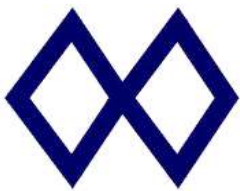
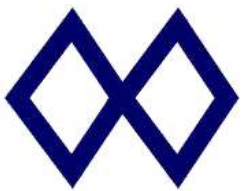


**WizPro** 系列烧写器二次  
开发说明  
V1.0



目录

1	操作说明:	3
1.1	需求提出:	3
1.2	命令行开发:	3
1.3	通过标准 APP 来实现更简洁的二次开发:	3
2	工程文件:	错误!未定义书签。
2.1	工程文件制作:	错误!未定义书签。
2.2	工程文件加载: 手动加载	错误!未定义书签。
2.3	工程文件加载: 命令行加载	4
3	数据返回说明	5
3.1	通讯方式:	5
3.2	数据包的结构:	5
3.3	数据的组成:	5
3.4	常用命令包:	7



## 1 操作说明:

### 1.1 需求提出:

- WizPro 系列烧写器可以支持二次开发，满足客户在不同情况下的需求和应用。为了简化二次开发得难度，我们将各种复杂得情况进行了一定得封装，开发工程师可以跳开不同芯片设置、烧写器的各种参数和配置，而将自己的主要工作放在自己擅长的方面，既提高了效率，简化了开发，又能避免各种繁琐的参数和重复的测试工作。我们提供 2 种不同的方式：基于纯命令行方式（每一个功能都通过调用命令行命令来实现，该方式需要提供独立的命令行软件，可能涉及到独立的 License 费用），其二是通过我们标准的 APP 加烧写器信息的输出监控来实现二次开发；

### 1.2 命令行开发:

- 具体见我司“WizPro 系列烧写器命令行参数使用说明”的相关文档，具体需求请联系我司相关人员；

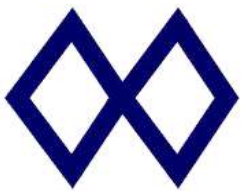
### 1.3 通过标准 APP 来实现更简洁的二次开发:

- 不同于命令行方式，该方式下，烧写器是工作与脱机模式，其分为 2 个独立的操作步骤：加载和下载烧写数据，烧写信息的获取和监控；
- 加载下载数据：有 2 种方式，一种是先做好工程文件，直接通过命令行调用下载，另一种则是标准操作，通过我们的标准软件来下载；

- 工程文件方式：有些型号还未支持工程文件，具体看实际的产品。该操作首先要求有准备好的工程文件，通过如下的命令行来加载下载数据，具体调用未：“[path]APP 名称 [path]工程文件名称 OFF（可选）”，其中“OFF”为下载后自动关闭调用的 APP，否则需要手工关闭。举例说明，若需要下载的工程文件名称是“PrjAAA.PMW”，APP 为 WizProGenST.exe，所在目录为：D:\MaxWizAPP，则完整的调用方式如下：


“D:\> D:\MaxWizAPP\WizProGenST.exe PrjAAA.PMW OFF”。执行该命令后，我们的 APP 则会自动完成一系列的操作，并将数据下载到烧写器中，退出 APP 后，会在 APP 所在目录生成一名为“ResDW.Txt”的文件，工程开发人员只要在他的 APP 中查询该文件的生成并解析文件的内容，即可了解下载操作的结果和情况；

- 标准下载，该方式就是直接打开我们的 APP，按照“选择芯片”-“加载数据”-“参数设置”-“下载”几个过程，将数据下载到烧写器，该方式，无需制作工程文件；
- 需要指出的是，在执行命令行下载指令前，用户的 APP 必须先关闭所打开的烧写器所对应的各 COM 口，因为 WizPro 系列 APP 在下载时需要使用各 COM 口；WizPro APP 会根据情况自动下载所搜索到的所有 WizPro 烧写器（通道）或者通过指定的 INI 配置文件来指定一个或几个对应的烧写器（通道）来实现数据下载；详细见 INI 文件的说明；
- 控制命令和监控信息是用户通过直接控制烧写器对应的 COM 口来实现，因此用户可利用此方式控制多个同样的烧写器来实现多路的异步操作；



## 1.4 工程文件加载：命令行加载

- 上位机可以使用命令行方式，将工程文件下载到烧录器
- 命令行格式：“上位机软件所在目录”+“空格”+“工程文件所在目录”+“空格”+参数
- 例如：
  - 上位机 WizProGenSTX.exe 存放在 D:\
  - 工程文件存放在 D:\test\
  - 则在命令行中输入 D:\WizProGenSTX.exe D:\test\ST32L030.pmw OFF 即可下载数据到烧录器中；

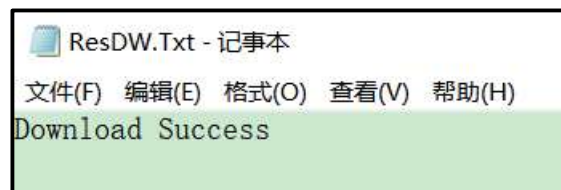
 命令提示符

```
Microsoft Windows [版本 10.0.19042.1526]
(c) Microsoft Corporation。保留所有权利。

C:\Users\11111>D:\WizProGenSTX.exe D:\test\ST32L030.pmw OFF
C:\Users\11111>_
```

※OFF 为下载完成后，上位机自动关闭

- 下载完成后，上位机所在目录会生成一个“ResDW.Txt”文件，查询此文件可获知下载结果；





## 2 烧写器监控命令说明

### 2.1 通讯方式:

- 烧写器采用 USB 转 UART 串口;
- UART 参数设定: 1M 波特率, 8-Bit Data, 无校验, 2-Bit Stop;

### 2.2 数据包的结构:

- 一个完整数据包为: 0xC0 + 数据 + 0xC0;
- 若数据是 0xC0, 则转换成: 0xDB+ 0xDC;
- 若数据是 0xDB, 则转换成: 0xDB+ 0xDD;

### 2.3 数据的组成:

#### 2.3.1 数据分类:

- 一类是控制烧写器得数据, 有应用程序发给烧写器;
- 一类是返回数据, 是烧写器对命令的响应或者自动向外输出;

#### 2.3.2 控制数据:

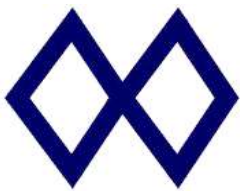
- 复位编程器: 0xC0+0x8A+'REST'+0xA8+0x02+0x00+0x00+0x00+0xCD+0x02+0xC0; 烧写器收到该命令后, 系统复位, 进入上电类似模式, 烧写器首先进行数据的自检, 以判断下载数据是否正常, 之后输出相关的信息和数据包, LED 指示相关状态; 用户 APP 可通过抓取其中的一个数据返回包来获取所需要的烧写器的各种基本信息, 该数据包说明如表 1 说明, 数据包以"0xFA"+"WDI"数据开始。
- 启动烧写命令: 0xC0+0xF9+" APRG" +0x9F+0x00+0x00+0xE3+0x4D+0xC0;
- 检查烧写器状态: 0xC0+0xF6+ "CKPG" +0x6F+0x00+0x00+0x7A+0x3A+0xC0;

#### 2.3.3 烧写器返回数据: 详见表 2.

- 烧写器过程中状态数据: 固定长度为 9 个字节:

0xFD + 状态码 1 (1B) +状态码 2 (1B) +数据(4B) + Check Sum (1B);

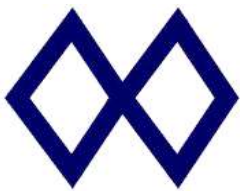
- 其中 CheckSum 为前面 8 个字节的累加和的补码, 也就是该 9 个字节累加和为 0;



符表：返回数据说明：

- 表 1：复位/Power On 时的返回数据包说明：

Offset	数据	说明
0-3	0xFA+“WDI”	数据包的识别 ID
4-5	数据 ID	固定为字符 “MW”
6-9	保留数据	
10	烧写器状态数据	Bit0: 0: 下载数据校验 OK/ 1: 下载数据校验错误
11	保留数据	
12-13	芯片主 Flash 大小(KB)	以 KB 为单位
14-15	芯片 EEP 数据大小	以字节为单位，若没有该数据，此数值为 0
16-19	保留数据	
20-21	Main Flash CRC16 Sum	芯片主数据区数据的 CRC16 校验和，内部使用
22-23	Main Flash ACC16 Sum	芯片主数据区数据的 ACC16 累加校验和，内部使用
20-21	EEP Data CRC16 Sum	芯片 EEP 数据（若有）CRC16 校验和，内部使用
20-21	EEP Data ACC16 Sum	芯片 EEP 数据（若有）ACC16 校验和，内部使用
22-25	保留数据	
26-83	保留数据	
22-83	保留数据	
84-99	Chip Name	芯片名称，最大 16 个字节，不够填数值 0x00
100-131	File Name	数据文件名称，最长 48 字节，不够填数值 0x00
132-n	保留数据	



- 表 2: 检查和状态数据说明: 该数据是自动输出, 用户 APP 只要去监控该输出即可。

状态码 1	状态码 2	数据	说明
0x10	0x00	4 字节数据, 忽略	进入烧写模式, 当前芯片 Page 数据
0x20	0x00	4 字节数据, 忽略	执行芯片查空
0x30	0x00	Current Page + Total page	执行芯片擦除
0x40	0x00	Current Page(2B) + Total page(2B)	写芯片主 Flash 数据区
0x50	0x00	Current Page(2B) + Total page(2B)	写芯片 EEP 或用户数据区
0x70	0x00	Current Page(2B) + Total page(2B)	校验主数据区数据
0xF0	0x06	4 字节返回数据, CRC16+ACC16	烧写完成, 成功
0xF0	错误代码	4 字节数据, 忽略	烧写完成, 失败, 状态 2 为错误代码
0xF1	0x06	4 字节返回数据, ACC32 或 CRC32, 低位在前	烧写完成, 成功
0xF1	错误代码	4 字节数据, 忽略	烧写完成, 失败, 状态 2 为错误代码
0xFA	0x01	0x57,0x50,0x4B,0x4F (WPKO)	编程器上电正常, 可以烧写
0xFA	0x01	0x57,0x50,0x4E,0x47 (WPGN)	编程器上电错误, 数据异常, 不能烧写
0xFA	0x03	0x43,0x45,0x52,0x52 (CERR)	Command Error.

注: “0xFA+数据”为检查烧写器状态命令返回数据包;

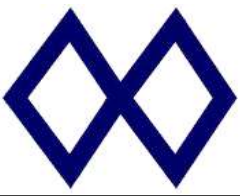
MD5 校验码输出:

FB F0 10 00 MD5 Data(Total 16 个字节), 例如: MD5=EA4ED1F73D865CA3E9721AA4F0D1CBB8, 输出如下:  
 FB F0 10 00 EA 4E D1 F7 3D 86 5C A3 E9 72 1A A4 F0 D1 C B B8 (16 进制数据)。

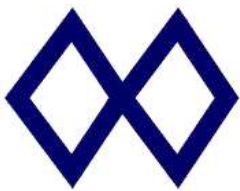
- INI 文件说明:

- 该文件名称固定为” ConfigWpg.ini”, 基本内容如下:

```
[MaxWizGxCFG]
[CHPG CFG]
EnPG1=1 // 开启通道 1
EnPG2=1
[Connection]
TotDev=2 // 总共 2 通道, 可根据实际需要调整
DEV1=COM3 // 通道 1 串口编号, 对应总通道数
DEV2=COM4 // 通道 2 串口编号
[END]
```







示例说明:

- 复位编程器: 0xC0+0x8A+'REST'+0xA8+0x02+0x00+0x00+0x00+0xCD+0x02+0xC0;烧录器将复位自校验, 并返回结果:
  - 编程器上电正常, 可以烧写;
  - 编程器上电错误, 数据异常, 不能烧写;

```
> 14:23:33 0xFD 0xFA 0x01 0x57 0x50 0x4B 0x4F 0xC7
> 14:23:33 Programmer is ready to Start!
> Current Programmer Information :
Current ROM Size is : 128KB.
Current Flash CRC Sum : 0x071A.
Current Flash ACC Sum : 0xA4EA.
Current Chip Name is : STM32F072xB
Current File Name is : STM32F2xxx.hex
No S/N is Used!
```

图: 复位自校验正常

```
> 14:25:05 0xFD 0xFA 0x01 0x57 0x50 0x47 0x4E 0xCC
> 14:25:05 Programmer has Error!
```

图: 复位自校验失败报错

- 启动烧写: 0xC0+0xF9+'APRG'+0x9F+0x00+0x00+0xE3+0x4D+0xC0; 烧录器将启动一轮烧写, 并返回状态信息, 见表 2, 其中状态码 0x10 数据包可用于检查判断启动命令是否发送成功:
  - 进入烧写模式, 初始化接口并执行一系列烧写操作;
  - 烧写 OK, 返回成功信息;
  - 烧写失败, 返回失败信息和错误代码;

```
> 11:02:51 0xFD 0xF0 0x06 0xEA 0xA4 0x1A 0x07 0x5E
> 11:02:51 Programming Completed : Succeeded !Chip ACC16=0x071A,CRC16=0xA4EA.
```

图: 烧录成功

```
> 14:18:42 0xFD 0xF0 0xE0 0x00 0x00 0x00 0x00 0x33
> 14:18:42 Programming Completed : Failed : Error=0xE0!
```

图: 烧录失败

- Check 烧写器: 0xC0+0xF6+'CKPG'+0x6F+0x00+0x00+0x7A+0x3A+0xC0; 将返回烧写器状态返回
  - 编程器上电正常, 可以烧写;
  - 编程器上电错误, 数据异常, 不能烧写;