纳秒GS激光器通讯协议

一、串口通讯设置

1、串口通讯参数：

波特率： 9600
奇偶校验： None
数据位 ：8
停止位 ：one

二、通讯规约

本产品采用开放式的 MODBUS－RTU 通讯规约，详细规约请参考相关文件；

MODBUS－RTU 通讯规约示例:

2.1、功能码 03H(0x03)：读取保存寄存器

数据长度: 0001H~0020H, 最多可一次读取 32 个连续寄存器

说明： 读取的是 16 位数据， 高字节位在前，低字节在后。

例:

	地址	功能码	起始地址	寄存器个数	CRC 校验	
主机命令：	01	03	00 00	00 01	CRCH CRCL	(8 字节)

	地址	功能码	字节数	寄存器数据	CRC 校验	
从机响应：	01	03	02	00 10	CRCH CRCL	(7 字节)

2.2、功能码 06H(0x06)：预置单寄存器

寄存器数量:可一次设置 1 个寄存器

	地址	功能码	起始地址	写寄存器数据	CRC 校验	
例：主机发送：	01	06	00 00	00 01	CRCH CRCL	(8 字节)
从机响应：	与主机发送的数据完全相同					(8 字节)

三、寄存器定义

表 1:只读寄存器,功能码 03H

寄存器地址	变量	备注	描述
0000H	REG_TDI	Return laser diode temperature in degrees Centigrade.	查询泵浦源温度
0001H	REG_TK1	Return the number of TEMP1 value	查询晶体 1 温度
0002H	REG_TK2	Return the number of TEMP2 value	查询晶体 2 温度
0003H	REG_ENVACT	Return the number of environment temperature value	环境温度
0004H	REG_LASACT	Return the number of laser temperature value	Laser 温度
0005H	REG_SERVICE_TIME		查询使用时间
0006H	REG_ALARM		报警寄存器
	BIT13~ BIT15	保留	
	BIT12	Time2_Alera	
	BIT11	Time1_Alera	
	BIT10		INTERLOCK OPEN ERROR
	BIT9	XL1ER_TEMP_L	二倍频温度过低报警
	BIT8	XL1ER_TEMP_H	二倍频温度过高报警
	BIT7	XL2ER_TEMP_H	三倍频温度过高报警
	BIT6	XL2ER_TEMP_L	三倍频温度过低报警
	BIT5	LH_TEMP_L	腔体温度过低报警
	BIT4	LH_TEMP_H	腔体温度过高报警

	BIT3	ENML_TEMP_L	环境温度过低报警
	BIT2	ENML_TEMP_H	环境温度过高报警
	BIT1	LD_TEMP_L	LD 温度过低报警
	BIT0	LD_TEMP_H	LD 温度过高报警

表 2：可读、 可写寄存器定义,功能码 03H/06H

寄存器地址	变量	备注	描述
0007H	REG_DIO	Turn the diode current on or off n = 1 is diode current on. n = 0 is diode current off	开关 DIODE
0008H	REG_IDI	Set the diode current to nn.n which ranges from 0 to 50 Amps	设置电流值 0-50A
0009H	REG_PRF	Set the number of pulses ~~Allowed range is 10~100000~~ Allowed range is 100~15000	设置 PRF 值 1k-150k
000AH	REG_SHT	Turn the shutter on or off n = 1 turns shutter on n = 0 turns shutter off	SHT 开关
000BH	REG_QSW	Turn the Q-Switch on or off n = 1 turns Q-Switch on n = 0 turns Q-Switch off	QS-ON 开关
000CH	REG_EXT	Turn the QS-Trigger on or off n = 1 turns QS-Trigger on n = 0 turns QS-Trigger off	QS-EXT 开关
0010H	REG_PWM	Set the number of PWM value Allowed range is 20~380 20---100% Max 380---10% Min	设置 PWM 值 10-100%
001FH	REG_IDI_R	0：MODE 功能关闭 1：MODE 功能打开	MODE 工作模式调节

开机命令功能码： 03H/06H

0030H	REG_DEVICEMODE	0：手动开机(此模式上电后必须写入系统启动命令) 1：自动开机(此模式，激光器上电后无需任何动作，系统自动完成开机过程)	设置开机模式 0-1 模式
0031H	REG_ON_OFF	0：关机 1：开机	开机命令
0032H	REG_SYS_STE	0：启动 1：启动完成 2：运行 3：报警 4：关机	查询系统状态

0033H			保留
0034H	REG_SYS_WORK_MODE	0:不加电流 1:加电流不开Q_SWITCH 2:加电流开Q_SWITCH	

串口指令对照说明：



激光器开机命令：01 06 00 31 00 01 +CRC 校验码
激光器关机命令：01 06 00 31 00 00 +CRC 校验码



手动操作开机模式：01 06 00 30 00 00 +CRCH CRCL 校验码
自动开机模式： 01 06 00 30 00 01 +CRCH CRCL 校验码
说明：如果设置为自动开机模式，激光器上电即进入预热、开机状态。

查询 LD 温度（0x0000）&倍频晶体温度（二倍频 0x0001，三倍频 0x0002）&环境温度(0x0003)&LT 温度(0x0004):

例如查询 LD 温度： 串口发送 01 03 00 00 00 01 +CRC 校验码，串口回读：01 03 02 温度参数 +CRC 校验码，回读的温度参数为 16 进制， 转化为十进制后再除以 10，则为实际温度参数。



电流参数设置：01 06 00 08 电流参数+CRC 校验码
（电流参数示例：如设置 10A 电流，则参数为 10*10=100，转化为 16 进制 0x0064,发送指令： 01 06 00 08 00 64 + CRC 校验码）。



频率参数设置： 01 06 00 09 频率参数+CRC 校验码
（频率参数示例：如设置 100KHz 频率，则参数为 100*100=10000，转化为 16 进制 0x2710，发送指令： 01 06 00 09 27 10 + CRC 校验码）。



PWM 参数设置：01 06 00 10 PWM 参数+CRC 校验码
PWM(%)”和“Power Ratio(us)”是对应关系，100%对应1us，其他百分比对应计算式为（1-(频率周期-0.4,再向下取整)）*百分比+(频率周期-0.4,再取整)，例如图中频率80KHz，周期为12.5us，则减去0.4us，再向下取整为12，当设置PWM为60%时,脉宽时间为（1-12）*60%+12=5.4us



打开 Diode：01 06 00 07 00 01 +CRC 校验码
关闭 Diode：01 06 00 07 00 00 +CRC 校验码



打开 QS_EXT：01 06 00 0C 00 01 +CRC 校验码
关闭 QS_EXT：01 06 00 0C 00 00 +CRC 校验码



打开 QS_ON：01 06 00 0B 00 01 +CRC 校验码
关闭 QS_ON：01 06 00 0B 00 00 +CRC 校验码



打开 Mode：01 06 00 1F 00 01 +CRC 校验码
关闭 Mode：01 06 00 1F 00 00 +CRC 校验码

CRC 校验码生成代码C#：

```
private static ushort[] crcTable = {
            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
        };
public static byte[] Crc16(byte[] data)
        {
            if (data == null)
                throw new ArgumentNullException("data");
            ushort crc = ushort.MaxValue;
            byte tableIndex;
            foreach (byte b in data)
            {
                tableIndex = (byte)(crc ^ b);
                crc >>= 8;
                crc ^= crcTable[tableIndex];
            }
            return BitConverter.GetBytes(crc);
        }
```