YSPROG FOR GD32 MCU 规格书

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，YSPROG FOR GD32 MCU 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。

YSPROG 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。

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 2YSPROG FOR GD32 MCU 正面外观

1.2 接口定义

YSPROG 输出使用 2x5 PIN 2.54 排针，排针定义见标签

SWD 调试通常只需要连接 5.0V、GND、CLK、DIO 四个管脚；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。

YSPROG 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 TDO/SWO 到 GND 后连接电脑，YSPROG 进入 IAP 模式，可升级固件，详见 3。

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。

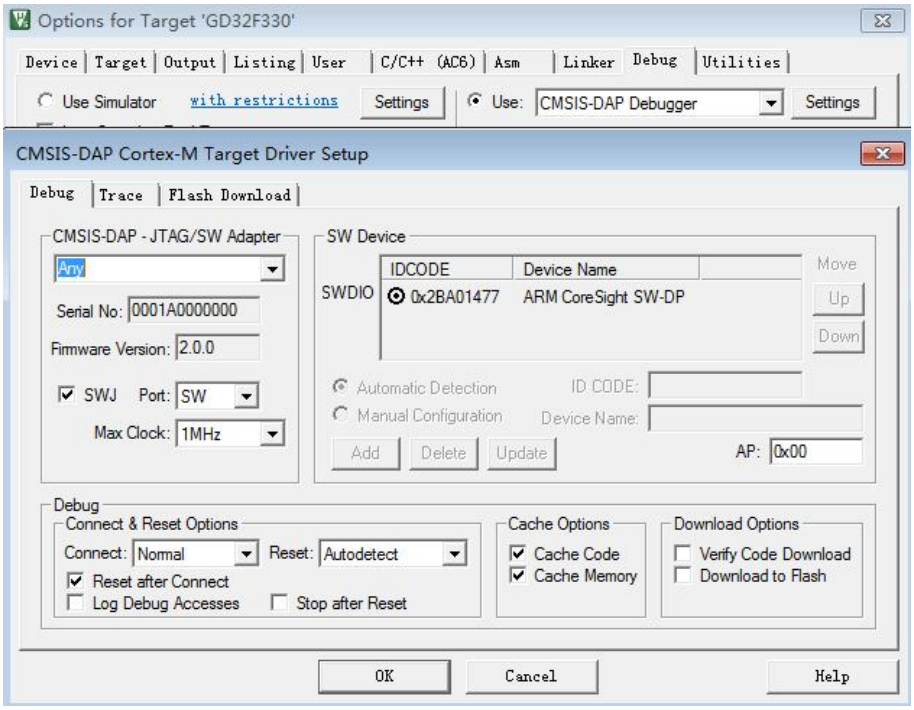

图 4 Cortex M4 内核的 GD32F330 调试设置

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

注意此时调试器的 TRESET 管脚要连接目标芯片的 NRST 管脚，M0/M3/M4 内核不需要。

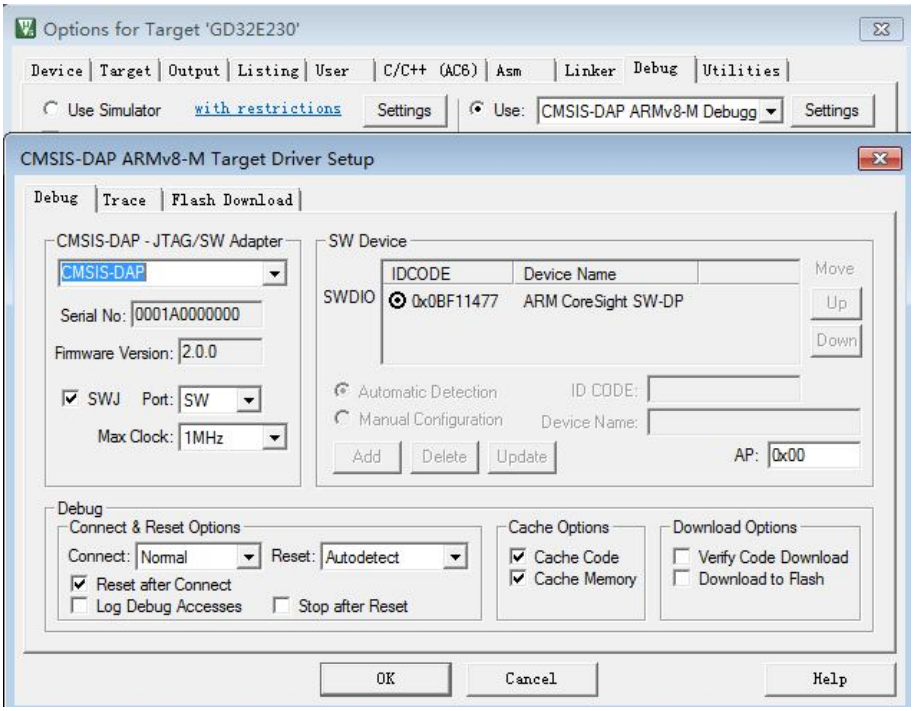

图 5 Cortex M23 内核的 GD32E230 调试设置

2.2 GD-Link Programmer

目前 YSPROG 支持的的 GD-Link Programmer 版本为 3.0.0.5950。

GD-Link Programmer 最新版本为 4.3.7.9954，实测该版本配合官方 GDLink 无法连上一些目标芯片，原因不明，因此 YSPROG 暂未适配 GD-Link Programmer 4.3.7.9954。

YSPROG 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

使用 YSPROG 读取整片闪存见图 6。

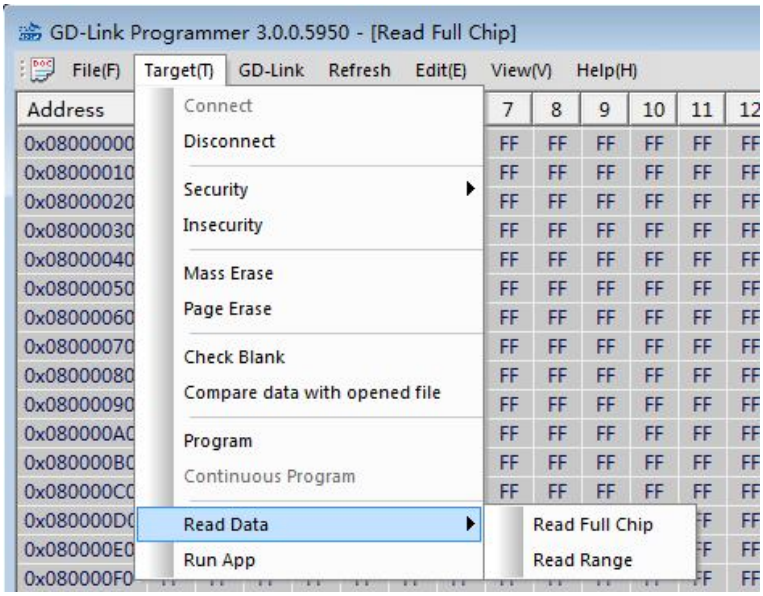

图 6 使用 GDLink-Lite 读取整片闪存

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 10。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（YSPROG 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

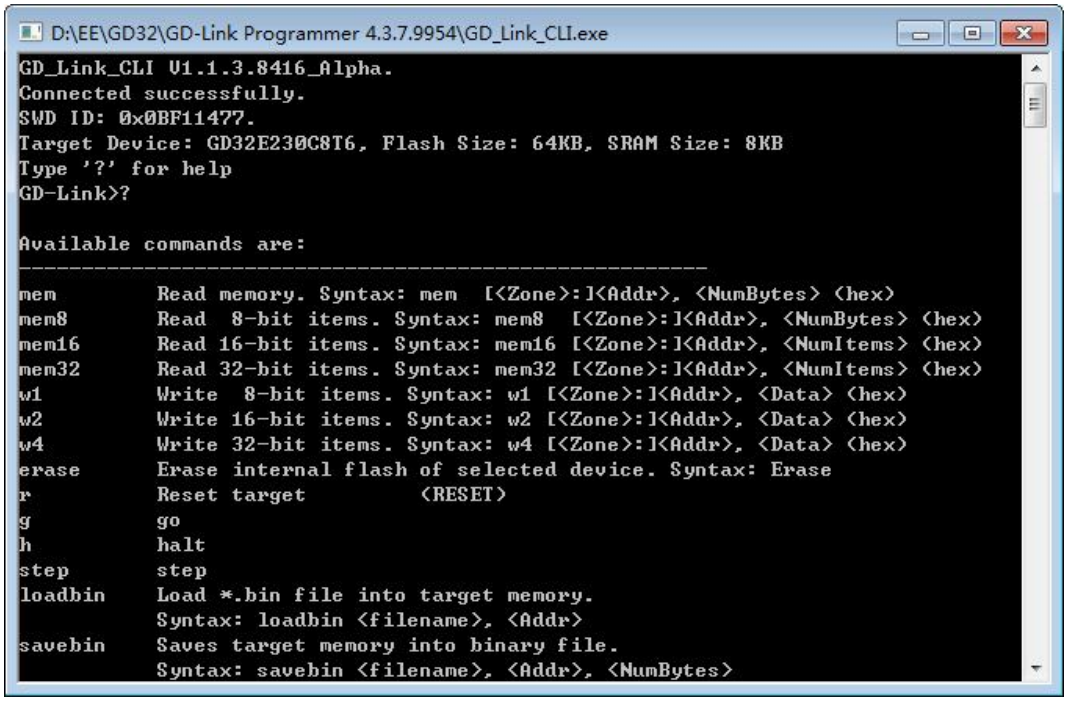

图 10 GD_Link_CLI 运行界面

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

YSPROG 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 11 pyocd -h 命令输出

使用"pyocd list"命令列出当前的调试器信息：

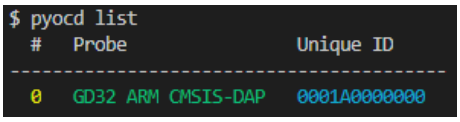

图 12 pyocd list 查看当前调试器信息

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 13。

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

图 13 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 14。

```
$ 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), skipp
```

图 14 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 15。

```
$ 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 15 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 16 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-Lite/Scripts

```
#!/bin/sh

TARGET="gd32e230c8"
PACK_NAME="$PACKS/GigaDevice.GD32E230_DFP.1.0.0.pack"

Usage() {
    echo "Usage: $0 [erase | flash | info]."
}

if [ $# -eq 0 ]; then
    Usage
else
    case $1 in
    erase)
        echo "Erase Device..."
        pyocd erase --chip --target $TARGET --pack=$PACK_NAME
    ;;
    flash)
        if [ $# -eq 2 ]; then
            echo "Flash Device..."
            pyocd flash $2 --target $TARGET --pack=$PACK_NAME
        else
            Usage
            echo "Usage: $0 erase [APP.hex]"
        fi
    ;;
    info)
        echo "Try unlock Device..."
        pyocd-flashtool --unlock --target $TARGET --pack=$PACK_NAME
    esac
fi
```

图 16 一个擦除、编程 GD32E230 的脚本

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，YSPROG 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 YSPROG 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 YSPROG 暂时只支持 GDLink Programmer 3.0.0.5950 版本。

使用 GDLink Programmer 4.3.7.9954 会弹出图 17 提示，无法使用。

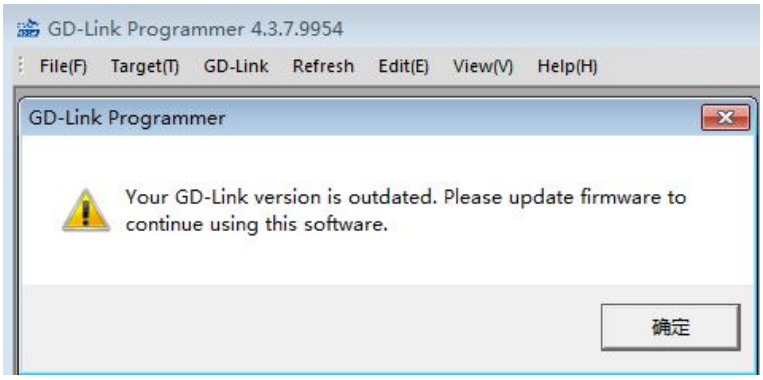

图 17 4.3.7.9954 版本提示

由于实测官方 GDLink 使用 4.3.7.9954 版本无法连上 GD32F150/GD32E230，原因未知，YSPROG 暂未适配 4.3.7.9954 版本，请使用 3.0.0.5950 版本。

4.3.7.9954 版本自带了一个命令行模式程序：GD_Link_CLI.exe，与 YSPROG 配合工作良好，详情见 2.3。

目前 YSPROG 有测试固件，可支持 GDLink Programmer 4.3.7.9954 和 4.5.1.10871，但是没有经过全面测试，客户如果需要可以提前提出，普通用户仍然建议使用 3.0.0 版本。

3.4 MDK 无法调试？

MDK 无法调试时，先尝试使用 GDLink Programmer 3.0.0 来连接，排除 MDK 软件配置问题。

MDK 调试时，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？

YSPROG 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，因此不支持，RISC-V 是 Cortex-M 的竞争对手。

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 YSPROG 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。

4 固件更新

使用短路冒短接 YSPROG 输出端子上的 TDO/SWO 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。