

# GDLink-OB 调试器用户手册

## 1 功能概述

GDLink 是 GigaDevice（兆易创新）针对旗下 GD32 微控制器推出的一款调试器，其功能类似 STM32 系列微控制器的 STLink 调试器。

GDLink 适用于 GD32 系列的 Cortex M3、M4、M23 内核产品，尤其是对于 ARMv8-M 架构的 Cortex M23 内核如 GD32E23x 系列支持较好。

GDLink 功能强大但成本较高，GDLink-OB 对 GDLink 不常用的功能进行了删减，只保留常用的 SWD 调试功能，降低了成本，使用上兼容 GD32 的 START 系列开发板板载调试器。

GDLink-OB 使用 ARM 标准的 CMSIS-DAP 协议，理论上可用于所有 ARM Cortex M 系列 MCU，同时 USB 接口使用 HID 协议，免去安装驱动的麻烦。

目前 GDLink-OB 已经升级为 GDLink-S 和 GDLink-Lite，支持官方版本除离线下载之外的完整功能。

### 1.1 主要特性

- ◆ 小巧便携，仅优盘大小，重量不足 8 克。
- ◆ 使用标准 CMSIS-DAP 协议，支持所有 ARM Cortex M 系列 MCU 的调试。
- ◆ USB 使用 HID 协议，Windows 下免驱动。
- ◆ 支持 MDK、pyOCD、GD-Link Programmer 等软件。
- ◆ 内置 500mA 自恢复保险丝。
- ◆ 支持固件升级。

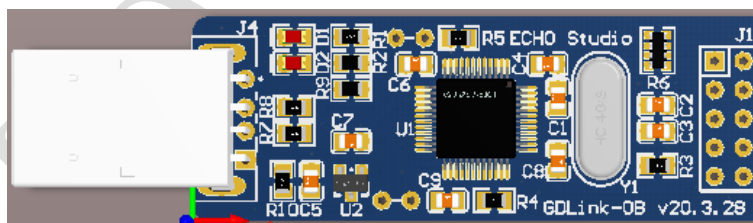


图 1 GDLink-S 正面外观



图 2 GDLink-Lite 正面外观

## 1.2 接口定义

GDLink-S 输出使用 2x5 PIN 2.54 排针，排针定义见背面丝印图 3。



图 3 2x5PIN 接口定义

SWD 调试通常只需要连接 5.0V、GND、CLK、DIO 四个管脚，**注意 CLK 和 DIO 在靠外的排针上，方形焊盘为 TDI，不是 CLK**；如目标板独立供电、可不连接 5.0V 电源，3.3V 电源对外输出能力有限，不建议使用 3.3V 电压对目标板供电。

GD32E23x 系列为 Cortex M23（ARMv8-M）内核，实测需要连接 TRESET 才能正常连接。

GDLink-OB 目前暂不支持标准 JTAG，因此 TCK、TMS、TDI、TDO 四个信号仅供参考。

短接 TRESET（早期版本为 TDO/SWO）到 GND 后连接电脑，GDLink-OB 进入 IAP 模式，可升级固件，详见 3。

## 1.3 版本描述

GDLink-OB: GDLink Programmer 限制了部分功能，已停产。

GDLink-S: 热缩管塑封，USB-A 接口。

GDLink-J: 热缩管塑封，兼容 JLink-OB。

GDLink-Lite: 带外壳，MiniUSB 接口。

S 版本和 Lite 版本功能完全相同，只有外观的和 USB 接口的差别。

## 2 使用方法

### 2.1 Keil MDK

测试 MDK 版本为 5.29，在 MDK 中正确设置调试参数后，可正常下载、调试代码。

Cortex M4 内核的 GD32F330 调试设置见图 4，调试器选择“CMSIS-DAP Debugger”。

其它 Cortex M3/M0 内核的 MCU 设置相同，不再赘述。

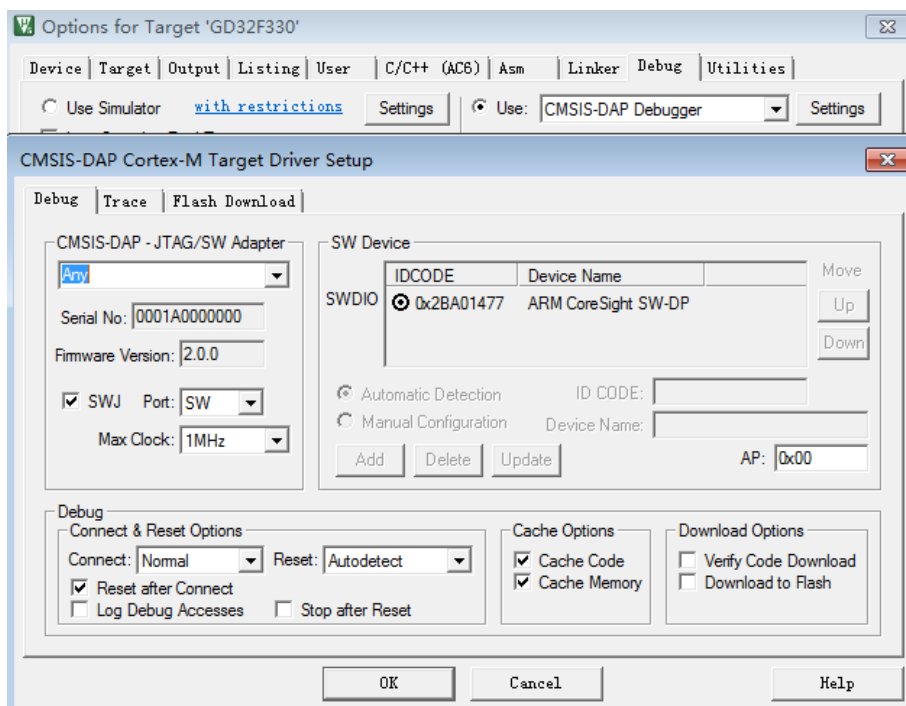


图 4 Cortex M4 内核的 GD32F330 调试设置

Cortex M23 内核的 GD32E230 调试设置见图 5，调试器选择“CMSIS-DAP ARMv8-M Debugger”。

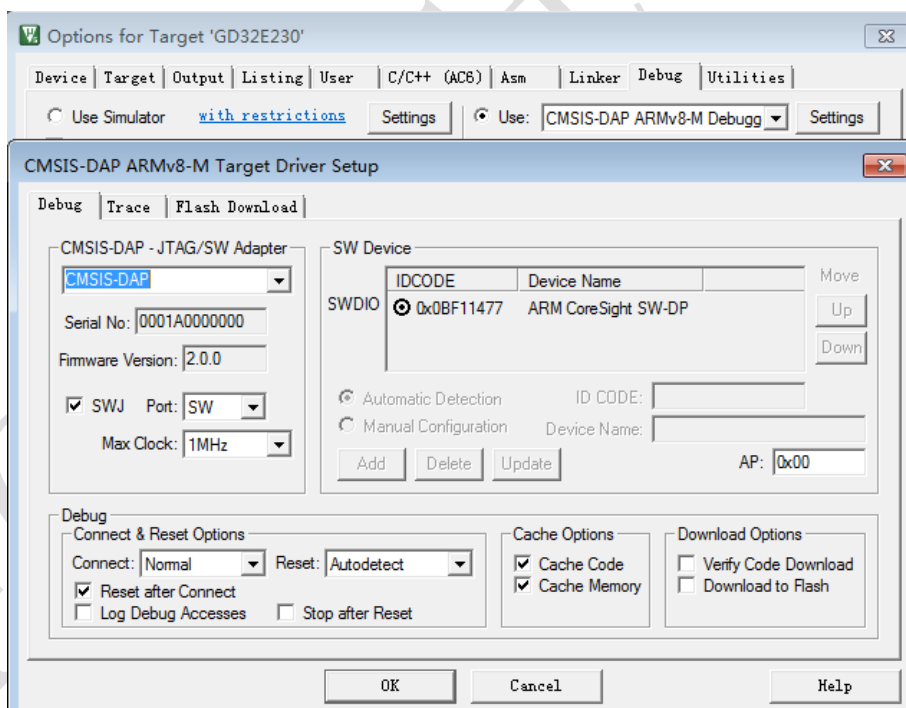


图 5 Cortex M23 内核的 GD32E230 调试设置

## 2.2 GD-Link Programmer

目前 GDLink-S 支持的 GD-Link Programmer 版本为 3.0.0.5950 和 4.5.1.10871。4.3.7.9954 不联网，测试比较充分，稳定性比较好；4.5.1.10871 支持一些新的芯片，客户可以按需选择。

### 2.2.1 GDLink-S/Lite

GDLink-S 为 GDLink-OB 的升级版，两者主要差别在使用 GD-Link Programmer 时，GDLink-S 不会弹出错

误！未找到引用源。的提示，并且支持 Target 菜单下的全部闪存操作，包括：

- 设置、解除读保护
- 整片擦除、页擦除闪存
- 闪存查空与对比
- 闪存编程
- 闪存整片读取与部分读取
- 运行程序

GDLink-Lite 为 GDLink-S 的带外壳版本，两者功能完全相同。

使用 GDLink-S 读取整片闪存见图 6。

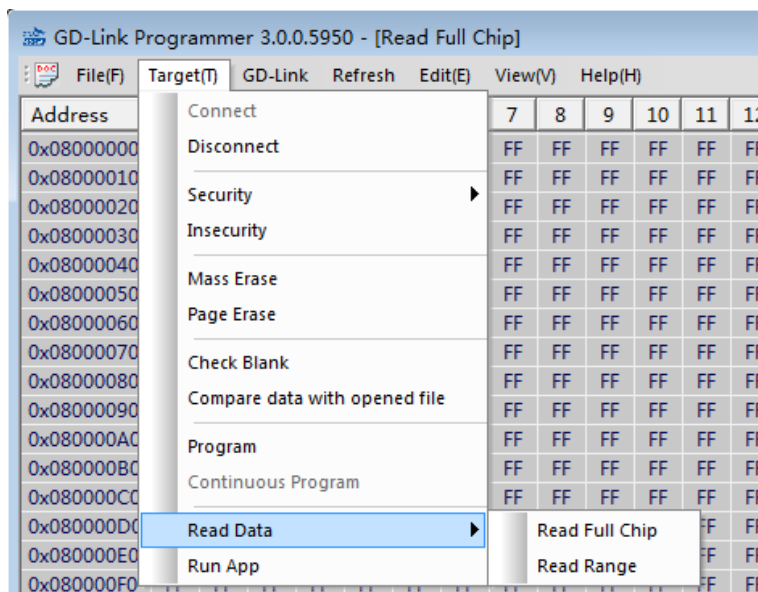


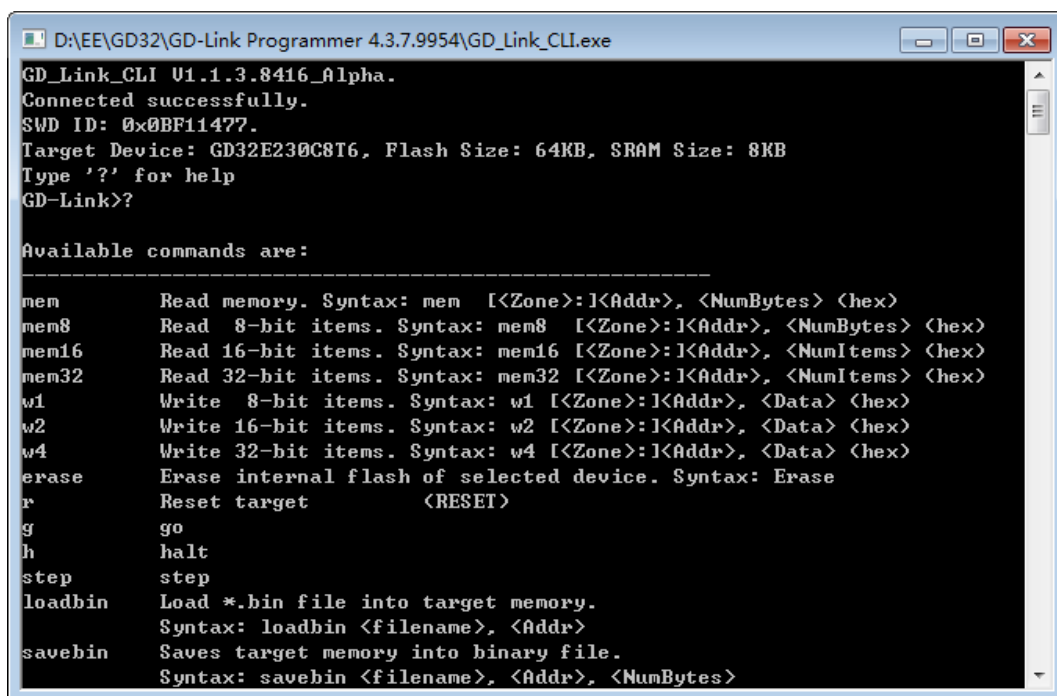
图 6 使用 GDLink-S 读取整片闪存

## 2.3 GD\_Link\_CLI

该工具来自 GDLink Programmer 4.3.7.9954 的安装包。直接点击 GD\_Link\_CLI.exe 即可运行，如图 7。输入 '?' 可以查看在线帮助。命令简介如下：

- mem/mem8/mem16/mem32 命令查看内存。
- w1/w2/w4 命令写内存。
- erase 命令擦除闪存。
- r/g/h/step 命令控制程序运行。
- laodbin/savebin 命令加载保存 bin 文件。
- SetPC 命令设置程序计数器。
- SetRDP 命令设置读保护。
- ReadAP/WriteAP/ReadDP/WriteDP 命令读写 CoreSight 寄存器。
- si 命令切换调试接口（GDLink-S 只支持 SWD，该命令无效）
- q 命令退出。

操作演示视频：<https://www.bilibili.com/video/bv18Z4y1w7Sy>



```
D:\EE\GD32\GD-Link Programmer 4.3.7.9954\GD_Link_CLI.exe
GD_Link_CLI V1.1.3.8416_Alpha.
Connected successfully.
SWD ID: 0x0BF11477.
Target Device: GD32E230C8T6, Flash Size: 64KB, SRAM Size: 8KB
Type '?' for help
GD-Link>?

Available commands are:
-----
mem      Read memory. Syntax: mem [<Zone>:]<Addr>, <NumBytes> <hex>
mem8     Read 8-bit items. Syntax: mem8 [<Zone>:]<Addr>, <NumBytes> <hex>
mem16    Read 16-bit items. Syntax: mem16 [<Zone>:]<Addr>, <NumItems> <hex>
mem32    Read 32-bit items. Syntax: mem32 [<Zone>:]<Addr>, <NumItems> <hex>
w1       Write 8-bit items. Syntax: w1 [<Zone>:]<Addr>, <Data> <hex>
w2       Write 16-bit items. Syntax: w2 [<Zone>:]<Addr>, <Data> <hex>
w4       Write 32-bit items. Syntax: w4 [<Zone>:]<Addr>, <Data> <hex>
erase    Erase internal flash of selected device. Syntax: Erase
r        Reset target          (RESET)
g        go
h        halt
step     step
loadbin  Load *.bin file into target memory.
         Syntax: loadbin <filename>, <Addr>
savebin  Saves target memory into binary file.
         Syntax: savebin <filename>, <Addr>, <NumBytes>
```

图 7 GD\_Link\_CLI 运行界面

## 2.4 pyOCD

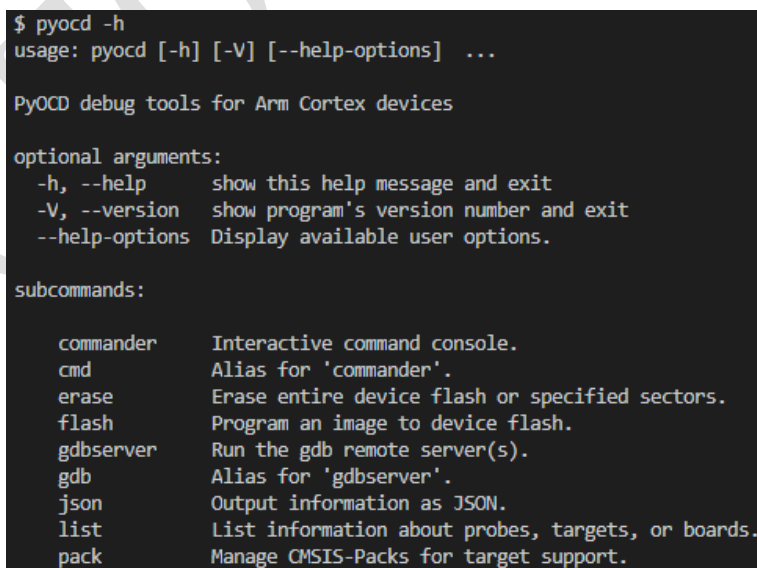
pyOCD 是一款使用 CMSIS-DAP 协议对 ARM Cortex-M 系列 MCU 进行编程、调试的开源 Python 库，功能十分强大。

项目主页: <https://github.com/mbedmicro/pyOCD>

GDLink-OB 完全支持使用 pyOCD 对目标芯片进行擦除、编程、调试等操作。

可使用"pip install pyocd"命令安装 pyocd, 建议使用 Python3.6 来安装 pyocd.

安装后, 帮助信息输出如下:



```
$ pyocd -h
usage: pyocd [-h] [-V] [--help-options] ...

PyOCD debug tools for Arm Cortex devices

optional arguments:
  -h, --help            show this help message and exit
  -V, --version          show program's version number and exit
  --help-options        Display available user options.

subcommands:

  commander            Interactive command console.
  cmd                  Alias for 'commander'.
  erase                 Erase entire device flash or specified sectors.
  flash                Program an image to device flash.
  gdbserver            Run the gdb remote server(s).
  gdb                  Alias for 'gdbserver'.
  json                 Output information as JSON.
  list                 List information about probes, targets, or boards.
  pack                 Manage CMSIS-Packs for target support.
```

图 8 pyocd -h 命令输出

使用"pyocd list"命令列出当前的调试器信息:

```
$ pyocd list
#   Probe                Unique ID
-----
0   GD32 ARM CMSIS-DAP    0001A0000000
```

图 9 pyocd list 查看当前调试器信息

pyocd 需要使用 PACK 包来支持对应的目标器件，PACK 包是一个.pack 后缀的压缩文件，内部存储了器件的支持信息。在 MDK 中，通过菜单：Project->Manage->Pack Installer 打开 PACK 包管理器。

为了输入命令方便，建议设置一个全局环境变量\$PACKS 来记录 PACK 包的绝对路径。通常这个目录在 MDK 的安装目录下，相对路径为"Keil\_v5\ARM\PACK\Download"。

使用"pyocd erase"命令擦除一颗 GD32E230C8 芯片的闪存内容，见图 10。

```
$ pyocd erase --chip --target gd32e230c8 --pack=$PACKS/GigaDevice.GD32E230_DFP.1.0.0.pack
0000895:INFO:eraser:Erasing chip...
0000977:INFO:eraser:Done
```

图 10 pyocd erase 命令擦除器件闪存

使用"pyocd flash"命令烧录一个 APP.hex 固件，见图 11。

```
$ pyocd flash APP.hex --target gd32e230c8 --pack=$PACKS/GigaDevice.GD32E230_DFP.1.0.0.pack
[=====] 100%
0004706:INFO:loader:Erased 57344 bytes (56 sectors), programmed 57344 bytes (56 pages), skip
```

图 11 pyocd erase 命令写入.hex 固件

使用"pyocd cmd"命令进入交互式命令行，可以执行更多的操作，比如查看寄存器、查看内存、外设，写入内存外设等操作。见图 12。

```
$ pyocd cmd --target gd32e230c8 --pack=$PACKS/GigaDevice.GD32E230_DFP.1.0.0.pack
Connected to GD32E230C8 [Lockup]: 0001A0000000
>>> r 0x08004000 0x80
08004000: 66 69 72 73 74 2e 0a 00 f8 b5 b9 4c 38 21 20 46 first.....L8! F
08004010: 00 f0 2e f9 30 b1 b6 48 39 21 00 f0 29 f9 08 b1 ....0..H9!..)...
08004020: 00 20 f8 bd 60 20 21 5a 48 48 c1 80 62 21 61 5a . ..`!ZHH..b!aZ
08004030: 01 81 66 21 61 5a 41 81 21 88 62 88 12 04 55 18 ..f!aZA!.b...U.
08004040: 43 4b 1d 60 68 21 61 5a 81 81 42 49 42 4a 95 42 CK.`h!aZ..BIBJ.B
08004050: 03 d9 21 80 19 60 0b 0c 63 80 23 8a 65 8a 2d 04 ..!..`.c.#.e.-.
08004060: ed 18 3e 4b 1d 60 95 42 03 d9 21 82 19 60 0b 0c ..>K.`.B..!...`..
08004070: 63 82 23 8c 65 8c 2d 04 eb 18 39 4e 33 60 39 4d c.#.e.-...9N3`9M
>>> show cores
Cores:      1
Core 0 type: Cortex-M23
```

图 12 pyocd cmd 命令进入交互命令行

对于常用的擦除、编程操作，可以将操作写成脚本。图 13 是一个对 GD32E230 芯片进行擦除、编程的脚本。保存为 gd32e23x.sh 放入命令搜索路径。

使用"gd32e23x.sh erase"命令擦除器件。

使用"gd32e23x.sh flash APP.hex"命令写入固件。

使用"gd32e23x.sh info"命令查看器件信息。

更多脚本见：<https://github.com/xjtuecho/CMSIS-DAP/tree/master/GDLink-OB/Scripts>

```

1  #!/bin/sh
2
3  TARGET="gd32e230c8"
4  PACK_NAME="$PACKS/GigaDevice.GD32E230_DFP.1.0.0.pack"
5
6  Usage(){
7  |   echo "Usage: $0 [erase | flash | info]."
8  | }
9
10 if [ $# -eq 0 ]; then
11 |   Usage
12 | else
13 |   case $1 in
14 |   erase)
15 |       echo "Erase Device..."
16 |       pyocd erase --chip --target $TARGET --pack=$PACK_NAME
17 |       ;;
18 |   flash)
19 |       if [ $# -eq 2 ]; then
20 |           echo "Flash Device..."
21 |           pyocd flash $2 --target $TARGET --pack=$PACK_NAME
22 |       else
23 |           Usage
24 |           echo "Usage: $0 erase [APP.hex]"
25 |       fi
26 |       ;;
27 |   info)
28 |       echo "Try unlock Device..."
29 |       pyocd-flashtool --unlock --target $TARGET --pack=$PACK_NAME
30 |   esac
31 | fi

```

图 13 一个擦除、编程 GD32E230 的脚本

## 2.5 JLink JFlash

GDLink-J 内置了 JLink-OB 和 GDLink-S 双固件，运行 GDLink-S 固件时与 GDLink-S 功能完全相同，运行 JLink-OB 固件时可以使用 JLink 和 JFlash 工具，推荐 **v6.82 或者更早版本**，见图 14。

内置的 JLink-OB 固件版本为：J-Link OB-STM32F103 V1 compiled Feb 2 2021 16:45:54

```

JLink D:\EE\MCU\SEGGER\JLink_V682\JLink.exe
SEGGER J-Link Commander V6.82 (Compiled Jul 20 2020 13:39:47)
DLL version V6.82, compiled Jul 20 2020 13:38:38

Connecting to J-Link via USB...O.K.
Firmware: J-Link OB-STM32F103 V1 compiled Feb 2 2021 16:45:54
Hardware version: V1.00
S/N: 19870422
License(s): JFlash, RDDI, FlashDL, FlashBP, RDI, GDB
VTref=3.300V

Type "connect" to establish a target connection, '?' for help
J-Link>

```

图 14 JLink v6.82 启动界面

### 2.5.1 双固件切换

GDLink-J 同一时间只能运行一种固件，切换方法为：



使用短路冒短接 GDLink-J 输出端子上的 TRESET 和 GND，然后插入电脑，安装驱动以后，设备管理器中会出现一个虚拟串口，使用“超级终端”连接该串口可以输入命令。

- 切换为 JLink-OB: entry 8004000
- 切换为 GDLink-S: entry 8014000

```
entry
APP_ENTRY=0x08004000. ← 当前为JLink-OB固件
entry 8014000 ← 切换为GDLink-S固件
Set APP_ENTRY=0x08014000.
entry
APP_ENTRY=0x08014000.
entry 8004000 ← 切换为JLink-OB固件
Set APP_ENTRY=0x08004000.
entry ← 查看当前固件
APP_ENTRY=0x08004000.
```

已连接 0:00:57 ANSIW 115200 8-N-1 SCROLL CAPS NUM 捕 打印

图 15 双固件切换演示

JLink-OB 和 GDLink-S 固件的起始地址分别为 0x08004000 和 0x08014000，使用 entry 命令切换即可，注意地址均为十六进制，不带参数的 entry 命令显示当前 APP 入口地址。除 0x08004000 和 0x08014000 之外的其它地址无效，请勿设置。

## 3 常见问题 FAQ

### 3.1 为什么连不上目标芯片？

根据实际客户反馈，连不上目标芯片大部分原因是杜邦线接线问题。包括但不限于以下情况：

- 连接 JTAG 接口，GDLink-S 只支持 SWD，不支持 JTAG。
- CLK 和 DIO 管脚接错，从背面看排针丝印 CLK 和 DIO 在靠外的那排排针上。图 3 中的方形焊盘不是 CLK，而是 TDI。
- CLK 和 DIO 管脚交叉，SWD 调试的 CLK 和 DIO 是直连，不是交叉。
- 杜邦线不通。可以用万用表通断档排除该问题。
- 目标板应用程序使用了 PA14(SWCLK)、PA13(SWDIO)两个管脚。
- 目标板应用程序使用了低功耗功能，调试时请暂时关闭低功耗功能。
- 使用 GDLink 的 3.3V 给目标板供电，3.3V 对外输出能力有限，目标板请独立供电。

实际目标板上的 3.3V 通常会连接很多器件，GDLink 板载的 SOT-23 电源芯片带载能力有限，无法带动那么多元器件。目标板请独立供电，或者使用 GDLink-S 的 5V 给目标板供电。

### 3.2 注意 PA14(SWCLK)、PA13(SWDIO)默认状态

直接使用寄存器的用户需要特别注意：**不要修改 PA14 和 PA13 相关的默认值。**

因为 SWD 接口在 PORTA，PORTA 的复位状态和其它端口不同，PA14 和 PA13 默认复位为 AF 功能。

以 GD32E230 为例：

- GPIOA\_CTL 复位值为 0x28000000，即 CTL14=10b，CTL13= 10b，**即 AF 功能。**
  - GPIOA\_OSPD 复位值为 0x0C000000，即 OSPD14=00b，OSPD13=11b，SWDIO 速度为 50M。
  - GPIOA\_PUD 复位值为 0x24000000，即 PUD14=10b，PUD13=01b，SWCLK 为下拉，SWDIO 为上拉。
- 推荐使用固件库来初始化 GPIO，直接使用寄存器需要注意如果不使用 PA14 和 PA13 不要修改寄存器默



认值。

### 3.3 GDLink Programmer 提示软件过时？

目前 GDLink-S 支持 GDLink Programmer 3.0.0.5950，4.3.7.9954 和 4.5.1.10871 版本。使用更新的版本可能会弹出图 16 提示。

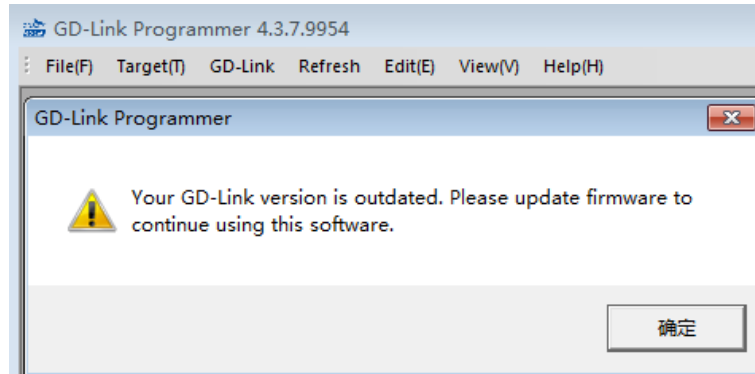


图 16 4.3.7.9954 版本提示

GDLink Programmer 从 4.3.7.9954 版本开始自带了一个命令行模式程序：GD\_Link\_CLI.exe，与 GDLink-S 配合工作良好，详情见 2.3。

### 3.4 MDK 无法调试？

MDK 无法调试时，先尝试使用 GDLink Programmer 来连接，排除 MDK 软件配置问题。

MDK 调试时，注意以下几点：

- 芯片不能有读保护；
- 用户的代码中不能使用 SWD 接口的两个管脚 PA13 和 PA14；
- 芯片中不能有低功耗操作。

### 3.5 是否支持全部 GD32 芯片？

GDLink-S 使用的协议为 ARM 的 CMSIS-DAP 协议，因此支持所有 GD32 的 ARM Cortex-M 芯片。

GD32VF103 系列使用 RISC-V 内核，只支持 JTAG，GDLink Programmer 的 Device Interface 切换为 JTAG 接口才能支持，使用完注意及时切换回 SWD 接口。

### 3.6 是否支持其它 Cortex-M 芯片比如 STM32？

实测可以在 MDK（Keil）中使用 GDLink-S 调试 STM32F103C8T6 芯片，MDK 目前属于 ARM 公司。其它 ARM 的 Cortex-M 芯片理论上也都支持。

由于厂商的限制，不能在 ST-Link Utility 这类专用上位机软件中使用。

### 3.7 下载固件时卡死

现象见图 17，固件下载中直接卡死。点击“确定以后”显示如图 18，进度条为红色，这时候固件并没有成功下载到芯片。原因在于用户的固件打开了选项字节中的写保护，解除读保护和全片擦除并不会关闭写保护功能。点击图 19 中的“Reset”按钮复位选项字节即可。

GDLink Programmer 3.0.0 版本不支持配置选项字节，4.3.7 开始支持，菜单：Target->Configure OptionBytes

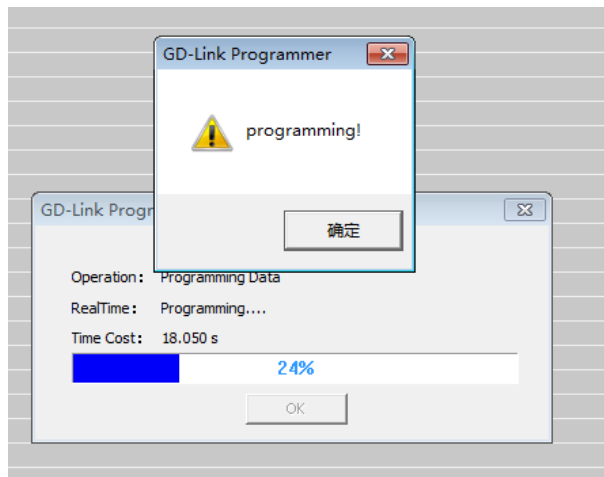


图 17 下载固件时卡死

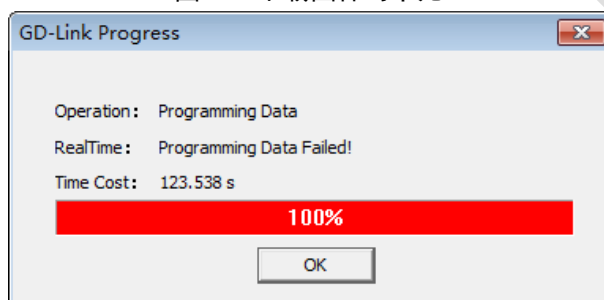


图 18 进度条为红色

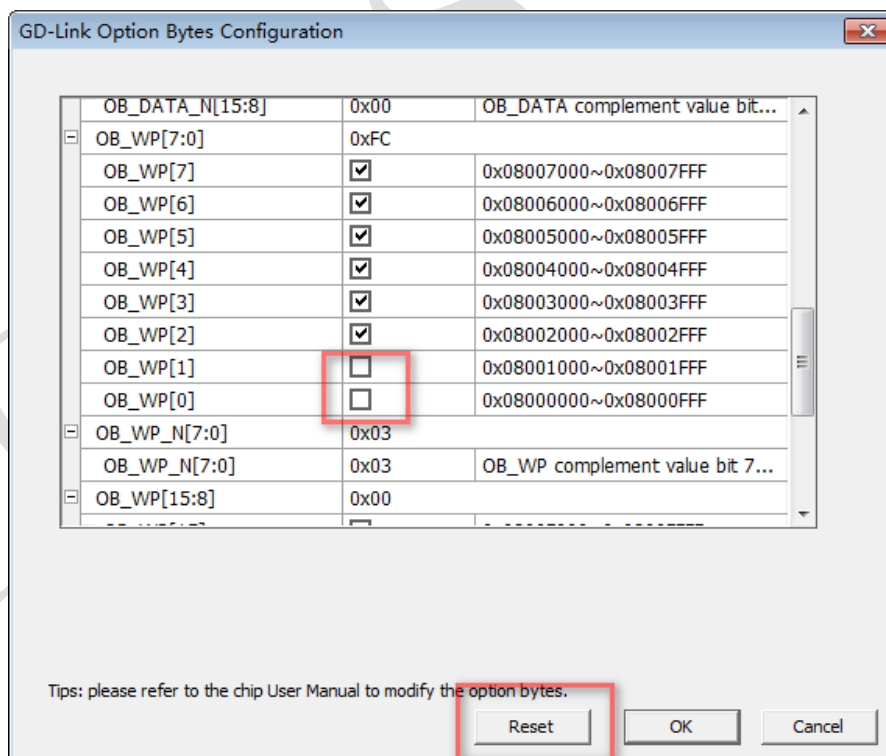


图 19 写保护被打开

## 4 固件更新

### 4.1 GDLink-S/Lite 更新方法

使用短路冒短接 GDLink-S 输出端子上的 TRESET（早期版本为 TDO）和 GND，然后插入电脑，安装驱动以后，设备管理器中会出现一个虚拟串口，使用“超级终端”连接该串口更新固件。可参考 UIMeter 使用 XBOOT 更新固件的视频：

<https://www.bilibili.com/video/av83660645>

注：执行 ymodem 命令会自动擦除固件，如果没有写入新固件导致设备无响应，重新执行 ymodem 命令写入固件即可。

如果您的设备使用正常，不建议进行固件升级操作。

### 4.2 GDLink-J 更新方法

GDLink-J 内置了 JLink-OB 和 GDLink-S 两种固件，需要单独更新。

- JLink-OB 固件起始地址 0x08004000，长度 48kB=0xC000
- GDLink-S 固件起始地址 0x08014000，长度 16kB=0x4000

进入 XBOOT 方法与普通版本相同，升级之前需要先擦除对应的固件。

升级 JLink-OB 固件时需要首先擦除 JLink-OB 固件，`erase 8004000 C000`，注意命令中的数字都是十六进制。擦除完成以后，输入 ymodem 命令，等待超级终端上出现字母`C`，选择菜单：传送->发送文件，弹出的对话框中选择好要升级的 hex 文件，协议选择 Ymodem，点击`发送`，等待发送完成即可。

升级 GDLink-S 固件时需要首先擦除 GDLink-S 固件，`erase 8014000 4000`，注意命令中的数字都是十六进制。擦除完成以后，输入 ymodem 命令，等待超级终端上出现字母`C`，选择菜单：传送->发送文件，弹出的对话框中选择好要升级的 hex 文件，协议选择 Ymodem，点击`发送`，等待发送完成即可。

如果您的设备使用正常，不建议进行固件升级操作。

## 5 更新记录

更新日期	更新类型	更新人	更新内容
2020/4/10	A	Echo	新建文档
2020/7/30	A	Echo	增加 XBOOT 中使用超级终端更新固件说明
2020/12/3	A	Echo	增加 GDLink-S 产品描述
2021/3/22	A	Echo	增加 GDLink-Lite 产品描述，完善文档
2021/4/30	A	Echo	补充常见问题
2021/5/27	A	Echo	增加 SWD 接口复位值说明
2021/9/3	A	Echo	增加写保护导致无法下载固件问题说明
2023/7/4	A	Echo	增加 GDLink-J 版本描述

注：

M-->修改

A -->添加

ECHO Studio 保留本文档最终解释权。

请使用 PDF 书签阅读本文档，快速定位所需内容！

项目主页：<https://github.com/xjtuecho/CMSIS-DAP>

国内镜像: <https://gitee.com/xjtuecho/CMSIS-DAP>

ECHO Studio