


# 字符流通讯协议

版本	日期	更新日志
V1.0	2021/10/20	添加版本号
V1.1	2022/05/10	1、添加扩展输入输出查询，输出控制查询； 2、添加输出控制失败错误信息；
V1.2	2022/08/26	增加读取波形 IO 值
V1.3	2022/09/30	C 选择打标指令支持选择多个；
V1.4	2022/10/21	红光预览指令支持选择多个；
V1.5	2022/11/01	增加立即执行指令
V1.6	2022/11/11	增加创建矩形和圆，增加修改矩形和圆属性
V1.7	2023/01/13	增加修改间隔延时距离和读取整个文档内容指令
V1.8	2023/02/20	增加弧形文本属性和坐标移动到中心动作
V1.9	2023/02/28	增加修改文字对齐方向属性
V1.10	2023/03/17	增加填充边距和重复次数属性
V1.11	2023/04/28	增加填充类型，交叉填充，启用填充属性 增加红光实时移动对象
V1.12	2023/05/11	增加扩展轴移动指令
V1.13	2023/08/25	增加扩展轴复位和获取状态指令
V1.14	2023/08/28	增加文件传输指令
V1.15	2023/09/15	增加使能/禁用打标操作指令
V1.16	2023/11/24	增加对象保持宽高比设置指令
V1.17	2023/12/11	增加读取当前编辑文档名称指令
V1.18	2023/12/18	增加读取触发器感应次数指令（仅飞行打标适用）
V1.19	2024/02/19	增加报警通知
V1.20	2024/03/05	增加清空通讯缓存内容
V1.21	2024/03/18	增加笔号螺旋标刻参数设置和读取 增加支持固定文本长度限制


1 协议说明 .....	4
2 命令类别 .....	5
2.1 测试/界面操作 .....	5
2.1.1 测试功能 .....	5
2.1.2 关闭提示界面 .....	5
2.2 文档操作 .....	5
2.2.1 调取文档 .....	5
2.2.2 创建文档 .....	5
2.2.3 保存文档 .....	6
2.2.4 获取文档列表 .....	6
2.2.5 获取当前打开文档名称 .....	6
2.3 标记操作 .....	6
2.3.1 创建标记 .....	6
2.3.2 删除标记 .....	6
2.3.3 清空所有标记 .....	7
2.3.4 清空所有标记通讯缓存内容 .....	7
2.3.5 修改标记内容 .....	7
2.3.6 读取标记内容 .....	10
2.3.7 标记动作 .....	10
2.3.8 获取标记列表 .....	11
2.4 打标笔号操作 .....	11
2.4.1 修改笔号参数 .....	11
2.4.2 读取笔号参数 .....	12
2.5 打标模式操作 .....	12
2.5.1 修改打标模式参数 .....	12
2.6 打标完成操作 .....	13
2.6.1 读取完成参数 .....	13
2.6.2 修改计划和完成数 .....	13
2.7 红光预览和打标 .....	13
2.7.1 开始打标 .....	13
2.7.2 开始红光预览 .....	14
2.7.3 开始红光预览指定大小位置 .....	14
2.7.4 移动振镜并控制红光点 .....	14
2.7.5 停止打标 .....	14
2.7.6 旋转标记打标 .....	14
2.7.7 偏移和旋转文档打标 .....	15
2.7.8 偏移和旋转振镜打标 .....	15
2.7.9 分组标签直接打标 .....	16
2.7.10 选择标记名直接打标 .....	16
2.7.11 打标完成输出 .....	16
2.7.12 打标中断输出 .....	16
2.7.13 查询打标状态 .....	16
2.7.14 开始出光 .....	17
2.7.15 停止出光 .....	17
2.7.16 实时红光移动当前对象 .....	17
2.7.17 软触发打标 .....	17
2.7.18 使能/禁用打标操作 .....	17
2.8 IO 配置 .....	18
2.8.1 输入口 .....	18
2.8.2 输出口 .....	18
2.9 错误信息 .....	18

2.9.1 错误输出 .....	18
2.9.2 错误输入 .....	19
2.10 其他信息 .....	19
2.10.1 外部 IO 触发输出指令 .....	19
2.10.2 立即执行指令 .....	19
2.10.3 读取触发感应次数 .....	19
2.10.4 报警通知 .....	20
2.11 板卡唯一 ID .....	20
2.12 扩展轴控制 .....	20
3 文件传输 .....	22
3.1 准备传输 .....	22
3.2 数据传输 .....	22
4 例子 .....	23

# 1 协议说明

配置界面为 。

①如内容不含中文，请使用【Ascii 命令字符】，配置界面为 ；

②如内容含中文，请使用【Unicode 中文命令字符】，配置界面为 ；  
Unicode 中文举例：

全部字符用 Unicode 双字节发送。

英文字符：第一个字节值为 0x00，第二个字节值为对应 Ascii 值（比如字符 A 对应 Ascii 值为 0x41）。

中文字符：查找中文 Unicode 值，比如中文‘中’的 Unicode 值为 0x4E2D，第一个字节值为 0x4E，第二个字节值为 0x2D。

Unicode 中文例子：

外部十六进制发送：

```
00 3C 00 45 6D 4B 8B D5 65 87 5B 57 00 3E 00 3C 00 4C 00 44 00 65 00 6D 00 6F 00 3E 00 3C  
00 44 65 87 67 2C 00 2C 8B F7 6C 42 4E 2D 65 87 00 3E 00 3C 00 58 00 3E
```

激光机十六进制返回：

```
00 3C 00 45 6D 4B 8B D5 65 87 5B 57 00 3E 00 3C 00 4C 00 44 00 65 00 6D 00 6F 00 3E 00 3C  
00 44 65 87 67 2C 00 2C 8B F7 6C 42 4E 2D 65 87 00 3E 00 3C 00 58 00 3E 00 3C 00 58 00 45  
00 3E
```

命令解释：

激光机接收到的字符内容为：<E 测试文字><LDemo><D 文本,请求中文><X>

激光机返回的字符内容为：<E 测试文字><LDemo><D 文本,请求中文><X><XE>

功能解释：激光机进入工作模式，测试返回<E 测试文字>，打开加载 Demo.lmf3 文档，修改标记名为文本的内容为请求中文，触发雕刻，雕刻完成。

**注意：**如果内容包含有英文右箭头>符号，请在前面加英文斜杠\符号，比如<Dtest,ABC\>>命令将修改标记名为 test 的内容为 ABC>。

## 2 命令类别

### 2.1 测试/界面操作

#### 2.1.1 测试功能

<E{测试文字}> 是测试指令，测试成功后返回指令。

例子：

发送：<Etest>

返回：<Etest>

接受正确的内容表示接线和配置都正确，可以进一步使用其他指令。

#### 2.1.2 关闭提示界面

<ECLOSE\_DLГ,{0/1}>

参数 0：关闭界面并取消；

参数 1：关闭界面并确定；

例子：

发送：<ECLOSE\_DLГ,1>

返回：<ECLOSE\_DLГ,1>

执行结果：关闭提示界面并确定。

## 2.2 文档操作

### 2.2.1 调取文档

<L {文件名}> 是加载激光模板文件（默认为文件夹根目录文件），成功加载后返回指令。

例子：

发送：<Lfile1>

返回：<Lfile1>

执行结果：调用系统预保存好文件，名为：file1.lmf3。

如果调用子目录里面的文件夹，命令格式为：<L{目录名}/{文件名}>。

### 2.2.2 创建文档

<LNEW,{文件名}>是创建激光模板文件，成功创建后返回指令。

例子：

发送：<LNEW,CommTest>

返回：<LNEW,CommTest>

执行结果：创建新文件，文件名为 CommTest.lmf3。

## 2.2.3 保存文档

<LSAVE>是保存当前编辑的模板文件，成功保存后返回指令。

例子：

发送：<LSAVE>

返回：<LSAVE>

执行结果：保存当前编辑的文件。

## 2.2.4 获取文档列表

<L>是读取前 20 个文档名称指令。

例子：

发送：<L>

返回：<L,001.lmf3,002.lmf3>

## 2.2.5 获取当前打开文档名称

<?L>是读取当前打开的文档名称指令。

例子：

发送：<?L>

返回：<?L,test.lmf3>

## 2.3 标记操作

### 2.3.1 创建标记

<DNEW,{标记类型},{名称}>是创建标记指令，成功创建后返回指令。

标记类型有：

| 标记类型     | 描述  |
|----------|-----|
| TEXT     | 文本  |
| BARCODE  | 条形码 |
| VECTOR   | 矢量图 |
| IMAGE    | 位图  |
| LINE     | 直线  |
| RECTANGL | 矩形  |
| CIRCLE   | 圆   |

**注意：**请勿将标记名称设置为 NEW。

例子：

发送：<DNEW,TEXT,tx>

返回：<DNEW,TEXT,tx>

执行结果：创建文本标记，标记名称为 tx。

### 2.3.2 删除标记

<DDEL,{名称}>是删除标记指令，成功删除后返回指令。

例子:

发送: <DDEL,tx>

返回: <DDEL,tx>

执行结果: 删除名称为 tx 的标记。

### 2.3.3 清空所有标记

<DCLEAR>是清空当前文档所有标记指令, 成功清空后返回指令。

例子:

发送: <DCLEAR>

返回: <DCLEAR>

执行结果: 清空当前文档所有标记。

### 2.3.4 清空所有标记通讯缓存内容

<DCLEAR\_DATA>是清空当前文档所有标记缓存内容指令, 成功清空后返回指令。

例子:

发送: <DCLEAR\_DATA>

返回: <DCLEAR\_DATA>

执行结果: 清空当前文档所有标记的缓存内容, 可用在缓存打标功能模块, 如通讯缓存打标模块等。

### 2.3.5 修改标记内容

①<D{名称},{文字}> 是修改文字指令, 成功修改后返回指令。

- 如果是文本/组合文本/条形码, 则修改的是标记的文本内容;
- 如果是图片/矢量图, 则修改的是标记的文件名称;

例子:

发送: <Dtext1,1234>

返回: <Dtext1,1234>

执行结果: 修改标记名为 text1 的内容为 1234。

例子:

发送: <Dimg,apple.png>

返回: <Dimg,apple.png>

执行结果: 修改标记名为 img 的图片文件名为 apple.png。

②<D{名称},{属性类型},{数值}>是修改标记属性指令, 成功修改后返回指令。

属性类型有:

| 属性类型             | 描述                     |
|------------------|------------------------|
| PENNO            | 标记的笔号, 范围为 0-20        |
| WIDTH            | 标记的宽度, 单位为 mm          |
| HEIGHT           | 标记的高度, 单位为 mm          |
| WIDTH_KEEP_RATIO | 标记的宽度, 单位为 mm, 对应自动按比例 |

|                   |   |
|-------------------|---|
|                   | 修改标记的高度   |
| HEIGHT_KEEP_RATIO | 标记的高度，单位为 mm，对应自动按比例修改标记的宽度   |
| X                 | 标记的 X 坐标（默认左上角），单位为 mm  |
| Y                 | 标记的 Y 坐标（默认左上角），单位为 mm  |
| POSR              | 标记的坐标参考基准<br>0=左上角<br>1=顶端中心<br>2=右上角<br>3=左端中心<br>4=中心<br>5=右端中心<br>6=左下角<br>7=底端中心<br>8=右小角 |
| FILL_SPACE        | 标记的填充间距，单位为 mm  |
| FILL_ANGLE        | 标记的填充角度，单位为度  |
| FILL_MARGIN       | 标记的填充边距，单位为 mm  |
| FILL_REPEAT       | 标记的填充重复次数   |
| FILL_TYPE         | 标记的填充类型<br>0=单向<br>1=双向<br>2=弓形<br>3=优化弓形<br>4=回形   |
| FILL_CROSS        | 标记的交叉填充<br>0=不启用<br>1=启用  |
| LINE_X1           | 直线标记点 1 的 X 坐标，单位为 mm   |
| LINE_Y1           | 直线标记点 1 的 Y 坐标，单位为 mm   |
| LINE_X2           | 直线标记点 2 的 X 坐标，单位为 mm   |
| LINE_Y2           | 直线标记点 2 的 Y 坐标，单位为 mm   |
| TEXT_SPACE        | 标记的文字间距，单位为 mm  |
| TEXT_CHAR_WIDTH   | 标记的文字字符宽度，单位为 mm  |
| TEXT_CHAR_HEIGHT  | 标记的文字字符高度，单位为 mm  |
| TEXT_ENFONT       | 标记的英文字体名  |
| TEXT_CNFONT       | 标记的中文字体名  |
| TEXT_ALIGN_DIR    | 标记的文字对齐方向<br>0=左端对齐<br>1=右端对齐<br>2=顶端对齐<br>3=底端对齐<br>4=水平中心对齐<br>5=竖直中心对齐                     |
| TEXT_ARC          | 标记的弧形文本<br>0=不启用<br>1=启用  |
| TEXT_ARC_DIAMETER | 弧形文本的直径，单位为 mm  |
| TEXT_ARC_START_AG | 弧形文本的起始角度   |
| TEXT_ARC_CENTER_X | 弧形文本的圆心 X 坐标，单位为 mm   |



|                        |   |
|------------------------|---|
| TEXT_ARC_CENTER_Y      | 弧形文本的圆心 Y 坐标，单位为 mm   |
| TEXT_ARC_SWEEP_AG      | 弧形文本的扫描角度   |
| BARCODE_TYPE           | 条码标记的类型<br>Code128 = 0<br>QRCode = 1<br>Code39 = 2<br>Code93 = 3<br>Code11 = 4<br>CodaBar = 5<br>C25Matrix = 6<br>C25Inter = 7<br>ExtendCode39 = 8<br>EAN128 = 9<br>EAN8 = 10<br>EAN13 = 11<br>UPCA = 12<br>UPCE = 13<br>ISBN = 14<br>PDF417 = 15<br>DataMatrix = 16<br>DataMatrixGS1 = 17<br>Code128A = 18<br>Code128B = 19<br>Code128C = 20 |
| BARCODE_SHOWTEXT       | 条码标记是否显示文字<br>隐藏 = 0<br>显示 = 1  |
| ENTITY_CLOSE_MARK      | 标记是否关闭打标<br>关闭打标 = 1<br>启用打标 = 0  |
| CIRCLE_DIRECTION       | 圆形方向<br>逆时针 = 0<br>顺时针 = 1  |
| CIRCLE_START_ANGLE     | 圆形开始角度  |
| CIRCLE_SWEEP_ANGLE     | 圆形扫描角度  |
| CIRCLE_SHOW_ARC        | 圆形仅显示弧线<br>是 = 1<br>否 = 0   |
| RECT_START_OFFSET      | 矩形开始位置偏移  |
| RECT_COINCIDE_LEN      | 矩形结束点重合长度   |
| RECT_ALL_ROUND_PERCENT | 矩形启用全部圆角，圆角计算方式为百分比，全部圆角百分比为此数值   |
| RECT_ALL_ROUND_RADIUS  | 矩形启用全部圆角，圆角计算方式为半径，全部圆角半径为此数值   |

例子：

发送：

<Dtext1,WIDTH,20.0><Dtext1,HEIGHT,7.0><Dtext1,POSR,0><Dtext1,X,-10.0><Dtext1,Y,3.5>

返回：

<Dtext1,WIDTH,20.0><Dtext1,HEIGHT,7.0><Dtext1,POSR,0><Dtext1,X,-10.0><Dtext1,Y,3.5>

返回结果：修改名称为 text1 的标记，宽度设为 20.0mm，高度设为 7.0mm，坐标参考基准设为左上角，X 坐标设为-10.0mm，Y 坐标设为 3.5mm。

## 2.3.6 读取标记内容

①<?{名称}>是读取文字指令，成功后返回读取的内容。

- 如果是文本/组合文本/条形码，则读取的是标记的文本内容；
- 如果是图片/矢量图，则读取的是标记的文件名称；

例子：

发送：<?text1>

返回：<?text1,1234>

执行结果：读取并返回标记名为 text1 的内容。

例子：

发送：<?img>

返回：<?img,apple.png>

执行结果：读取并返回标记名为 img 的当前图片名称为 apple.png。

②<?{名称},{属性类型}>是读取标记属性指令。

<?{名称},{属性类型},{数值}>是读取返回结果指令。

属性类型参考【2.3.3 修改标记内容】。

例子：

发送：<?text1,WIDTH><?text1,HEIGHT><?text1,X><?text1,Y>

返回：<?text1,WIDTH,20.0><?text1,HEIGHT,7.5><?text1,X,-10.0><?text1,Y,3.5>

返回结果：读取名称为 text1 的标记，宽度为 20.0mm，高度为 7.5mm，X 坐标为-10.0mm，Y 坐标为 3.5mm。

③<?CONTENT>是读取整个文档内容，多个内容用英文逗号分隔开。

返回格式为<?{文档名称},{对象内容},{对象内容},...>。

例子：

发送：<?CONTENT>

返回：<?Demo.lmf3,text1,text2,text3>

返回结果：当前打开的文档为 Demo.lmf3，文档含 3 个对象，对象内容分别为 text1，text2 和 text3。

## 2.3.7 标记动作

<D{名称},{动作类型},{数值}>是修改标记指令，成功修改后返回指令。

动作类型有：

| 动作类型        | 描述                             |
|-------------|--------------------------------|
| ROTATE      | 标记以中心为基准进行旋转，正值为顺时针旋转，负值为逆时针旋转 |
| MOVE_CENTER | 标记以中心为基准进行，移动到视图中心，数值需设置为 0    |

例子：

发送：<Dtext1,ROTATE,90.0>

返回：<Dtext1,ROTATE,90.0>

返回结果：将名称为 text1 的标记，顺时针旋转 90 度。

## 2.3.8 获取标记列表

<D,{标记类型}>是获取标记对象列表，成功返回文档中所有指定标记类型的对象名称。

<D>返回所有标记对象名称。

标记类型选项可见【2.3.1 创建标记】。

例子：

发送：<D>

返回：<D,text1,img1,barcode1>

返回结果：文档中总共有 3 个标记对象，名称分别为 text1，img1 和 barcode1。

发送：<D,IMAGE>

返回：<D,IMAGE,img1,img2,img3>

返回结果：文档中总共有 3 个位图对象，名称分别为 img1，img2 和 img3。

## 2.4 打标笔号操作

### 2.4.1 修改笔号参数

<D{功能名称},{数值}>是修改**第一个笔号**打标参数指令，成功修改后返回指令。

固定功能名称有：

| 属性类型              | 描述                                    |
|-------------------|---------------------------------------|
| SPEED             | 笔号的打标速度，单位为 mm/s                      |
| POWER             | 笔号的功率，单位为%                            |
| FREQ              | 笔号的频率，单位为 kHz                         |
| MARKLOOP          | 笔号的打标次数                               |
| OPEN_LIGHT_DELAY  | 笔号的开光延时，单位为 us                        |
| CLOSE_LIGHT_DELAY | 笔号的关光延时，单位为 us                        |
| END_DELAY         | 笔号的结束延时，单位为 us                        |
| CORNER_DELAY      | 笔号的拐角延时，单位为 us                        |
| JUMP_SPEED        | 笔号的跳转延时，单位为 mm/s                      |
| JUMP_MIN_DELAY    | 笔号的最小跳转延时，单位为 us                      |
| JUMP_MAX_DELAY    | 笔号的最大跳转延时，单位为 us                      |
| JUMP_MAX_LEN      | 笔号的最大跳转长度，单位为 mm                      |
| DOT_MODE          | 笔号的打点模式<br>时间 = 0<br>脉冲 = 1<br>菱形 = 2 |
| DOT_DELAY         | 笔号的打点时间（仅打点模式为时间或菱形时有效），单位为 us        |
| DOT_PULSE         | 笔号的打点脉冲个数（仅打点模式为脉冲时有效）                |
| DOT_SIZE          | 笔号的打点大小（仅打点模式为菱形时有效），单位为 mm           |
| PULSE_WIDTH       | 笔号的脉冲宽度，单位为 us                        |
| WAVE_IO           | 笔号的波形 IO 值，范围 0-31                    |
| SPIRAL_EN         | 笔号的启用螺旋标刻(抖动)                         |

|                 |                  |
|-----------------|------------------|
|                 | 启用= 1<br>不启用=0   |
| SPIRAL_DIAMETER | 笔号的螺旋标刻直径，单位为 mm |
| SPIRAL_DIST     | 笔号的螺旋标刻螺距，单位为 mm |

**注意：**请勿将标记名称设置为固定功能名称。

例子：

发送：<DSPEED,2000><DPOWER,80><DFREQ,20>

返回：<DSPEED,2000><DPOWER,80><DFREQ,20>

执行结果：修改第一个笔号的打标速度为 2000mm/s，功率为 80%，频率为 20kHz。

<D{功能名称},{笔号},{数值}>是修改**指定笔号**打标参数指令，成功修改后返回指令。笔号范围为 0-20。

例子 1：

发送：<DSPEED,0,2000><DPOWER,1,80><DFREQ,2,20>

返回：<DSPEED,0,2000><DPOWER,1,80><DFREQ,2,20>

执行结果：修改笔号 0 的打标速度为 2000mm/s，修改笔号 1 的功率为 80%，修改笔号 2 的频率为 20kHz。

## 2.4.2 读取笔号参数

<?{功能名称}>是读取**第一个笔号**的打标参数指令。

<?{功能名称},{数值}>是返回**第一个笔号**的打标参数结果指令。

固定功能名称参考【2.4.1 修改笔号参数】。

例子：

发送：<?SPEED><?POWER><?FREQ>

返回：<?SPEED,2000><?POWER,80><?FREQ,20>

执行结果：返回第一个笔号的打标速度为 2000mm/s，功率为 80%，频率为 20kHz。

<?{功能名称},{笔号}>是读取**指定笔号**的打标参数指令。笔号范围为 0-20。

例子 1：

发送：<?SPEED,0><?POWER,1><?FREQ,2>

返回：<?SPEED,0,2000><?POWER,1,80><?FREQ,2,20>

执行结果：返回笔号 0 的打标速度为 2000mm/s，笔号 1 的功率为 80%，笔号 2 的频率为 20kHz。

## 2.5 打标模式操作

### 2.5.1 修改打标模式参数

<D{功能名称},{数值}>是修改打标模式指令，成功修改后返回指令。

固定功能名称有：

| 属性类型      | 描述                              |
|-----------|---------------------------------|
| MARK_MODE | 文档的打标模式<br>静态：0<br>飞行：1<br>管线：2 |

|                          |                       |
|--------------------------|-----------------------|
| MARK_START_DIST_DELAY    | 飞行/管线模式的开始延时长度(mm)    |
| MARK_INTERVAL_DIST_DELAY | 飞行/管线连续模式下的间隔延时距离(mm) |

例子:

发送: <DMARK\_MODE,1><DMARK\_START\_DIST\_DELAY,100>

返回: <DMARK\_MODE,1><DMARK\_START\_DIST\_DELAY,100>

执行结果: 修改当前文档打标为飞行模式, 修改飞行模式的开始延时长度为 100mm。

## 2.6 打标完成操作

### 2.6.1 读取完成参数

<?{固定名称}>是读取参数指令。

<?{固定名称},{数值}>是读取返回结果指令。

固定功能名称有:

| 属性类型      | 描述            |
|-----------|---------------|
| MARKTIME  | 文档打标时间 (单位为秒) |
| MARKCOUNT | 文档已打标个数       |
| PLANCOUNT | 文档计划个数        |

**注意:** 请勿将标记名称设置为固定功能名称。

例子:

发送: <?MARKCOUNT><?PLANCOUNT><?MARKTIME>

返回: <?MARKCOUNT,100><?PLANCOUNT,1000><?MARKTIME,2.50>

执行结果: 返回当前文档已打标个数为 100, 计划个数为 1000, 当前打标时间为 2.50s。

### 2.6.2 修改计划和完成数

<D{功能名称},{数值}>是修改计划和完成数指令, 成功修改后返回指令。

固定功能名称有:

| 属性类型      | 描述       |
|-----------|----------|
| MARKCOUNT | 文档已打标个数  |
| PLANCOUNT | 文档计划打标个数 |

例子:

发送: <DPLANCOUNT,1000><DMARKCOUNT,0>

返回: <DPLANCOUNT,1000><DMARKCOUNT,0>

执行结果: 修改当前文档计划打标个数为 1000, 已打标个数为 0。

**注意:** 计划个数要大于等于已打标个数。

## 2.7 红光预览和打标

### 2.7.1 开始打标

<X> 是触发雕刻指令。

例子:

发送: <X>

返回: <X><XE>

执行结果: 开始雕刻后返回指令<X>, 雕刻完成后返回<XE>。

## 2.7.2 开始红光预览

<XP> 是触发预览指令。

例子:

发送: <XP>

返回: <XP>

执行结果: 开始红光预览, 预览启动后返回指令<XP>。

发送: <CTX1, TX2, TX3, XP>

返回: <CTX1, TX2, TX3, XP>

执行结果: 仅预览名称为 TX1, TX2 和 TX3 的标记。

## 2.7.3 开始红光预览指定大小位置

<XP,{宽度},{高度},{中心 X 位置},{中心 Y 位置}>, 触发预览指定位置大小红光, 参数单位为 mm。

例子:

发送: <XP,30.00,10.00,0.0,0.0>

返回: <XP>

执行结果: 开始红光预览, 预览的内容为矩形, 宽度为 30.00mm, 高度为 10.00mm, 矩形中心 X 位置为 0.00mm, 中心 Y 位置为 0.00mm。

## 2.7.4 移动振镜并控制红光点

<XP,{坐标 X},{坐标 Y},{是否显示红光点 Y/N}>, 用于控制振镜当前位置, 可选显示红光点 (Y=显示, N=不显示)。

例子:

发送: <XP,30.00,10.00,Y>

返回: <XP>

执行结果: 移动当前振镜位置到 X=30.00mm, Y=10.00mm 位置, 并显示红光点。

## 2.7.5 停止打标

<P> 是停止雕刻指令。

例子:

发送: <P>

返回: <P>

执行结果: 停止雕刻后返回指令<P>。

## 2.7.6 旋转标记打标

<T{名字},ROTATE,{参考点基准},{数值}>是旋转标记指令, 此指令仅在打标的时候起效果, 而且在打标停止/文档关闭/打标出现错误后自动清为 0。

参考点基准值说明：

- 0=左上角
- 1=顶端中心
- 2=右上角
- 3=左端中心
- 4=中心
- 5=右端中心
- 6=左下角
- 7=底端中心
- 8=右小角

例子：

发送：<Ttext1,ROTATE,4,90.0><X>

返回：<Ttext1,ROTATE,4,90.0><X><XE>

执行结果：将名为 text1 的标记以中心位置旋转 90 度，并进行打标。

## 2.7.7 偏移和旋转文档打标

<T>是偏移和旋转文档（以坐标中心为基准）指令，一般用于视觉定位打标。此指令仅在打标的时候起效果，而且在打标停止/文档关闭/打标出现错误后自动清为 0。

例子：

发送：<T,6,1.0,2.0,3.0,4.0,5.0,6.0><X>

返回：<T,6,1.0,2.0,3.0,4.0,5.0,6.0><X><XE>

执行结果：打标两次，第一次打标水平偏移 1.0mm，垂直偏移 2.0mm，旋转 3.0 度后的文档，第二次打标水平偏移 4.0mm，垂直偏移 5.0mm，旋转 6.0 度的文档。

指令解释：

T 解析为偏移和旋转指令

6 表示坐标值个数，每 3 个坐标值对应文档偏移和旋转一次，在这里总共两次

1.0 表示在第一次中水平偏移 1.0mm

2.0 表示在第一次中垂直偏移 2.0mm

3.0 表示在第一次中旋转 3.0 度

4.0 表示在第二次中水平偏移 4.0mm

5.0 表示在第二次中垂直偏移 5.0mm

6.0 表示在第二次中旋转 6.0 度

<X>解析为开始打标，在这里表示文档打标两次，先打标第一次偏移和旋转后的文档，然后打标第二次偏移和旋转后的文档，以此类推。打标完成后返回 XE。

## 2.7.8 偏移和旋转振镜打标

<T,MIRROR,偏移 X,偏移 Y,角度>是偏移和旋转振镜指令。收到此指令后，将偏移和角度参数应用到激光校正，打标完成后，自动恢复到原来校正。

例子：

发送：<T,MIRROR,-1,-2,0><X>

接收：<T,MIRROR,-1,-2,0><X><XE>

执行结果：水平偏移-1mm，垂直偏移-2mm，旋转 0 度校正激光并打标，打标完成后系统自动恢复原来校正。

## 2.7.9 分组标签直接打标

<S{0},X> S0 表示选中所有为 0 序号的标签，X 表示立刻打标。

例子：

发送：<S0,X>

返回：<S0,X><XE>

执行结果：标签控制打标组号为 0 的所有标签打标立触发刻发一次打标，打标完成后返回 XE。

## 2.7.10 选择标记名直接打标

<C{名称},X>是根据名称选中标记并打标。

例子：

发送：<CAPPLE,X>

返回：<CAPPLE,X><XE>

执行结果：仅打标名称为 APPLE 的标记，打标完成后返回 XE。

发送：<CTX1,CTX2,CTX3,X>

返回：<CTX1,CTX2,CTX3,X><XE>

执行结果：仅打标名称为 TX1，TX2 和 TX3 的标记，打标完成后返回 XE。

## 2.7.11 打标完成输出

①<XE>，<XE>为默认打标完成输出指令；

②<XE{名称},{打标内容}>，此命令输出打标完成的内容，软件需要配置后才可使用。

## 2.7.12 打标中断输出

<XT>为打标中断指令。

下列两种情况会发出此命令：

①单次模式下，在打标过程中，点击/IO 触发停止打标；

②连续/触发器/飞行模式下，在打标过程中，点击/IO 触发停止打标或者遇到序列号/数据库/文档计划个数已打标完等错误；

## 2.7.13 查询打标状态

<?X>是查询打标状态指令；

| 返回值 | 状态      |
|-----|---------|
| 0   | 未工作状态   |
| 1   | 正在打标中   |
| 2   | 正在红光预览中 |

例子：

发送：<?X>

返回：<?X,1>

执行结果：返回值为 1，表示当前正在打标工作中。



## 2.7.14 开始出光

<XL,{功率},{频率},{持续时间}>是开始出光指令，用于持续无间断出光打点，直到{持续时间}。

如果{持续时间}小于等于 0，则保持一直出光不停止。  
功率单位为%，频率单位为 kHz，持续时间单位为 ms。

例子 1:

发送: <XL,100,20000,-1>

返回: <XL,100,20000,-1>

执行结果: 以功率 100%，频率 20kHz，持续开光，直到接收【停止出光】指令后停止。

例子 2:

发送: <XL,100,20000,1000>

返回: <XL,100,20000,1000>

执行结果: 以功率 100%，频率 20kHz，持续开光，直到 1000 毫秒后停止出光。

注意: 即使 1000 毫秒后停止出光，但仍需要用【停止出光】指令才能退出。

## 2.7.15 停止出光

<PL>是停止出光指令，仅配合【开始出光】指令使用。

## 2.7.16 实时红光移动当前对象

<XPM,{偏移 X},{偏移 Y}>，红光预览中移动当前对象，并重新红光预览。此指令仅在当前为红光预览情况下有效。

例子:

发送: <XP><XPM,1.00,1.00>

返回: <XP><XPM,1.00,1.00>

执行结果: 先执行红光预览，然后将预览内容往右移动 1.0mm，往上移动 1.0mm 然后再重新预览。

## 2.7.17 软触发打标

<XTG>为软触发打标，适用于以下情况:

①静态触发器打标;

②飞行触发器打标;

此命令需在打标状态下才能正常执行。

## 2.7.18 使能/禁用打标操作

<XEN,{使能/禁用}>，使能或者禁用打标操作（包括脚踏触发/界面点击开始按钮/界面按键快捷键 F2）。

例子:

发送: <XEN,0>

返回: <XEN,0>

执行结果: 禁用打标。

例子:

发送: <XEN,1>

返回: <XEN,1>

执行结果: 使能打标。

## 2.8 IO 配置

### 2.8.1 输入口

<?I>为查询输入口 IN1 至 IN15 高低电平命令, 0 代表该输入口为低电平, 代表该输入口为高电平。

例子:

发送: <?I>

返回: <?I,0,1,1,1,1,1,1,1,1,1,1,1,1,1>

执行结果: 输入口 IN1 对应端口为低电平, IN2 至 IN15 端口为高电平。

### 2.8.2 输出口

<O,端口号,电平信号,持续时间>为输出口控制命令, 支持板卡 OUT1 - OUT15 输出控制:

| 参数   | 说明                        |
|------|---------------------------|
| 端口号  | 1 - 15, 代表板卡 OUT1 - OUT15 |
| 电平信号 | 0 = 低电平, 1 = 高电平          |
| 持续时间 | 1 - 999999999, 电平保持输出时间   |

例子 1:

发送: <O,1,0,100>

成功返回: <O,1,0,100>

执行结果: 将板卡 OUT1 输出为低电平, 保持输出 100 毫秒。

如果端口号已被其他地方占用, 会出现输出失败, 见【错误输出】。

<?O>为查询输出口命令, 返回 OUT1 - OUT15 端口电平状态, 0 = 低电平, 1 = 高电平。

例子 2:

发送: <?O>

返回: <?O,1,0,0,0,0,0,0,0,0,0,0,0,0,0>

执行结果: 板卡 OUT1 当前为高电平, OUT2-OUT15 为低电平。

## 2.9 错误信息

### 2.9.1 错误输出

格式: <FAIL {n}... ..>, 错误代码为 n 的输出指令, 此命令为板卡主动输出。其中:

①代码 1, <FAIL1>, 表示打标发生重码;

②代码 2, <FAIL2>, 表示接收到无效指令, 比如指令格式错误;

③代码 3, <FAIL3>, 表示输出口控制的 OUT 口已被其他地方占用, 控制输出失败;

④代码 4, <FAIL4>, 表示配置底层服务通讯失败;

⑤代码 5, <FAIL5>, 表示飞行视觉序号匹配失败;

## 2.9.2 错误输入

格式: <FAIL,{内容文字提示},{输出端口号},{输出信号类型},{持续时间}>, 此命令为客户发送给板卡, 板卡收到后界面和 IO 输出提示。

**内容文字提示:** 如果是 Ascii 传输模式, 则仅支持英文提示; 如果用 Unicode 传输模式, 则可支持英文中文提示。

**输出端口号:** 范围 0-9, 如果是 0, 则无输出。如果是 1-9, 则对应 OUT1-OUT9 端口;

**输出信号类型:** 0=低电平, 1=高电平。

**持续时间:** 单位为秒。-1=界面一直保持和输出口一直输出, 直到界面关闭。

**注意:** 如果当前正在打标中, 则会停止当前打标后再执行此命令。

例子 1:

发送: <FAIL,出现错误,1,1,-1>

返回: <FAIL,出现错误,1,1,-1>

执行结果: 界面提示【出现错误】, 并在 OUT1 输出高电平, 当界面关闭后, OUT1 恢复低电平。

例子 2:

发送: <FAIL,出现错误,0,1,5>

返回: <FAIL,出现错误,0,1,5>

执行结果: 界面提示【出现错误】, 无输出口, 当显示 5 秒后, 界面自动关闭。

## 2.10 其他信息

### 2.10.1 外部 IO 触发输出指令

**注意:** 仅支持含 IO 输入口配置版本使用。

**功能:** 当外部 IO 触发后, 通讯发出此指令。

**格式:** <COMMIO>

### 2.10.2 立即执行指令

**字符串格式:** <\n\r>

**Ascii 十六进制格式:** 3C 0A 0D 3E

**Unicode 十六进制格式:** 00 3C 00 0A 00 0D 00 3E

**说明:** 在发送的指令最后面加上此命令, 当板卡接收到含此指令的内容时, 立即解析执行此指令前面内容, 不等待接收超时, 可节省一部分时间。

### 2.10.3 读取触发感应次数

**格式**<?TRIGGER\_COUNT>, 读取飞行打标状态下, 当前触发器感应次数。

**例子:**

发送: <?TRIGGER\_COUNT>

返回: <?TRIGGER\_COUNT,3>

## 2.10.4 报警通知

查询格式:

发送: <ALARM,{类型}>

返回: <ALARM,{类型},{报警 1},{报警 2},{报警 3},{报警 4}>

{类型}对应数值:

0=自定义报警

{报警 1/2/3/4}对应数值:

0=无报警状态

1=报警状态

例子:

发送: <ALARM,0>

返回: <ALARM,0,1,0,1,0>

执行结果: 当前自定义报警输入 1 和 3 处于报警状态。

如果软件界面勾选【自定义报警触发通知】，则当触发自定义报警时，软件将发送<ALARM,{类型},{报警 1},{报警 2},{报警 3},{报警 4}>内容。

## 2.11 板卡唯一 ID

格式:<?BOARD\_ID>, 读取板卡唯一 ID 指令。

例子:

发送: <?BOARD\_ID>

返回: <?BOARD\_ID,9496561E62ED1085>

## 2.12 扩展轴控制

固定功能名称有:

| 属性类型          | 描述                              |
|---------------|---------------------------------|
| AXIS_RESET    | 扩展轴复位                           |
| AXIS_POS      | 扩展轴当前位置                         |
| AXIS_STATUS   | 扩展轴当前状态<br>0 = 空闲<br>1 = 移动/复位中 |
| AXIS_REL_MOVE | 扩展轴相对移动                         |
| AXIS_ABS_MOVE | 扩展轴绝对移动                         |

**注意:** 使用上述命令前先复位扩展轴。

**注意:** 请勿将标记名称设置为固定功能名称。

<DAXIS\_RESET,{复位方式},{轴 X 是/否},{轴 Y 是/否},{轴 Z 是/否},{轴 R 是/否}>

例子:

<DAXIS\_RESET,0,1,1,0,0>是设置当前扩展轴 X 轴位置为打标零点位置指令，打标完成后将被重置，当前仅供【旋转轴】插件打标使用。

<DAXIS\_RESET,1,1,1,0,0>是复位扩展轴 X 和 Y 轴到硬件零点位置指令。

<DAXIS\_RESET,2,1,0,0,0>是复位扩展轴 X 和 Y 轴当前位置为零点位置指令。

<?AXIS\_POS>是读取扩展轴当前位置指令。

<?AXIS\_POS,{轴 X 位置},{轴 Y 位置},{轴 Z 位置},{轴 R 位置}>是读取返回结果指令。

例子:

发送: <?AXIS\_POS>

返回: <?AXIS\_POS,10.0,20.0,30.0,0.0>

执行结果: 返回当前扩展轴绝对位置, 轴 X 位置=10.0, 轴 Y 位置=20.0, 轴 Z 位置=30.0, 轴 R 位置=0.0。

<?AXIS\_STATUS>是读取扩展轴运动状态。

<?AXIS\_STATUS,{状态值}>是读取返回结果指令。

例子:

发送: <?AXIS\_STATUS>

返回: <?AXIS\_STATUS,1>

执行结果: 返回当前轴正在移动/复位中状态。

<D{固定名称},{轴 X 距离},{轴 Y 距离},{轴 Z 距离},{轴 R 距离}>是移动扩展轴指令。

例子 1:

发送: <DAXIS\_REL\_MOVE,1.0,-1.0,2.0,0.0>

返回: <DAXIS\_REL\_MOVE,1.0,-1.0,2.0,0.0>

执行结果: 扩展轴 X 相对移动 1.0 距离, 轴 Y 相对移动-1.0 距离, 轴 Z 相对移动 2.0 距离, 轴 R 不移动。

例子 2:

发送: <DAXIS\_ABS\_MOVE,10.0,20.0,30.0,0.0>

返回: <DAXIS\_ABS\_MOVE,10.0,20.0,30.0,0.0>

执行结果: 扩展轴 X 移动到绝对位置 10.0, 轴 Y 移动到绝对位置 20.0, 轴 Z 移动到绝对位置 30.0, 轴 R 移动到绝对位置 0.0。

## 3 文件传输

### 3.1 准备传输

<SFILE,Ready,{文件格式},{文件名},{文件大小}>

文件格式：支持位图 BMP、PNG、JPG、JPEG，矢量图 DXF、PLT、AI；

文件名：保存到控制卡的文件名称，不需要包含文件格式后缀；

文件大小：单位为字节(byte)；

例子：

发送：<SFILE,Ready,PNG,logo,7468>

返回：<SFILE,Ready,PNG,logo,7468>

执行结果：通知控制卡准备接收大小为 7468 字节的 PNG 文件数据，并保存文件名为 logo.png。

### 3.2 数据传输

【3.1 准备传输】指令结束后，板卡自动进入接收数据状态（默认接收超时为 5 秒），直到接收到指定大小字节后，返回接收结果指令。

返回：<SFILE,Data,ok>

执行结果：接收文件数据成功。

返回：<SFILE,Data,error>

执行结果：接收文件失败。

## 4 例子

范例

外部串口发送: <Etest><Lfile2><Dtext1,1234><Dtext2,9876><X>

激光机返回: <Etest><Lfile2><Dtext1,1234><Dtext2,9876><X><XE>

功能解释: 激光进入工作模式, 测试返回<Etest>, 加载文件 file2.lmf3, 修改标记名 text1 内容为 1234, 修改标记名 text2 内容为 9876, 触发雕刻, 雕刻完成。