6C1, 0xA781, 0x6740, 0xA501, 0x65C0, 0x6480, 0xA441,
    0x6C00, 0xACC1, 0xAD81, 0x6D40, 0xAF01, 0x6FC0, 0x6E80, 0xAE41,
    0xAA01, 0x6AC0, 0x6B80, 0xAB41, 0x6900, 0xA9C1, 0xA881, 0x6840,
    0x7800, 0xB8C1, 0xB981, 0x7940, 0xBB01, 0x7BC0, 0x7A80, 0xBA41,
    0xBE01, 0x7EC0, 0x7F80, 0xBF41, 0x7D00, 0xBDC1, 0xBC81, 0x7C40,
    0xB401, 0x74C0, 0x7580, 0xB541, 0x7700, 0xB7C1, 0xB681, 0x7640,
    0x7200, 0xB2C1, 0xB381, 0x7340, 0xB101, 0x71C0, 0x7080, 0xB041,
    0x5000, 0x90C1, 0x9181, 0x5140, 0x9301, 0x53C0, 0x5280, 0x9241,
    0x9601, 0x56C0, 0x5780, 0x9741, 0x5500, 0x95C1, 0x9481, 0x5440,
    0x9C01, 0x5CC0, 0x5D80, 0x9D41, 0x5F00, 0x9FC1, 0x9E81, 0x5E40,
    0x5A00, 0x9AC1, 0x9B81, 0x5B40, 0x9901, 0x59C0, 0x5880, 0x9841,
    0x8801, 0x48C0, 0x4980, 0x8941, 0x4B00, 0x8BC1, 0x8A81, 0x4A40,
    0x4E00, 0x8EC1, 0x8F81, 0x4F40, 0x8D01, 0x4DC0, 0x4C80, 0x8C41,
    0x4400, 0x84C1, 0x8581, 0x4540, 0x8701, 0x47C0, 0x4680, 0x8641,
    0x8201, 0x42C0, 0x4380, 0x8341, 0x4100, 0x81C1, 0x8081, 0x4040
};
```