Handheld Laser Welding Head Driver API Description

Version	Date	Authur	Auditing	Note
V1.0	2024-12-30	李军波		first edition

content

1. Overview

This protocol is used for the API interface between the handheld control card and the handheld laser welding head driver (hereinafter referred to as the “laser head”), which is written in accordance with the communication protocol.

1.1 Basic Parameters of Communication Module

In this communication protocol, the main control card of the welder acts as the master and the driver acts as the slave, and the communication protocol is compatible with the Modbus RTU specification. The recommended communication interval is more than 100ms.

Slave address	0x09
Coding format	8-bit binary
Data bit	8 bit
Parity bit	Free
Stop bit	1 bit
Baud rate	115200

2.1 Function Table

Function	Description
u8 setAutomaticallyObtainData (u16 var1,u8 var2);	Set up the automatic update obtain data function
u8 setScanFrequencyData(u8 var);	Set the scan speed（HZ）
u16 setScanWidthData(u16 var);	Set the scan width（mm）
u8 setDriveType(u8 var);	Set the driver type
u8 setAddress(u8 var);	Set the driver address 0~247
u8 setAlarm(u8 var);	Set up alarm detection
u8 setRestoreData(u8 var);	Restore factory parameters
u8 saveParameters(u8 var);	Save parameters
u8 setGalvanometerEn(u8 var);	Galvo amplitude enable
u8 setScanCorrect(u8 var);	Galvo swing width correction
s8 setCenterCorrect(s8 var);	Red light center correction
u16 setDriveTempAlarm(u16 var);	Drive temperature alarm threshold

u16setProtectiveGlassTempAlarm(u16 var);	Protective glass temperature alarm threshold
u16 setCollimatorTempAlarm(u16 var);	Collimer temperature alarm threshold
u8 setP15V_V_AlarmHigh(u8 var);	+ 15v voltage alarm threshold-upper limit voltage
u8 setP15V_V_AlarmLow(u8 var);	+ 15v voltage alarm threshold-lower limit voltage
u8 setN15V_V_AlarmHigh(u8 var);	-15v voltage alarm threshold-upper limit voltage
u8 setN15V_V_AlarmLow(u8 var);	-15v voltage alarm threshold-lower limit voltage
u16 setP15V_A_AlarmHigh(u16 var);	+ 15v Current alarm threshold-upper limit current
u16 setN15V_A_AlarmHigh(u16 var);	-15v Current alarm threshold-upper limit current
u8 setTempDiffAlarm(u8 var);	Temperature alarm restore the voltage difference value
u8 setGunLED(u8 var);	Set the laser head LED turn on / off
u8 setDriveComHeartbeat(u8 var);	Set up the drive communication heartbeat function
u8 setDriveComErrorTime(u8 var);	Set the function of drive communication abnormal time
void setDriverCOMTestEn(u8 var1 ,u8 var2);	Set up communication with driver test enable
u8 getAutomaticallyObtainData(u16 *var1,u8 *var2);	Obtain the settings automatically obtain data parameters
u8 getScanFrequencyData();	Obtain the galvo scan speed（HZ）
u16 getScanWidthData();	Obtain the galvo scan width（mm）
u8 getDriveData();	Obtain the driver type
u8 getAddressData();	Obtain the driver address
u8 getAlarmData();	Obtain the alarm detection mode
u8 getGalvanometerEnData();	Obtain the galvo amplitude enable
u8 getScanCorrectData();	Obtain the galvo swing width correction
s8 getCenterCorrectData();	Obtain the red light center correction
u16 getDriveTempAlarmData();	Obtain the drive temperature alarm threshold
u16 getProtectiveGlassempAlarmData();	Obtain the protective glass temperature alarm threshold
u16 getCollimatorTempAlarmData();	Obtain the collimator lens temperature alarm threshold
u8 getP15V_V_AlarmHighData();	Obtain the + 15v voltage alarm threshold-upper limit voltage

u8 getP15V_V_AlarmLowData();	Obtain the + 15v voltage alarm threshold-lower limit voltage
u8 getN15V_V_AlarmHighData();	Obtain the-15v voltage alarm threshold-upper limit voltage
u8 getN15V_V_AlarmLowData();	Obtain the-15v voltage alarm threshold-lower limit voltage
u8 getP15V_A_AlarmHighData();	Obtain the + 15v voltage alarm threshold-upper limit voltage
u8 getP15V_A_AlarmHighData();	Obtain the + 15v current alarm threshold-upper limit current
u8 getTempDiffAlarmData();	Obtain the temperature alarm restore voltage difference value
s16 getP15_VData();	Obtain the + 15V supply voltage
s16 getN15_VData();	Obtain the-15V supply voltage
s16 getDriveTempData();	Obtain the motor driver card temperature
s16 getProtectiveGlassTempData();	Obtain the protective glass temperature
s16 getCollimatorTempData();	Obtain the collimator temperature （This version does not have）
u8 getP15_V_AlarmData();	Obtain the + 15v voltage alarm
u8 getN15_V_AlarmData();	Obtain the -15v voltage alarm
u8 getP15_A_AlarmData();	Obtain the + 15v current alarm （This version does not have）
u8 getN15_A_AlarmData();	Obtain the -15v current alarm （This version does not have）
u8 getDriveAlarmData();	Obtain the motor drive card temperature alarm
u8 getProtectiveGlassAlarmData();	Obtain the protective glass temperature alarm
u8 getCollimatorAlarmData();	Obtain the collimator temperature alarm （This version does not have）
u16 getDriveID_HData();	Obtain the drive id high 2 bytes
u16 getDriveID_LData();	Obtain the drive id low 2 bytes
u16 getDriveHardwareVData();	Obtain the drive hardware version
u16 getDriveSoftwareVData();	Obtain the drive software version
u8 getDriveCOMData();	Obtain communication status
u8 getDriveComHeartbeat();	Obtain the heartbeat enabling status
u16 getDriveComErrorTime();	Obtain the communication abnormal time
u8 getCOMPacketLossRate();	Obtain communication packet loss rate%
u32 getCOMPacketLossRate();	Obtain communication packet loss number
u32 getCOMPacketSendData();	Obtain the total number of packets sent by the communication
u32 getCOMPacketReceiveData();	Obtain the total number of normal packets received by the communication

u32 getCOMPacketErrData();	Obtain the total number of abnormal packets received by the communication
u8 getGunKeyData();	Obtain the laser head key value
u8 getDoubleKeyData();	Obtain the laser head double key value （This version does not have）
u8 getWireFeederKeyData();	Obtain the laser head key value of wire feeder
u8 getGunLEDData();	Obtain the laser head indicator light value
u8 getGunCOMData();	Obtain the laser head and the driver communication value
u16 getGunSoftwareVData();	Obtain the software device hardware version
u16 getGunHardwareVData();	Obtain the drive hardware version
u16 getGunHardwareVData();	Obtain the drive hardware version
u8 getUsartLen();	Obtain the serial port data length
u8 getUsartData(u8 *buff,u8 len);	Obtain the serial port data
void initDriveAPIData();	Initialize the data in the API
u8 initAddress(u8 var);	Initialize the address
u8 clearCOMTestData();	Clear the test communication data
void insertDataFunction(u8 var);	Write the data to the API cache
u8 SendDataFunction(u8 *buff);	Obtain the packet to be sent
u8 ParsingData();	Analyze packets

2. API Interface introduction

2.1 Set the parameter function

u8 setAutomaticallyObtainData(u16 var1,u8 var2);

Set up the automatic update obtain data function

Parameter var1- -Starting Address: If entering an invalid address, stop obtaining data automatically

var2 -- Number of obtaining: If input 2, obtain 2 address data from var1 address

return: 1 --- automatic obtaining， 0 --- stop automatic obtaining

E.g.： u8 a=setAutomaticallyObtainData(0x00,0)； ---stop obtaining data a = 0

u8 a=setAutomaticallyObtainData(0x01,1)； ---obtain 1 address data a=1

u8 a=setAutomaticallyObtainData(0x01,5)； ---obtain 5 address data a=1

u8 a=setAutomaticallyObtainData(0x01,0)； ---obtain all address data a=1

u8 setScanFrequencyData(u8 var);

Set the galvo scan speed（HZ）,

Parameter var -- scan speed（HZ）： welding range 5~150HZ，cleaning range 5~50HZ

return: return to set parameter value

E.g.： u8 a=setScanFrequencyData(0x1E)； ---set parameter a = 0x1E

u16 setScanWidthData(u16 var);

Set the galvo scan width（mm），

Parameter var -- scan width（mm）： welding range 0.0~5.0mm，cleaning range 0.0~85.0mm，expand 10 times

Retain 1 decimal place.

return: return to set parameter value

E.g.： u16 a=setScanWidthData(0x1E)； ---Setting parameter 30，Corresponding to 3mm scan width， a = 0x1E

u8 setDriveType(u8 var);

Set the driver type，

Parameter var -- driver type：0----welding mode，1----cleaning mode

return: return to set parameter value

E.g.： u8 a=setDriveType(0)； ---setting welding mode a = 0

u8 setAddress(u8 var);

Set the driver address 0~247，

Parameter var -- driver address：the default address is 0x09，0x00 address is an advanced address，When forget the address, user can use the 0x00 address to modify the data

return: return to set parameter value

E.g.： u8 a=setAddress(0x09)； ---set parameter 0x09， a = 0x09

u8 setAlarm(u8 var);

Set up the alarm detection, and the abnormal communication alarm of the laser head and the driver card are independent.

Parameter var -- alarm detection：1---turn on alarm，0----turn off alarm
return: return to set parameter value
E.g.： u8 a=setAddress(0x09)； ---set parameter 0x09， a = 0x09

u8 setRestoreData(u8 var);

Set to restore factory parameters. Note: if the address is 0x00, after the factory data is restored, the address is 0x09, and the other address and the driver type will not change.
Parameter var --save parameter：1---restore factory parameter，0----invalid
return: return to set parameter value
E.g.： u8 a=setRestoreData(0x01)； ---set parameter 0x01， a = 0x01

u8 saveParameters(u8 var);

Set to save parameters. Note: Some parameters will be automatically saved after modification, please see the protocol document for details.
Parameter var --save parameter：1---save parameter，0----invalid。
return: return to set parameter value
E.g.: u8 a=saveParameters(0x01)； ---set parameter 0x01， a = 0x01

u8 setGalvanometerEn(u8 var);

Galvo swing enable
Parameter var --Galvo swing enable: 1---start swing, 0----stop swing
return: return to set parameter value
E.g.: u8 a=setGalvanometerEn(0x01); ---set parameter 0x01, a = 0x01

u8 setScanCorrect(u8 var);

Galvo swing width correction. It can be manually adjusted for width correction, expand by 100 times, retain 2 decimal places, the default is 0x60, corresponding to 1.
Parameter var --Galvo swing width correction: less than 0x60---swing width lessening, greater than 0x60----swing width largen, recommended adjusting pace is at each time + 1
return: return to set parameter value
E.g.: u8 a=setScanCorrect(0x61); ---set parameter 0x61, a = 0x61

s8 setCenterCorrect(s8 var);

Red light center correction, center correction can be manually fine-adjusted, expand by 100 times, retain 2 decimal places, range: -100（-1.00）～ 100（1.00）, the default is 0x00.
Parameter var --red light center correction: greater than 0x00---red light goes left, less than 0x00----red light goes right, recommended adjusting pace is at each time + 1
return: return to set parameter value
E.g.: s8 a=setCenterCorrect(0x10); ---set parameter 0x10, red light goes right, a = 0x10

u16 setDriveTempAlarm(u16 var);

Driver temperature alarm threshold, expand by 10 times, retain 1 decimal place, range: 0.0~100.0, the default is 65.
Parameter var --driver temperature alarm threshold: 0x00---on alarm, greater than 0x00 ---- start the alarm function.
return: return to set parameter value
E.g.: u16 a=setDriveTempAlarm(0x41); ---set parameter 0x41, temperature alarm threshold 65, a = 0x41

u16setProtectiveGlassTempAlarm(u16 var);

Protective glass temperature alarm threshold, expand by 10 times, retain 1 decimal place, range: 0.0~100.0 the default is 65.

Parameter var --protective glass temperature alarm threshold: 0x00---no alarm, greater than 0x00 ---- start the alarm function.

return: return to set parameter value

E.g.: u16 a=u16setProtectiveGlassTempAlarm(0x41); ---set parameter 0x41, temperature alarm threshold 65, a = 0x41

u16 setCollimatorTempAlarm(u16 var);

Collimator lens temperature alarm threshold, expand by 10 times, retain 1 decimal place, range:0.0~100.0 the default is 65. This version does not have this feature.

Parameter var --collimator lens temperature alarm threshold: 0x00---no alarm, greater than 0x00 ---- start the alarm function.

return: return to set parameter value

E.g.: u16 a=setCollimatorTempAlarm(0x41); ---set parameter 0x41, temperature alarm threshold 65, a = 0x41

u8 setN15V_V_AlarmHigh(u8 var);

+15v voltage alarm threshold -upper limit voltage. Range 0~17 the default is 17

Parameter var -- +15v voltage alarm threshold -upper limit voltage: 0x00---no alarm, greater than 0x00 ---- start the alarm function.

return: return to set parameter value

E.g.： u8 a=setN15V_V_AlarmHigh(0x11)； ---set parameter 0x11，voltage threshold 17， a = 0x11

u8 setN15V_V_AlarmLow(u8 var);

+15v voltage alarm threshold -lower limit voltage，range 0~17 the default is 12，

Parameter var -- +15v voltage alarm threshold -lower limit voltage：0x00---no alarm，greater than 0x00 ---- start the alarm function.

return: return to set parameter value

E.g.： u8 a=setN15V_V_AlarmLow(0x0C)； ---set parameter 0x0C，voltage threshold 12， a = 0x0C

u8 setP15V_V_AlarmHigh(u8 var);

-15v voltage alarm threshold -upper limit voltage，range 0~-17 the default is -17，

Parameter var -- -15v voltage alarm threshold -upper limit voltage：0x00---no alarm，greater than 0x00 ---- start the alarm function.

return: return to set parameter value

E.g.： u8 a=setP15V_V_AlarmHigh(0x11)； ---set parameter 0x11，voltage threshold 17， a = 0x11

u8 setP15V_V_AlarmLow(u8 var);

-15v voltage alarm threshold -lower limit voltage，range 0~17 the default is -12，

Parameter var -- -15v voltage alarm threshold -lower limit voltage：0x00---no alarm，greater than 0x00 ---- start the alarm function.

return: return to set parameter value

E.g.： u8 a=setP15V_V_AlarmLow(0x0C)； ---set parameter 0x0C，voltage threshold 12， a = 0x0C

u16 setN15V_A_AlarmHigh(u16 var);

+15v current alarm threshold -upper limit current， range 0~1500mA the default is 600mA， This version does not have this feature.

Parameter var -- +15v current alarm threshold -upper limit current: 0x00---no alarm， greater than 0x00 ---- start the alarm function.

return: return to set parameter value

E.g.： u16 a=setN15V_A_AlarmHigh(0x0258)； ---set parameter 0x0258，voltage threshold 600， a = 0x0258

u16 setP15V_A_AlarmHigh(u16 var);

-15v current alarm threshold -upper limit current， range 0~1500mA the default is 600mA，This version does not have this feature.

Parameter var -- -15v current alarm threshold -upper limit current: 0x00---no alarm， greater than 0x00 ---- start the alarm function.

return: return to set parameter value

E.g.： u16 a=setP15V_A_AlarmHigh(0x0258)； ---set parameter 0x0258，current threshold600， a = 0x0258

u8 setTempDiffAlarm(u8 var);

Temperature alarm restore voltage difference, range 0~10, the default is 5℃, when the temperature returns to 5 ℃ below the alarm threshold, cancel the alarm. All temperature alarms use this parameter.

Parameter var --temperature alarm restore voltage difference

return: return to set parameter value

E.g.: u8 a=setTempDiffAlarm(0x05) ; ---set parameter 0x05, a = 0x05

u8 setGunLED(u8 var);

Set the laser head LED on/off, standby- -yellow light flashes, laser output -- green light always on, alarm -- red light flashes

Parameter var --set the laser head LED on/off: 0x00 --- yellow light flashes, 0x01 ---- green light always on, 0x02 ---- red light flashes.

return: return to set parameter value

E.g.: u8 a=setGunLED(0x01); ---set parameter 0x01, laser output -- green light always on, a = 0x01

u8 setDriveComHeartbeat(u8 var);

Set the drive communication heartbeat function, which is enabled by default.

Parameter var -- set up the drive communication heartbeat function: 0x01 --- start heartbeat function, 0x00 ---- stop heartbeat function.

return: return to set parameter value

E.g.: u8 a=setDriveComHeartbeat(0x01); ---set parameter 0x01, start heartbeat function, a = 0x01

u8 setDriveComErrorTime(u8 var);

Set the driver communication exception time function, this depends on the

SendDataFunction () function traversal time, assuming that traversal of the data once 100ms, var set to 20, then the exception time is 2000ms, more than 2000ms did not receive the correct data, communication exceptions. range 0~200 the default is 20. The total time should not be less than 1000ms.

Parameter var -- set the driver communication exception time function: 0x14 --- 20.

return: return to set parameter value

E.g.: u8 a=setDriveComErrorTime(0x14) ; ---set parameter 0x14, start heartbeat function, a = 0x14

void setDriverCOMTestEn(u8 var1 ,u8 var2);

Set up the driver communication test enable, after enabling, it can be able to count the number of sending data, the number of receiving data, the number of errors, etc., the use of the test needs to be called before clearCOMTestData() for data 0, specific use as the following case.

Parameter var1 -- sent number statistics enable: 0x01---start，0x00 ---- stop.

var2 -- received number statistics enable: 0x01---no alarm，0x00 ----stop.

E.g.: setDriverCOMTestEn(0x01,0x01); ---all start

2.2 Obtain data function

u8 getAutomaticallyObtainData(u16 *var1,u8 *var2);

Obtain the parameters for setting up automatic data obtaining

Obtain data 1: var1- -start address

Obtain data 2: var2- -quantity

Data 3 return: return to whether to auto-fetch, 1---turn on automatic obtain, 0----turn off automatic obtain

u8 getScanFrequencyData();

Obtain galvo scan speed（HZ）. Type u8

u16 getScanWidthData();

Obtain galvo scan width（mm）. Expand by 10 times, retain 1 decimal place. Type u16,welding mode range 0.0~5.0mm, cleaning mode range 0.0~85.0mm

u8 getDriveData();

Obtain driver type. Type u8 , Driver type: 0----welding mode, 1----cleaning mode

u8 getAddressData();

Obtain driver address. Type u8 , address 0~247

u8 getAlarmData();

Obtain alarm detection mode. Type u8 , 1---turn on alarm, 0----turn off alarm.

u8 getGalvanometerEnData();//Set the swing enabling

Obtain galvo swing enabling. Type u8 , 1---start swing, 0----stop swing.

u8 getScanCorrectData();

Obtain the galvo swing width correction. Type u8, expand by 100 times, retain 2 decimal places, value 0x60, corresponding to 1.

s8 getCenterCorrectData();

Obtain red light center correction. Type s8, expand by 100 times, retain 2 decimal places,value 0x60--100, range -100（-1.00）～ 100（1.00）

u16 getDriveTempAlarmData();

Obtain driver temperature alarm threshold. Type u16, expand by 10 times, retain 1 decimal place，range 0.0~100.0

u16 getProtectiveGlassTempAlarmData();

Obtain protective glass temperature alarm threshold. Type u16, expand by 10 times, retain 1 decimal place, range 0.0~100.0

u16 getCollimatorTempAlarmData();

Obtain collimator lens temperature alarm threshold. Type u16，expand by 10 times, retain 1 decimal place，range 0.0~100.0

u8 getP15V_V_AlarmHighData();

Obtain +15v voltage alarm threshold -upper limit voltage. Type u8，range 0~17

u8 getP15V_V_AlarmLowData();

Obtain +15v voltage alarm threshold -lower limit voltage. Type u8，range 0~17

u8 getN15V_V_AlarmHighData();

Obtain -15v voltage alarm threshold -upper limit voltage. Type u8，range 0~17

u8 getN15V_V_AlarmLowData();

Obtain -15v voltage alarm threshold -lower limit voltage. Type u8，range 0~17

u8 getP15V_A_AlarmHighData();

Obtain +15v voltage alarm threshold -upper limit voltage. Type u8，range 0~17

u8 getP15V_A_AlarmHighData();

Obtain +15v current alarm threshold -upper limit current. Type u8，range 0~1500mA

u8 getTempDiffAlarmData();

Obtain temperature alarm restore voltage difference. Type u8

s16 getP15_VData();

Obtain +15V supply voltage. Type u8，range 0~17

s16 getN15_VData();

Obtain -15V supply voltage. Type u8，range -17~0

s16 getDriveTempData();

Obtain motor driver card temperature. Type s16, range -20.0-999.0 expand by 10 times, retain 1 decimal place

s16 getProtectiveGlassTempData();

Obtain protective glass temperature. Type s16 , range -20.0-999.0 expand by 10 times, retain 1 decimal place

s16 getCollimatorTempData();

Obtain the collimator temperature. Type s16, range -20.0-999.0 expand by 10 times, retain 1 decimal place.

u8 getP15_V_AlarmData();

Obtain the + 15v voltage alarm. Type u8， 0---no alarm 1---alarm

u8 getN15_V_AlarmData();

Obtain the-15v voltage alarm. Type u8， 0---no alarm 1---alarm

u8 getN15_A_AlarmData();

Obtain the-15v voltage alarm. Type u8， 0---no alarm 1---alarm

u8 getN15_A_AlarmData();

Obtain the-15v current alarm. Type u8， 0---no alarm 1---alarm

u8 getDriveAlarmData();

Obtain the motor drive card temperature alarm. Type u8， 0---no alarm 1---alarm

u8 getProtectiveGlassAlarmData();

Obtain the protective glass temperature alarm. Type u8， 0---no alarm 1---alarm

u8 getCollimatorAlarmData();

Obtain the collimator temperature alarm. Type u8， 0---no alarm 1---alarm

u16 gegetDriveID_HData();

Obtain the drive id high 2 bytes

u16 getDriveID_LData();

Obtain the drive id low 2 bytes

u16 getDriveHardwareVData();

Obtain the drive hardware version

u16 getDriveSoftwareVData();

Obtain the drive software version

u8 getDriveCOMData();

Obtain the communication status. Type u8, 0- - -abnormal; 1- - -normal, 2- - - -heartbeat function is not on. User needs to turn on the heartbeat function

u8 getDriveComHeartbeat();

Obtain the heartbeat enabling conditions. Type u8, 0- - -off; 1- - -on.(On by default)

u16 getDriveComErrorTime();

Obtain the communication abnormal time. Type u16.

u8 getCOMPacketLossRate();

Obtain the communication packet loss rate%. Type u8 , range: 0 - 100, extend 100 times.

u32 getCOMPacketLossRate();

Obtain the number of lost packets. Type u32.

u32 getCOMPacketSendData();

Obtain the total number of packets sent by communication. Type u32.

u32 getCOMPacketReceiveData();

Obtain the total number of normal packets received by communication. Type u32

u32 getCOMPacketErrData();

Obtain the total number of abnormal data packets received by the communication. Type u32

//Laser head key

u8 getGunKeyData();

Obtain the laser head key value. Type u8. When using this value as the switch, mainly need to check the laser head communication, data processing time and other issues.

u8 getDoubleKeyData(); In this version it is not yet available

Obtain the double click key value, double click in 1s is 1, otherwise is 0. Type u8. When using this value as the switch, mainly need to check the laser head communication, data processing time and other issues.

u8 getWireFeederKeyData();

Obtain laser head wire feeding key value. Type u8.

u8 getGunLEDData();

Obtain laser head indicator light value. Type u8, 0x00 --- yellow light flashing, 0x01 ---- green light flashing, 0x02 ---- red light flashing.

u8 getGunCOMData();

Obtain the laser head and driver communication value. Type u8, 0x00 --- communication abnormal, the ALM signal on the DB20 connector will be raised to indicate an alarm，0x01 ---- communication normal.

u16 getGunSoftwareVData();

Obtain the software device hardware version

u16 getGunHardwareVData();

Obtain the drive hardware version

u16 getGunHardwareVData();

Obtain the drive hardware version

u8 getUsartLen();

Obtain the serial port data length, type u8

u8 getUsartData(u8 *buff,u8 len);

Obtain the serial port data, return data -actual data length, buff- -data, len- -the data length to be obtained.

2.3 Important function

void initDriveAPIData();

Initialize the data in the API

u8 initAddress(u8 var);

Initialize address, only valid in API, not write driven

u8 clearCOMTestData();

Empty the test communication data

void insertDataFunction(u8 var);

Write the data to the API cache

u8 SendDataFunction(u8 *buff);

Obtain the packet to be sent, see the following example for usage suggestions.

u8 ParsingData();

Parsing packets, see the following examples for usage recommendations.

3. Use cases and precautions

1、For API normal use, user must call **“u8 initAddress(u8 var);void insertDataFunction(u8 var);u8 SendDataFunction(u8 *buff);u8 ParsingData();”**. Lacking one of these five functions, it can not work properly, the function is placed in the position as the following case

```c
int main(void)
{

  /***** 配置过程省略*****/
  int a=0,b=0,len;
  u8 buff[250];
  initDriveAPIData();//数据初始化 *
  initAddress(0x09);//地址初始化  *
  setAutomaticallyObtainData(0x01,0);//自动获取全部数据
  setScarWidthData(30); //设置摆宽3mm
  setScarFrequencyData(50);//设置速度50hz

  while(1)
  {
    ParsingData();//自动解析数据函数，放在主函数运行，大约在1ms访问一下。*
    if(a>100) //发送数据速度在100ms每次
    {
        len = SendDataFunction(buff);//获取要发送的数据包   *
        printf_data_send(buff,len);//通过串口485将数据发出
        a=0;
    }

    if(b>=1000)
    {
      /*
        可直接调用get函数获取数据或者调用set函数设置数据。
      */
      printf("%d",getP15_VData());//打印电压值
      b=0;
    }
    a++;
    b++;
    delay_ms(1);
  }

}

void USART1_IRQHandler(void)
{
    u8 Res;
    if(USART_GetITStatus(USART1, USART_IT_RXNE) != RESET)
    {
        Res =USART_ReceiveData(USART1); //读取接收到的数据
        insertDataFunction( Res); //将该函数放入串口接收中断里面 *
    }

}
```

Note：1、Send function **u8 SendDataFunction(u8 *buff)**; with the serial port output call time is recommended in > 100ms, if cycle send too fast, it maybe not stable, if call too slow, it will affect the speed of obtaining data.

2、 Parsing function **u8 ParsingData()**; the call time is recommended to be 1ms~10ms. If call too fast will waste cpu resources, if call too slow will affect the data obtaining speed, and may lead to time loss.

3、 Obtain the serial data function **void insertDataFunction (u8 var)**; It's fine to put the function on the serial port interrupt, and it's fine to write the data outside, but make sure that the data written is complete and fast.

4、 The data in Get is a slow beat, so take note.

5、 Set set data will not be sent out immediately, there is a buffer, and also not sent in the order of the call, so use **u8 save_Parameters (u8 var)**; when save the parameters, pay attention to sent out all the set parameters, then call to save the function.

6、 It is recommended to always check the connection to the driver and if the connection is abnormal, the laser must be turned off for troubleshooting.

2、Communication function test requires the **u8 clearCOMTestData(); void setDriverCOMTestEn (u8 s_var, u8 r_var)**; This function is mainly to test whether the communication is stable. Use as follows

```
int main(void)
{

  /***** 配置过程省略*****/
  int a=0,b=0,len;
  u8 buff[250];
  initDriveAPIData();//数据初始化 *
  initAddress(0x09);//地址初始化  *
  setAutomaticallyObtainData(0x01,0);//自动获取全部数据
  clearCOMTestData();//清空数值
  setDriverCOMTestEn(1,1);//开始统计
  while(1)
  {
    ParsingData();//自动解析数据函数，放在主函数运行，大约在1ms访问一下。*
    if(a>100) //发送数据速度在100ms每次
    {
        len = SendDataFunction(buff);//获取要发送的数据包   *
        printf_data_send(buff,len);//通过串口485将数据发出
        a=0;
    }
    if(b>=11000)
    {
      /*
        可直接调用get函数获取数据或者调用set函数设置数据。
      */
      //分别打印丢包率，丢包数，发送总数，接收正常总数，错误数
       setDriverCOMTestEn(0,0);//完成
      printf("%d",getCOMPacketLossRate());
      printf("%d",getCOMPacketLossData());
      printf("%d",getCOMPacketSendData());
      printf("%d",getCOMPacketReceiveData());
      printf("%d",getCOMPacketErrData());
      b=0;
    }
    else  if(b>=10000) //统计10s发收数据
    {
       setAutomaticallyObtainData(0x00,0);//停止发送数据
       setDriverCOMTestEn(0,1);//发送完成，但是接收会慢一个节拍，所以还没接收完。
    }
    a++;
    b++;
    delay_ms(1);
  }

}
void USART1_IRQHandler(void)
{
    u8 Res;
    if(USART_GetITStatus(USART1, USART_IT_RXNE) != RESET)
    {
        Res =USART_ReceiveData(USART1); //读取接收到的数据
        insertDataFunction( Res); //将该函数放入串口接收中断里面 *
    }

}
```

Note: If all the data is automatically obtained and travels very quickly, a small number of packet exceptions are normal. So user needs to check the right speed to match through this function.