

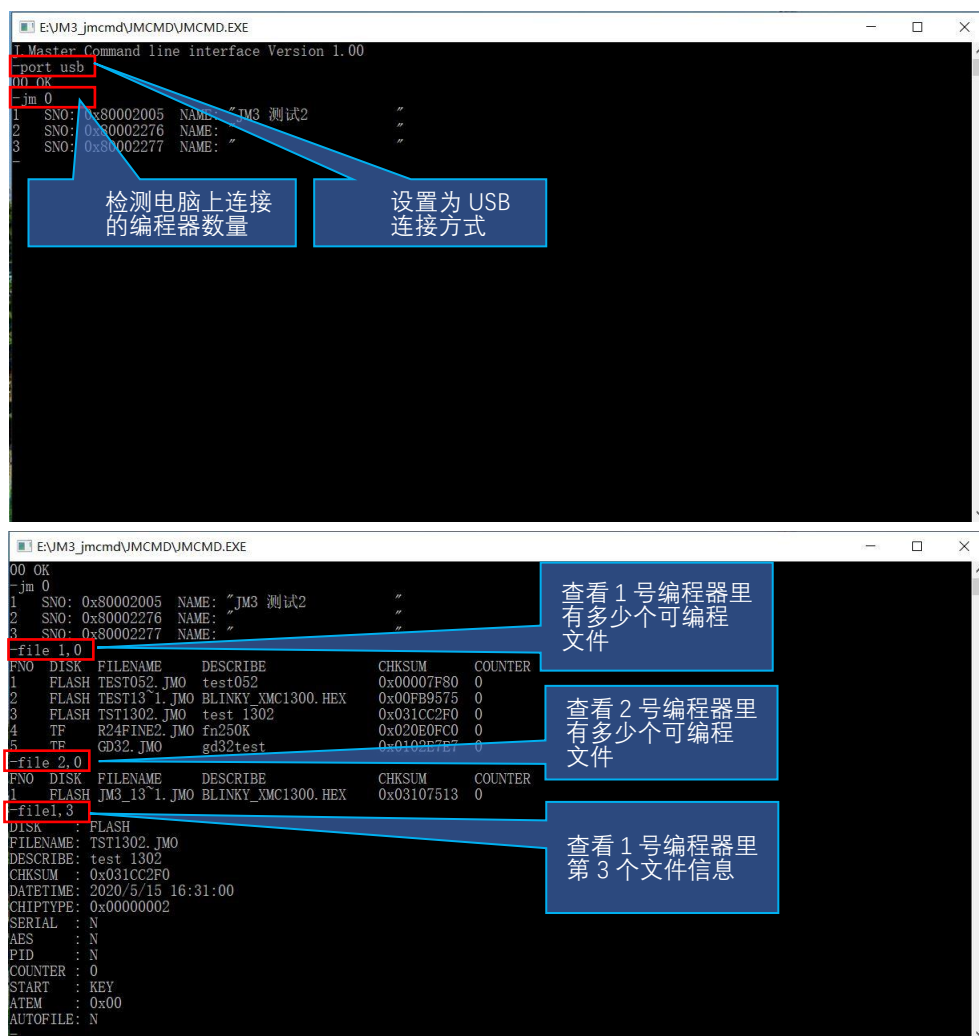
JMCMD 使用及 DLL 使用方法说明

伟福 J-MasterIII 编程器除了使用自带的编程控制界面，用户也可以使用伟福提供的 DLL 接口，自己写程序控制编程器，可以通过 USB 连接或 COM 连接两种方式控制，除了通过 USB 端口控制，还可以通过侧面 14 脚插座的第 13(TX)，14(RX)脚来控制，这就是 COM 控制方式。

TXD	NG2	NG1	OK1	BUSY1	START1	VCC
13	11	9	7	5	3	1
14	12	10	8	6	4	2
RXD	GND	GND	OK2	BUSY2	START2	VCC

侧面 14 脚管脚定义

JMCMD 是一款控制台界面（CMD 窗口）运行的 JM3 编程器控制软件，用户可能通过这个软件来了解编程器控制方法，用户如果要编写自己的 JM3 控制软件，也可以通过它了解第三方开发用到的 DLL 接口。JMCMD 使用的 JMO 必须在 JM3 上，如果不在，先用 DOWNLOAD 命令下载到编程器中。JMO 文件要以“按键启动”方式保存。我们先用几条常用指令说明怎么进行编程操作，后面再详细介绍每条指令。



```

E:\JM3_jmcmd\JMCMDD\JMCMDD.EXE
DISK : FLASH
FILENAME: TST1302.JMO
DESCRIBE: test 1302
CHKSUM : 0x031CC2F0
DATETIME: 2020/5/15 16:31:00
CHIPTYPE: 0x00000002
SERIAL : N
AES : N
PID : N
COUNTER : 0
START : KEY
ATEM : 0x00
AUTOFILE: N
-openjmo 1,3
00 OK
-run 1,1
00 OK
-run 1,2
35 PROG CONNECT
-run 2,1
70 ERROR CONNECT
-closejmo 1,3
ERROR: 71 ERROR_PARAMETER
-closejmo
ERROR: 71 ERROR_PARAMETER
-closejmo
ERROR: 71 ERROR_PARAMETER
-closejmo 1
00 OK
    
```

打开 1 号编程器里第 3 个文件，准备编程操作

启动 1 号编程器端口 1 进行编程操作，成功编程

启动 1 号编程器端口 2 进行编程操作，但端口 2 没有接芯片显示出错信息

启动 2 号编程器端口 1 编程操作，编程器拔除，显示出错信息

关闭编程文件时参数错的信息

关闭 1 号编程器打开的编程文件

```

E:\JM3_jmcmd\JMCMDD\JMCMDD.EXE
-help
JM      0/JM_NO
PORT    USB/COM
BAUDRATE 115200/230400/460800/500000/1000000
CMD     "FILENAME"

LOGON   CREATE/APPEND, "FILENAME"
LOGOFF
WRITELOG "STRING"

JM      0/JM_NO
MODE    JM_NO
RESET   0/JM_NO
LASTERROR JM_NO

FID      0/JM_NO, TF/FLASH, "FILENAME"
OPENJMO  0/JM_NO, 0/F_NO
CLOSEJMO 0/JM_NO
FILE     JM_NO, 0/F_NO
DELETE   0/JM_NO, 0/F_NO
DOWNLOAD 0/JM_NO, TF/FLASH, "FILENAME"
FORMAT   0/JM_NO

NAME     JM_NO
SNO      JM_NO
SETNAME  0/JM_NO, "NAME"
SETAUTOFILE 0/JM_NO, 0/F_NO
SETPASSWORD 0/JM_NO, "PASSWORD"
VERSION  JM_NO

RUN      0/JM_NO, 0/1/2
START    0/JM_NO, 0/1/2
MESSAGE  JM_NO, 0/1/2
COUNTER  JM_NO
SERIAL   0/JM_NO, "STRING"HEX

POWER    0/JM_NO, VCC
READIO   JM_NO, 1/2, A
WRITEIO  0/JM_NO, 0/1/2, A, D

EXIT

更详细说明参见DLL使用说明
    
```

1.BAUDRATE

功能: 设置波特率 只在用串口(COM)连接编程器时有效。连接的缺省波特率为: 115200

参数: 115200/230400/460800/500000

返回: 状态信息(见附录)

2.CLOSEJMO JM_NO

功能: 关闭打开的 JMO 文件, 退出编程状态, 返回监控态

参数: JM_NO:编程器号(0 表示所有连接的编程器), 如果只连接一台编程器, 就是 1

返回: 状态信息(见附录)

3.CMD "FILENAME"

功能: 运行命令文件, 可以把多个编程命令保存在一个文件中, 用此命令一次执行。

参数: "FILENAME" FILENAME 文件中是要执行的命令行

返回: 状态信息(见附录)

4.COUNTER JM_NO

功能: 读取 JMO 文件的编程限定次数, 对当前打开的 JMO 文件有效。

参数: JM_NO:编程器号

返回: 当前所剩余的编程次数

5. DELETE JM_NO, F_NO/0

功能: 删除编程器中的 JMO 文件

参数: JM_NO: 编程器号(0 表示所有连接的编程器) F_NO: JMO 文件序号 或 0: 用 FID 命令查找出的 JMO 文件序号

返回: 状态信息(见附录)

6. DOWNLOAD JM_NO, TF/FLASH,"FILENAME"

功能: 下载 JMO 文件到编程器

参数: JM_NO: 编程器号(0 表示所有连接的编程器) TF/FLASH 指定要下载到的盘, TF 表示下载到 TF 卡, FLASH 表示下载到内部 FLASH

返回: 状态信息(见附录)

7. EXIT

功能: 退出命令行窗口 在退出前会自动 DISCONNECT 已连接的编程器并 LOGOFF 关闭 LOG 文件

参数: 无

返回: EXIT 前一条命令的状态信息

8. FID JM_NO, TF/FLASH,"FILENAME"

功能: 根据文件名查该文件的序号

参数: JM_NO: 编程器号(0 表示所有连接的编程器), TF/FLASH 指定要查找的盘, TF 表示 TF 卡, FLASH 表示内部 FLASH。"FILENAME"指令 JMO 文件名

返回: 文件的序号。在 OPENJMO/SETAUTOFILE/DELETE 命令中也可以使用序号 0 指定此文件

9. FILE JM_NO, 0/F_NO

功能: 读取编程器中的 JMO 文件信息

参数: JM_NO: 编程器号 0: 读取编程器中 JMO 的文件个数 1..N: 显示文件序号指定 JMO 的信息

返回: 参数为 0 时返回编程器中 JMO 的文件个数, 参数为 1..N 时返回文件序号指定 JMO 的信息

DISK : TF/FLASH JMO 文件是在 TF 卡还是内部 FLASH 盘

FILENAME : JMO 文件名

DESCRIBE : JMO 文件描述

CHKSUM : JMO 文件校验和

DATETIME : 创建 JMO 文件的日期、时间

CHIPTYPE : JMO 文件中的芯片型号

SERIAL : Y/N JMO 文件中是否设置了编程串号

AES : Y/N JMO 文件中是否设置了加密密码

PID : Y/N JMO 文件中是否限定了编程器号

COUNTER : 设定的限定编程次数 0 表示没有限定编程次数

START : KEY/AUTO/ATE 编程启动方式

ATEM : JMO 文件中设定的 ATE 接口有效电平

AUTOFILE : Y/N JMO 文件是不是自启动文件

10. FORMAT JM_NO

功能: 格式化编程器中的 FLASH 盘

参数: JM_NO: 编程器号(0 表示所有连接的编程器)

返回: 状态信息(见附录)

11. JM JM_NO

功能: 检测电脑连接的编程器台数/连接方式

参数: JM_NO: 编程器号

返回: 当 JMO=0 时返回电脑连接的编程器台数

当 JMO=1..N 时返回这台编程器的的序列号及名称。

HB.LB: 高位表示通过编程器扩展板连接, 高字节(HB)表示搞展板号,低字节(LB)表示扩展板上编程器的序号

12. LASTERROR JM_NO

功能: 读编程器上一次操作的返回值

参数: JMNO: 编程器号

返回: 上条命令的返回状态信息

13. HELP

功能: 显示常用命令的命令列表

参数: 无

返回: 显示常用命令的命令列表

14. LOGOFF

功能: 关闭 LOG 文件

参数: 无

返回: 状态信息(见附录)

15. LOGON CREATE/APPEND, "FILENAME"

功能: 创建一个记录编程信息文件, 或往信息文件中添加一个记录。

参数: CREATE: 创建一个新的名为"FILENAME"LOG 文件

APPEND: 在原"FILENAME"LOG 文件最后添加

返回: 状态信息(见附录)

16. MESSAGE JM_NO, 1/2/0

功能: 读取编程器进度信息

参数: JM_NO: 编程器号 1/2/0 指定编程口 1: P1 2: P2 0: P1 及 P2

返回: 当前进度信息, 是一个 32 位的字。如果当前没有新的信息, 就返回最近的一次信息

// B3	B2	B1	B0
// PNO	SPPEDBAR	ATE	STATUS
// 1: P1 口信息	进度条位置	NG,OK,BUSY	编程序所处状态
// 2: P2 口信息			PROG_NULL 到 PROG_FINISH

17. MODE JM_NO

功能: 显示编程器当前所处状态 (状态分监控态和编程态, 可以用 OPENJMO 和 CLOSEJMO 切换状态, 有些命令只能在监控态执行, 有些只能在编程态执行, 详见后面说明)

参数: JM_NO: 编程器号

返回: 状态信息(见附录)

18. NAME JM_NO

功能: 读取编程器名称

参数: JM_NO: 编程器号

返回: 编程器名称字符串

19. OPENJMO JM_NO, F_NO/0

功能: 打开 JMO 文件, 进入编程状态

参数: JM_NO: 编程器号(0 表示所有连接的编程器) F_NO: JMO 文件序号 0: 由 FID 命令指定的 JMO 文件序号

返回: 状态信息

20. POWER JM_NO, 1/2/0, VCC 电压

功能: 控制编程口的电源脚

参数: JM_NO: 编程器号(0 表示所有连接的编程器) 1/2/0: 编程器端口号 VCC: 电压值 0: OFF
3: 1.8V 4: 2.5V 5: 3.3V 6: 3.6V 7: 5.0V

返回: 状态信息

21. READIO JM_NO, 1/2, ADR

功能: 读取编程寄存器

参数: JMNO: 编程器号 1/2: 编程器端口号 ADR: 编程寄存器地址 0..15

返回: 指定编程端口号的指定寄存器值

22. RESET JM_NO

功能: 复位编程器, 复位后重新连接编程器, 使编程器处于重新加电状态。

参数: JM_NO: 编程器号(0 表示所有连接的编程器)

返回: 状态信息

23. RUN JM_NO, 1/2/0

功能: 执行一次编程, 编程完成后返回

参数: JM_NO: 编程器号(0 表示所有连接的编程器) 1/2/0 1: 编程端口 1 编程 2: 编程端口 2 编程 0: 编程端口 1 及编程端口 1 同时编程

返回: 状态信息

24. SERIAL JM_NO, 1/2/0, "STRING"HEX

功能: 往数据区填充外部编号数据, 填充数据的具体地址长度在生成 JMO 时设定, 可根据编程次数, 时间等生成不同的编程数据, 可参考编程器说明书“自动编号”部分。

参数: JM_NO: 编程器号(0 表示所有连接的编程器) 1/2/0 1: 编程端口 1 2: 编程端口 2 0: 编程端口 1 及编程端口 2

返回: 状态信息

25. SETAUTOFILE JM_NO, F_NO/0/255

功能: 设定自启动文件

参数: JM_NO: 编程器号(0 表示所有连接的编程器) F_NO: 将此序号的 JMO 文件设为自启动文件, 当 F_NO 为 255 时表示取消自启动文件, 0: 由 FID 命令指定的 JMO 文件序号

返回: 状态信息

26. SETNAME JM_NO, "NAME"

功能: 设定编程器名称

参数: JM_NO: 编程器号(0 表示所有连接的编程器) "NAME": 编程器名称

返回: 状态信息

27. SETPASSWORD JM_NO, "PASSWORD"

功能: 设定编程器密码

参数: JM_NO: 编程器号(0 表示所有连接的编程器) "PASSWORD": 编程器密码

返回: 状态信息

28. SNO JM_NO

功能: 读取编程器编号

参数: JM_NO: 编程器号

返回: 编程器编号

29. START JM_NO, 1/2/0

功能: 启动编程, 启动后不等待编程结束立即返回。需要用 MESSAGE 命令读回状态并判断编程是否完成

参数: JM_NO: 编程器号(0 表示所有连接的编程器) 1/2/0 1: 编程端口 1 编程 2: 编程端口 2 编程
 0: 编程端口 1 及编程端口 1 同时编程

返回: 状态信息

30. VERSION JM_NO

功能: 读取编程器软件版本号

参数: JM_NO: 编程器号

返回: 编程器软件版本号

31. WRITEIO JM_NO, 1/2/0, ADR, DAT

功能: 设置编程寄存器

参数: JM_NO: 编程器号(0 表示所有连接的编程器) 1/2/0: 编程器端口号 ADR: 编程寄存器地址
 0..15 DAT: 待写数据

返回: 状态信息

32. PORT USB/COM

功能: 选择软件与编程器的连接方式 缺省方式是 USB

参数: USB: USB 口接连 COM: 串口连接

返回: 状态信息

编程器上电后处于监控态，如果编程器有自启动文件，编程器上电后自动打开 JMO 文件，进入编程态。出错不影响编程器的状态。可以用 MODE 命令查看当前状态。

// 状态转换 只有下面两条命令会改变所处编程器状态

OPENJMO 将编程器从监控态转为编程态

CLOSEJMO 将编程器从编程态转为监控态

// 以下命令必须在监控态执行

(如果编程器处于编程态，执行
CLOSEJMO 退出编程态进入监控态，
再执行以下操作)

DELETE
DOWNLOAD
FID
FILE
NAME
OPENJMO
SETAUTOFILE
SETNAME
SETPASSWORD
SNO
VERSION
MODE

// 以下命令必须在编程态执行

(如果编程器处于监控态，执行
OPENJMO 退出监控态进入编程态
再执行以下操作)

CLOSEJMO
COUNTER
MESSAGE
POWER
READIO
SERIAL
RUN
START
WRITEIO
MODE

一个以批处理方式运行 JMCMD 进行编程的例子

```
@echo off
```

```
echo.
```

```
echo Start to execute MCU on board programming
```

```
echo It will take a few seconds, please wait..
```

```
echo.
```

```
start /wait jmcmd.exe -openjmo 1,4 -run 1,1 -exit
```

```
REM -openjmo 1,4 打开第一台编程器中的第四个 JMO。
```

```
REM 如果想要用 JMO 的文件名操作需要用两条命令
```

```
-fid 1,FLASH,"abc.jmo" -openjmo 1,0
```

```
REM -fid 命令找出 FLASH 盘上的"abc.jmo"的 JMO 文件序号
```

```
-openjmo 1,0 打开这个序号的 JMO
```

```
REM -run 1,1 启动第一台编程器 P1 编程口编程，并等待编程结束，编程如果正确返回 OK (0)，不  
正确返回 31-63 的值，表示是在那个状态出错了
```

```
REM -exit 退出 jmcmd 程序，最后一条必须是这条命令，否则 jmcmd 会等待用户在界面输入新命令
```

```
if errorlevel 1 goto error
```

```
if errorlevel 0 goto success
```

```
:error
```

```
echo FAIL>MESSAGE.TXT
```

```
echo.
```

```
echo programming FAILED!!!
```

```
goto end
```

```
:success
```

```
echo PASS>MESSAGE.TXT
```

```
echo.
```

```
echo programming SUCCESSFUL
```

```
:end
```

```
echo.
```

DLL 调用说明

了解了 JMCMD 的执行过程, DLL 调用就好理解了, DLL 函数与 JMCMD 的命令完全一样。

没有注明返回值类型的函数就是返回一个状态字, 参见附录状态字定义

输入参数:JMNO 要操作的编程器号, 如果只连接一台就是 1, 如果连接有 N 台, 就是 1, 2..N, 0 号表示操作所有编程器

DLL 有两个: JM_USB.DLL PC 机使用 USB 口控制编程器

JM_COM.DLL PC 机使用 COM 口控制编程器

两个 DLL 的接口是完全一样的, 下面的接口说明

C : u32 JM_BAUDRATE(u32 BAUDRATE);

PAS : function JM_BAUDRATE(BAUDRATE: u32): u32;

功能: 在用串口通信时设置波特率, 上电后缺省值为 115200

可以设置的波特率有: 115200/230400/460800/500000

C : u32 JM_CLOSEJMO(u32 JMNO);

PAS : function JM_CLOSEJMO(JMNO: u32): u32;

功能: 关闭打开的 JMO, 返回监控态

C : u32 JM_COUNTER (u32 JMNO);

PAS : function JM_COUNTER (JMNO: u32): u32;

功能: 读剩余编程次数

返回值: 剩余的编程次数 如果 JMO 没有限定编程次数, 返回的值无意义

C : u32 JM_DELETE(u32 JMNO, u32 N);

PAS : function JM_DELETE(JMNO: u32; N: u32): u32;

功能: 删除编程器中文件序号为 N 的 JMO 文件

N: 文件序号

C : u32 JM_DOWNLOAD(u32 JMNO, u32 F, char * p);

PAS : function JM_DOWNLOAD(JMNO: u32; F: u32; p: pointer): u32;

功能: 下载 JMO 文件到编程器

F: 下载到的盘 0: 内部 FLASH 1: TF p: 文件名字符串指针(0 结束)

C : u32 JM_EXIT(void);

PAS : function JM_EXIT: u32;

功能: 在主程序结束调用 DLL 前必须调用一次

C : u32 JM_FID(u32 JMNO, u32 F, char * p);

PAS : function JM_FID(JMNO: u32; F: u32; p: pointer): u32;

功能: 用文件名查找对应的文件序号

F: 文件所在盘 0: 内部 FLASH 1: TF p: 文件名字符串指针(0 结束)

返回值: 文件所对应的文件序号 0 表示未找到

C : char * JM_FILE(u32 JMNO, u32 F);

PAS : function JM_FILE(JMNO: u32; F: u32): pointer;

功能: 读编程器 JMO 文件属性 F: 文件序号

返回值: JM_FILE(0) 返回编程器中 JMO 文件个数 (字符串, 0 结束)

JM_FILE(N) 返回编程器中 N 号文件的属性 (字符串, 0 结束) DISK, FILENAME, DESCRIBE, CHKSUM, DATETIM, CHIPTYPE, SERIAL, AES, PID, COUNTER, S TART, ATEM, A UTOFILE, 每个属性用", "(逗号) 隔开

C : u32 JM_FORMAT(u32 JMNO);

PAS : function JM_FORMAT(JMNO: u32): u32;

功能: 格式化编程器中的 FLASH 盘

C : u32 JM_JM(u32 JMNO);

PAS : function JM_JM(JMNO: u32): u32;

功能: 计算电脑连接的编程器台数/连接方式

返回值: 当 JMO=0 时返回电脑连接的编程器台数, 当 JMO=1..N 时返回这台编程器的序列号及名称。HB.LB: 高位表示通过编程器扩展板连接, 高字节(HB)表示搞展板号, 低字节(LB)表示扩展板上编程器的序号

C : u32 JM_LASTERROR(u32 JMNO);

PAS : function JM_LASTERROR(JMNO: u32): u32;

功能: 读编程器上一次操作的返回值

C : u32 JM_MESSAGE(u32 JMNO, u32 P);

PAS : function JM_MESSAGE(JMNO: u32; P: u32): u32;

功能: 读取编程器信息 编程器在编程的过程中会将当前的进度, 编程操作等信息放到一个信息队列中(最多 32 条信息), 由电脑读取

P: 编程口 JM_MESSAGE(JMNO, 1)只读取编程口 P1 的信息, 读到编程口 P2 的信息会丢弃
JM_MESSAGE(JMNO, 2)只读取编程口 P2 的信息, 读到编程口 P1 的信息会丢弃
JM_MESSAGE(JMNO, 0)会读取编程口 P1 及编程口 P2 的信息

返回值: 一个 32 位字 (4 个字节), 每个字节对应不同的状态

最高字节 BYTE3: 1: 这个信息是编程口 P1 的 2: 这个信息是编程口 P2 的

BYTE2: 进度条信息 100 表示进度条满

BYTE1: ATE 接口信息(第 0 位是 BUSY, 第 1 位是 OK, 第 2 位是 NG)

BYTE0: 编程口所处的状态

如果当前没有新的信息, 会返回最后一次的信息

JM_MESSAGE 用于 JM_START 启动后判断编程进度及编程状态, 过程如下:

1. 用 JM_START 启动编程

2. 用 JM_MESSAGE 等待 ATE.BUSY 有效, 这时表示编程开始

3. 用 JM_MESSAGE 等待 ATE.BUSY 无效, 这时表示编程结束

4. 用 JM_MESSAGE 看 ATE.OK 及 ATE.NG 如果 ATE.OK 有效, 表示编程正确 如果 ATE.NG 有效表示编程出错 出错的信息在 BYTE0 中

BUSY, OK, NG 的有效电平可以在生成 JMO 时设置, 在用 JM_FILE 读文件属性时的 ATEM 就是用户设置的有效电平。

C : u32 JM_MODE(u32 JMNO);

PAS : function JM_MODE(JMNO: u32): u32;

功能: 当前编程器所处的工作状态, 编程有两个工作状态 MODE_MONITOR: 监控态
MODE_PROGRAM: 编程态

C : char * JM_NAME(u32 JMNO);

PAS : function JM_NAME(JMNO: u32): pointer;

功能: 读取编程器名

返回值: 编程器名字符串指针(0 结束)

C : u32 JM_OPENJMO(u32 JMNO, u32 N);

PAS : function JM_OPENJMO(JMNO: u32; N: u32): u32;

功能: 打开一个 JMO 文件 如果当前是 MODE_PROGRAM 编程态, 会保持原来打开的
JMO 文件, 报出错。必须先用 JM_CLOSEJMO 关闭打开的 JMO 文件后, 才能打开另一个 JMO

C : u32 JM_POWER(u32 JMNO, u32 P, u32 V);

PAS : function JM_POWER(JMNO: u32; P: u32; V: u32): u32;

功能: 控制编程口 PIN1 的电源输出

P: 编程口 1: 编程口 P1 2: 编程口 P2 0: 两个编程口 V: 电压 0: 关闭 3: 1.8V
4: 2.5V 5: 3.3V 6: 3.6V 7: 5V

C : u32 JM_READIO(u32 JMNO, u32 P, u32 A);

PAS : function JM_READIO(JMNO: u32; P: u32; A: u32): u32;

功能: 读取编程口控制寄存器值(寄存器说明参见 JM_WRITEIO 函数)

P: 编程口 1: 编程口 P1 2: 编程口 P2 A: 编程口控制寄存器地址

C : u32 JM_RESET(u32 JMNO);

PAS : function JM_RESET(JMNO: u32): u32;

功能: 复位编程器 复位后重新连接编程器, 使编程器处于重新加电状态。

C : u32 JM_RUN(u32 JMNO, u32 P);

PAS : function JM_RUN(JMNO: u32; P: u32): u32;

功能: 开始编程, 编程结束后才返回 如果要显示编程的进度(擦除、编程、校验等过程信息,
请使用 JM_START 及 JM_MESSAGE)

P: 编程口 1: 编程口 P1 2: 编程口 P2 0: 编程口 P1 和编程口 P2

C : u32 JM_SERIAL(u32 JMNO, u32 P, u32 L, char *s);

PAS : function JM_SERIAL(JMNO: u32; P: u32; L: u32; s: pointer): u32;

功能: 往数据区填充外部编号数据, 填充数据的具体地址长度在生成 JMO 时设定, 可根据编程次数, 时间等生成不同的编程数据, 可参考编程器说明书“自动编号”部分。

P: 编程口 1: 编程口 P1 2: 编程口 P2 0: 编程口 P1 和编程口 P2 L: 数据长度 s: 数据串指针

在启动编程(JM_RUN、JM_START)前注入数据

C : u32 JM_SETAUTOFILE(u32 JMNO, u32 N);

PAS : function JM_SETAUTOFILE(JMNO: u32; N: u32): u32;

功能: 设置自启动文件

N: 将文件序号为 N 的 JMO 设为自启动文件, N 为 255 是表示取消自启动文件, N 为 0 时, 将 JM_FID 命令查到的序号文件设为自启动文件。(参见前面 JM_FID 命令)

C : u32 JM_SETNAME(u32 JMNO, char * p);

PAS : function JM_SETNAME(JMNO: u32; p: pointer): u32;

功能: 设置编程器名

p: 编程器名字符串指针(0 结束)

C : u32 JM_SETPASSWORD(u32 JMNO, char * p);

PAS : function JM_SETPASSWORD(JMNO: u32; p: pointer): u32;

功能: 设置编程器的密码

p: 密码字符串指针(0 结束)

C : u32 JM_SNO(u32 JMNO);

PAS : function JM_SNO(JMNO: u32): u32;

功能: 读取编程器序列号

返回值: 编程器序列号

C : u32 JM_START(u32 JMNO, u32 P);

PAS : function JM_START(JMNO: u32; P: u32): u32;

功能: 开始编程, 启动后立即返回 需要用 JM_MESSAGE 来判断编程进度和状态。

P: 编程口 1: 编程口 P1 2: 编程口 P2 0: 编程口 P1 和编程口 P2

C : u32 JM_VERSION(u32 JMNO);

PAS : function JM_VERSION(JMNO: u32): u32;

功能: 读取编程器版本号

返回值: 版本号

C : u32 JM_WRITEIO(u32 JMNO, u32 P, u32 A, u32 D);

PAS : function JM_WRITEIO(JMNO: u32; P: u32; A: u32; D: u32) : u32;

功能: 写编程口控制寄存器值

P: 编程口 1: 编程口 P1 2: 编程口 P2 0: 两个口同时 A: 编程口控制寄存器地址 D: 写入数据

JM_READIO, JM_WRITEIO 操作的编程口控制寄存器定义如下:

不同方式(UART,LIN,JTAG,SWD,BDM 等)的编程接口, 编程口控制寄存器的定义不同, 但地址 0, 1, 2 编程口控制寄存器是统一的

0: LED 只写寄存器, 用于控制编程口的 LED

1: IOCTRL 只写寄存器, 用于控制编程口的方向

2: IODATA 读写寄存器, 用于读入/输出编程口的信号

LED 寄存器 bit7..bit5, bit1..bit0: 保留

bit2: 编程口灯红色

bit3: 编程口灯绿色

bit4: 编程口灯闪烁

写入 0x00 编程口灯不亮

写入 0x04 编程口灯红色

写入 0x08 编程口灯绿色

写入 0x0C 编程口灯黄色

写入 0x14 编程口灯红色并闪烁

IOCTRL 寄存器 bit6..bit0: 7 个编程 IO 输入/输出方向控制

0: 输入 1: 输出

bit6..bit0 分别对应 PIN9,PIN 7 ..PIN2

IODATA 寄存器 bit6..bit0: 7 个编程 IO 输入/输出值

0 : 低电平 1 : 高电平

bit6..bit0 分别对应 PIN9,PIN 7 ..PIN2

C : u32 JM_WRITELOG(char *p);

PAS : function JM_WRITELOG(p: pointer): u32;

功能: 在 LOG 文件中写入一个字串

附录：状态信息定义

```
OK = 0; // 运行正常，没有错误
```

```
// 下面 1-10 为编程器上电自检错误信息
```

```
ERROR_MONI = 1; // 编程器监控程序错误
ERROR_ALGO = 2; // 编程器编程算法错误
ERROR_RTC = 3; // 编程器中的实时时钟错误，通常是电池没电了
ERROR_FLASH_FORMAT = 4; // 内部 FLASH 格式错误，需要重新格式化
ERROR_FLASH_TF = 5; // TF 卡错误
ERROR_FPGA_ID = 6; // 编程器中的 FPGA ID 错误
ERROR_FPGA_CFG = 7; // 编程器中的 FPGA 配置错误
ERROR_FPGA_IP = 8; // 编程器中的 FPGA IP 码错误
ERROR_FLASH_ID = 9; // 编程器中的 FLASH ID 错误
ERROR_FLASH_CFG = 10; // 编程器中的 FLASH 配置错误
```

```
// 下面 20-24 为打开 JMO 的错误信息
```

```
ERROR_CHIPTYPE = 20; // 芯片类型错误
ERROR_PROGID = 21; // JMO 文件限定了编程器编号，
// 已连接的编程序器编号不在允许列表中
ERROR_COUNTER = 22; // JMO 限定了编程记数，现在剩余编程次数为 0
ERROR_AES = 23; // JMO 设定了加密，现在编程器中的密码不对
ERROR_CHKSUM = 24; // JMO 的 CHKSUM 不对
```

```
// 下面 31-63 为编程器当前的工作状态
```

```
PROG_NULL = 31; // 编程器空闲
PROG_WAIT_KEY = 32; // 编程器在等待用户按键启动编程
PROG_AUTO = 33; // 编程器在等待用户连接芯片
PROG_TEST_PIN = 34; // 编程器在编程完毕，用户可以用 plugin 程序测试芯片
PROG_CONNECT = 35; // 编程器在连接芯片
PROG_RESTORE = 36; // 编程器在恢复芯片
PROG_ERASE = 37; // 编程器在擦除芯片
PROG_WRITE = 38; // 编程器在编程芯片
PROG_READ = 39; // 编程器在读芯片
PROG_VERIFY = 40; // 编程器在校验芯片
PROG_PROTECT = 41; // 编程器在编程加密信息
PROG_UNPROTECT = 42; // 编程器在解除芯片加密
PROG_BLANKCHECK = 43; // 编程器在做空片检查
PROG_CTRL = 44; // 编程器在 CTRL
PROG_EPROM = 45; // 编程器在编程 EPROM 或 OTP
PROG_BOOTLOAD = 46; // 编程器在编程 BOOTLOAD
PROG_ATE = 47; // 编程器在等待 ATE 接口的 START 信号
```

PROG_SVF	= 48;	// 下面 48-57 用于 JTAG 口编程
PROG_IDCODE	= 49;	// 编程器在执行 IDCODE
PROG_CHECK_OPT	= 50;	// 编程器在执行 CHECK_OPT
PROG_PIO	= 51;	// 编程器在执行 PIO
PROG_WRITE_OPT	= 52;	// 编程器在执行 WRITE_OPT
PROG_READ_OPT	= 53;	// 编程器在执行 READ_OPT
PROG_TEST	= 54;	// 编程器在执行 TEST
PROG_USER_ID0	= 55;	// 编程器在执行 USER_ID0
PROG_USER_ID1	= 56;	// 编程器在执行 USER_ID1
PROG_USER_ID2	= 57;	// 编程器在执行 USER_ID2
PROG_EXECUTE	= 58;	// 编程器在运行芯片中的程序
PROG_JMO_NO	= 59;	// 编程器在执行多 JMO 合并中的一个 JMO
PROG_DOWNLOAD	= 60;	// 编程器在下载编程程序
PROG_POWERON	= 61;	// 编程器在上电
PROG_POWEROFF	= 62;	// 编程器在下电
PROG_FINISH	= 63;	// 编程器完成工作

// 70-79 是命令执行过程中的错误信息

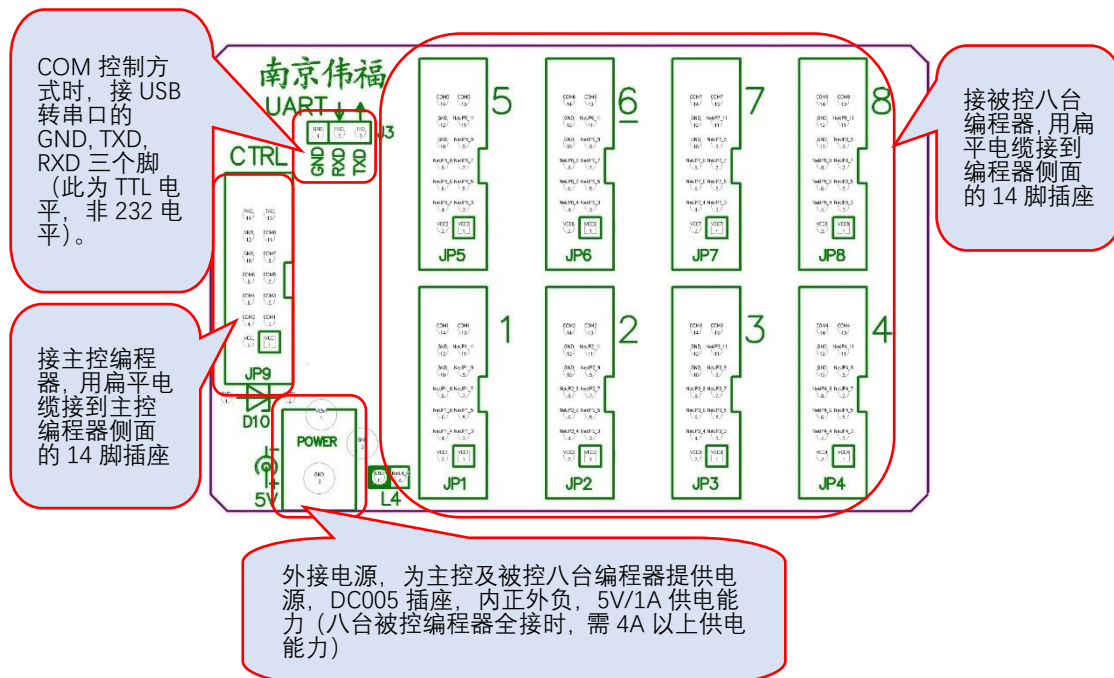
ERROR_CONNECT	= 70;	// 连接编程器错误
ERROR_PARAMETER	= 71;	// 命令的参数不对
ERROR_FILE_NOT_FOUND	= 72;	// 文件未找到
ERROR_READ_FILE	= 73;	// 文件读错误
ERROR_WRITE_FILE	= 74;	// 文件写错误
ERROR_TOO_MANY_COMMAND	= 75;	// 命令行太多（最多 1024 行， 超过了可以放在不同的文件中用 CMD 命令调用）
ERROR_UNKNOWN_COMMAND	= 76;	// 命令错
ERROR_JMO_FILE	= 77;	// JMO 文件错
ERROR_JMO_DOWNLOAD	= 78;	// JMO 文件下载错
ERROR_UNKNOWN	= 79;	// 未知错误

// 80-81 为编程器状态

MODE_MONITOR	= 80;	// 编程器在监控态
MODE_PROGRAM	= 81;	// 编程器在编程态

伟福一带八编程扩展板使用说明

伟福一带八编程扩展板可以用一台主控编程器带八台编程器进行编程，为大批量生产提供方便，主控编程器可以用电脑控制八台编程器，也可以脱机控制八台编程器。主控编程器通过侧面 14 插座接到扩展板，再控制八台编程器，电脑可以通过 USB 或 COM(串口)连接控制主控编程器。

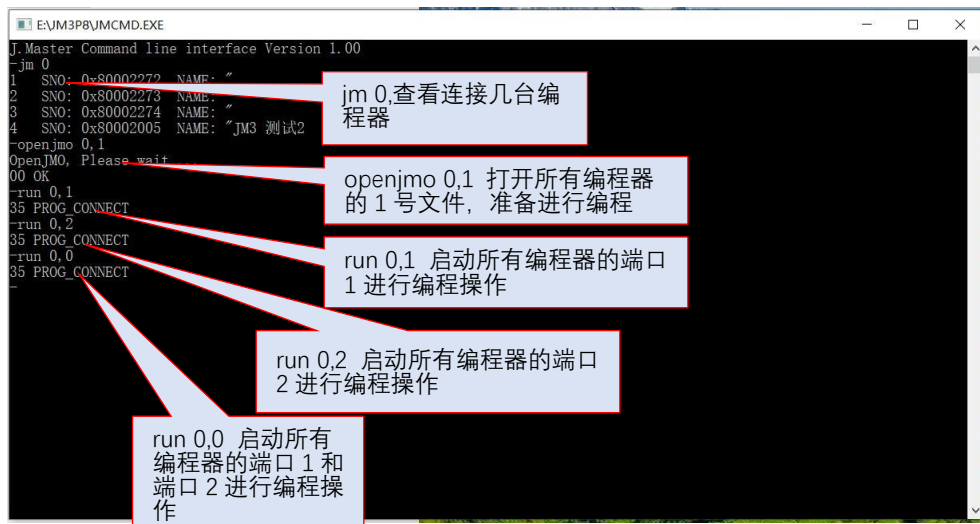


操作步骤:

1. 将一台普通编程器转为主控编程器，先按住此编程器的三个红键不松，接上 USB 线，加电约三秒后松手，将编程器固件清除，启动 JM3E.EXE 软件将固件刷成主控编程器（如图）。此处可以设置主控编程器与被控编程器的通信速率。如果以后想将主控编程器恢复为普通编程器，也是按住三个红键不松，加电清空固件，启动 JM3P.EXE 编程软件，会自己刷成普通编程器。



2. 如果是早期的 JM3 编程器也需逐一接到 JM3P.EXE 编程软件，无需清空固件，软件会自动更新为新的固件。更新完成后，关闭 JM3P.EXE 编程软件，以免与后面用到的 JMCMD 软件起冲突。
3. 启动 JMCMD 控制台软件，输入“JM 0”查看连接了几台编程器，JMCMD 使用方法参见上面的使用说明。



使用编程扩展板注意事项:

- 1, 上位机软件用 JMCMD 控制台软件, 可以通过 USB 或者 COM 方式进行控制。
- 2, 被编程的 JMO 文件须由 JM3P 编程软件生成 (即伟福单片机编程软件), 并下载到被控编程器里的 FLASH 或 TF 卡里。
- 3, 所有被控制的编程器里被编程的 JMO 文件须存在相同位置, 例, 想控制多台编程器编程 A. JMO 到用户板, 那么 A. JMO 都必须存在编程器内部 FLASH 或 TF 卡的 1 号位置, 或同在 2 号位置。如果不在相同位置, 请格式化 FLASH 或删除 TF 卡时所有文件后再下载 JMO 文件。
- 4, 加电顺序, 将主控编程器及被控编程器的侧面 14 脚用扁平电缆接到编程扩展板相应插座后, 先接 5V 外接电源, 看到主控编程器显示屏上出被编程文件后, 再接 USB 电缆 (USB 控制方式) 或转串口模块 (COM 控制方式), 再启动 JMCMD 软件。
- 5, USB 控制方式是将 USB 电缆接主控编程器的端口, 被控的普通编程器不能接 USB 电缆,
- 6, COM 控制方式是用串口方式控制主控编程器进行编程操作, 将串口信号的 TXD 接编程扩展板的 RXD 信号, 串口信号的 RXD 接扩展板的 TXD 信号, 地接地, 串口信号为 TTL 电平, 而不是经电压转换过的 232 电平, 串口信号可以由 USB 转串口模块得到, 也可以把 232 电平 (高压信号) 转换为 TTL 电平后得到。COM 控制方式时, 主控编程器无须接 USB 电缆。JMCMD 软件缺省控制方式为 USB 控制方式, 如果用 COM 控制方式, 首先要用 “port COM” 命令将控制方式设为 COM 方式才能正常工作。
- 7, 脱机编程 (不用电脑控制) 时, 无论按主控编程器上 P1 或 P2 键启动编程, 都会启动被控编程器的两个端口进行编程。这点与单机脱机编程有所不同。

