life.augmented

UM2237
用户手册

STM32CubeProgrammer软件说明

引言

STM32CubeProgrammer（STM32CubeProg）为任意环境下的STM32器件编程提供了一个一体化的软件工具：多操作系统，图形用户界面或命令行界面，支持多种连接选择（JTAG、SWD、USB、UART、SPI、CAN和I2C），采用手动操作或通过脚本自动操作。

本文档详细介绍了硬件和软件环境先决条件，以及可用的STM32CubeProgrammer软件功能。

目录

表格索引

图片目录

1 入门指南

本节介绍安装STM32CubeProgrammer软件工具的要求和步骤。

STM32CubeProgrammer支持基于Arm®(a)Cortex®-M处理器的STM3232位MCU和基于Arm® Cortex®-A处理器的STM32 32位MPU。

1.1 系统要求

支持的操作系统和架构为：

- Linux® 64位
- Windows® 7/8/10/11 32位和64位
- macOS®（最小版本OS X® Yosemite）

自版本2.6.0起，无需安装任何Java™SERunTimeEnvironment。在STM32CubeProgrammer运行时，使用下载的软件包中提供的绑定JRE，不再使用您的计算机上安装的JRE。

注： *绑定JRE为Liberica 8.0.265。*

对于macOS软件，最低要求如下

- Xcode®必须安装在macOS计算机上
- Xcode®和Rosetta®必须安装在内置Apple® M1处理器的macOS计算机上

支持的最小屏幕分辨率为1024x768。

1.2 安装STM32CubeProgrammer

本节介绍使用STM32CubeProgrammer软件的要求和步骤。该装置还提供了“STM32 trusted package creator”工具的可选安装，该工具可用来创建安全的固件文件，用于安全的固件安装与更新。请参考*STM32 Trusted Package Creator工具软件说明*（UM2238）*（可从ST网站www.st.com下载）*了解更多信息。

1.2.1 Linux安装

如果使用USB端口连接STM32器件，则输入以下指令安装libusb1.0软件包：

sudo apt-get install libusb-1.0.0-dev

要使用ST-LINK工具或USB DFU连接到目标，需要将位于*Driver/rules* 文件夹下的规则文件复制在Ubuntu上的*/etc/udev/rules.d/*文件夹中（*"sudo cp *.* /etc/udev/rules.d"*）。

注： *需要使用libusb1.0.12版本或更高版本来运行STM32CubeProgrammer。*

为了安装STM32CubeProgrammer工具，在Linux计算机上用STM32CubeProg-Linux产品编号从网站下载zip压缩包并解压缩，并执行*SetupSTM32CubeProgrammer-vx.y.z.linux*，它将指导您完成安装过程。Ubuntu 20 STM32CubeProgrammer图标默认不启用。如需启用，右键单击图标并选择*“允许启动”*。

a. Arm是Arm Limited（或其子公司）在美国和/或其他地区的注册商标。

1.2.2 Windows安装

为了安装STM32CubeProgrammer工具，从STM32CubeProg-Win-32bits或STM32CubeProg-Win-64bits（分别适用于Windows 32位和Windows 64位操作系统）下载zip压缩包并解压缩，并执行*SetupSTM32CubeProgrammer-vx.y.z.exe*，它将指导您完成安装过程。

1.2.3 macOS安装

为了安装STM32CubeProgrammer工具，用STM32CubeProg-Mac产品编号从网站下载zip压缩包并解压缩，并执行*SetupSTM32CubeProgrammer-vx.y.z.app*，它将指导您完成安装过程。

注： *如果安装失败，则在CLI模式下使用指令./SetupSTM32CubeProgrammer-x.y.z.app/Contents/MacOs/SetupSTM32CubeProgrammer-x_y_z_macos进行启动。*

确保拥有管理员权限，然后双击*SetupSTM32CubeProgrammer-macos*安装文件启动安装向导。

如果发生错误，则尝试以下修复方式中的一种：

- *$sudo xattr -cr ~/SetupSTM32CubeProgrammer-macos.app*
- 用指令启动.exe文件
  sudo java -jar SetupSTM32CubeProgrammer-2.7.0.exe。

1.2.4 DFU驱动程序

如果您在USB DFU模式下使用STM32器件，则需要通过运行*"STM32 Bootloader.bat"*文件来安装STM32CubeProgrammer的DFU驱动程序。该驱动程序随发布包提供，可在DFUDriver文件夹中找到。

如果您的计算机上安装了DFUSE驱动程序，则首先卸载程序，然后重启计算机并运行前面提到的".bat"文件。为避免插入板件时和之后重复安装旧版驱动程序，必须选中"删除此器件的驱动程序软件"选项。

图1. 删除旧版驱动程序软件

图2. 具有DfuSe驱动程序的STM32 DFU器件

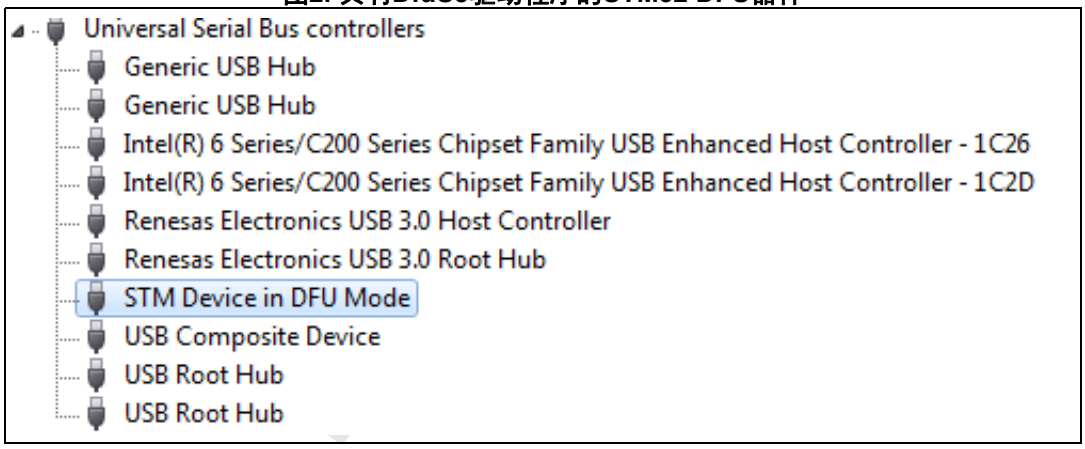
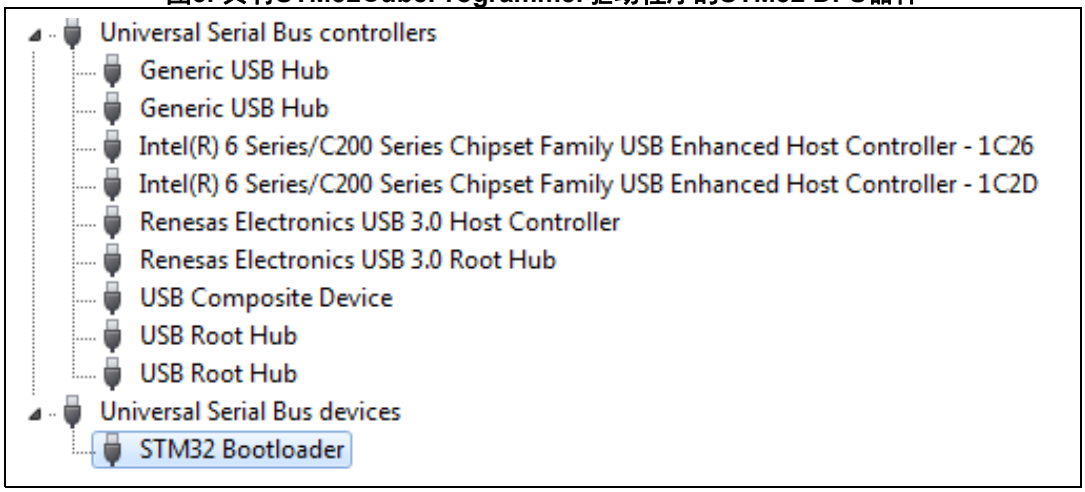


图3. 具有STM32CubeProgrammer驱动程序的STM32 DFU器件



注： *在Windows 7计算机上使用USB DFU接口或STLink接口时，确保更新了所有USB 3.0控制器驱动程序。老版本的驱动程序可能存在错误，该错误会阻止访问或导致USB设备连接问题。*

1.2.5 ST-LINK驱动程序

为了使用ST-LINK/V2、ST-LINKV2-1或ST-LINK-V3通过调试接口连接到STM32器件，运行*“stlink_winusb_install.bat”*文件以便安装ST-LINK驱动程序。该驱动程序随发布包提供，可在*“Driver/stsw-link009_v3”*文件夹中找到。

1.3 更新

STM32CubeProgrammer更新程序使用户能够执行软件及其相关软件包的自动更新。支持的所有操作系统（即Windows 10/11、Linux和macOS）均可使用该更新程序。

1.3.1 更新步骤

1. 检查连接，必要时更新连接设置
2. 检查更新
3. 下载新版本
4. 安装下载的版本（更新后工具重启）

1.3.2 代理设置

用户可使用“代理设置”窗口手动检查连接，该窗口可通过帮助按钮中提供的子菜单打开（参见*图 4*）。有三种不同的代理设置可供选择（*图 5*）：

- 无代理
- 使用系统参数
- 使用服务器的手动配置：输入HTTP代理名称、端口和凭证

图4. 代理设置子菜单

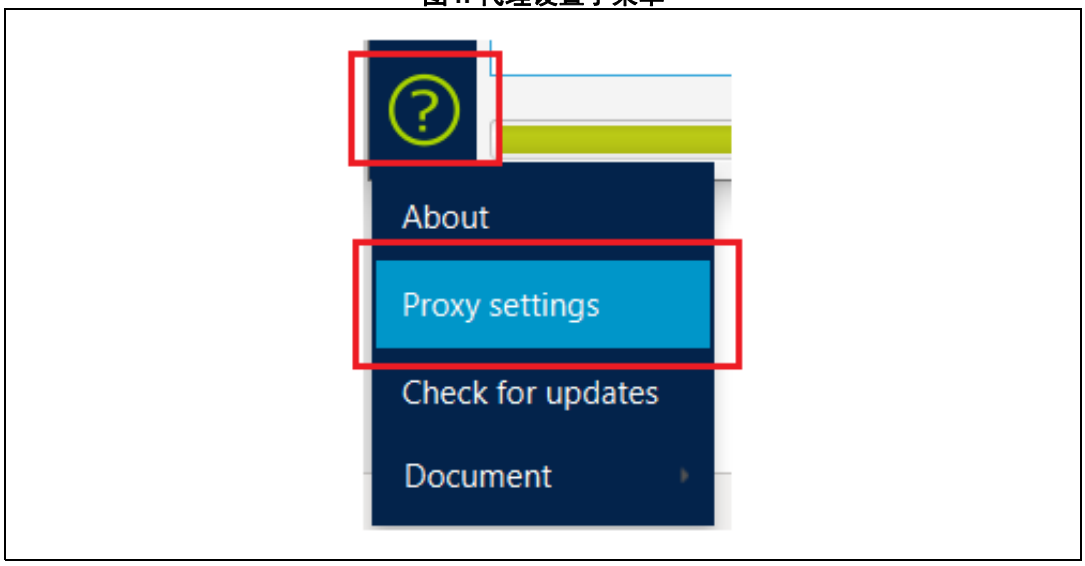
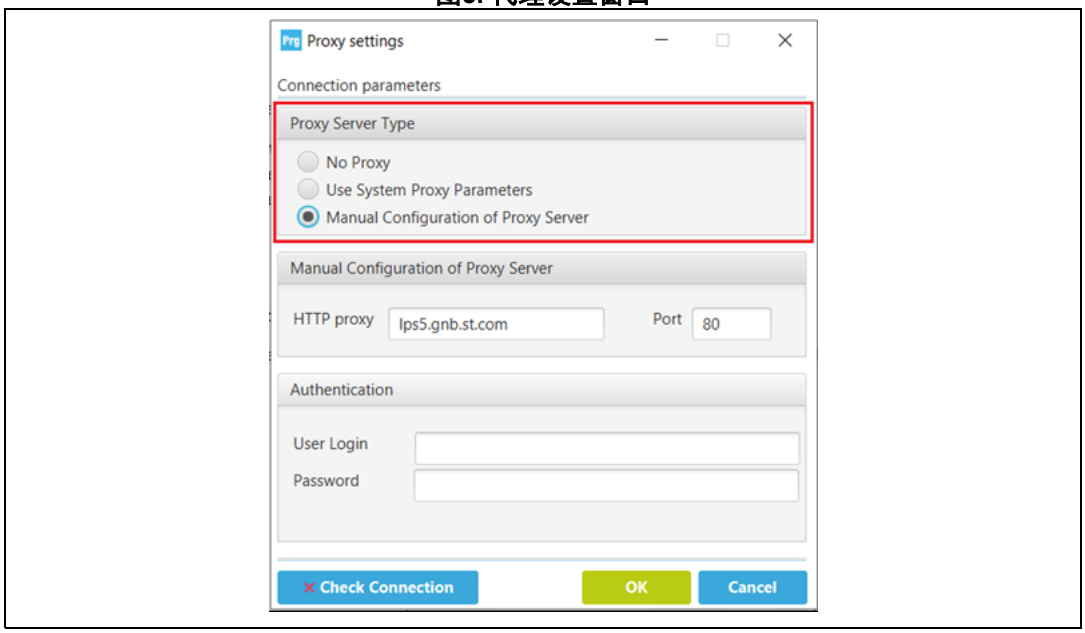


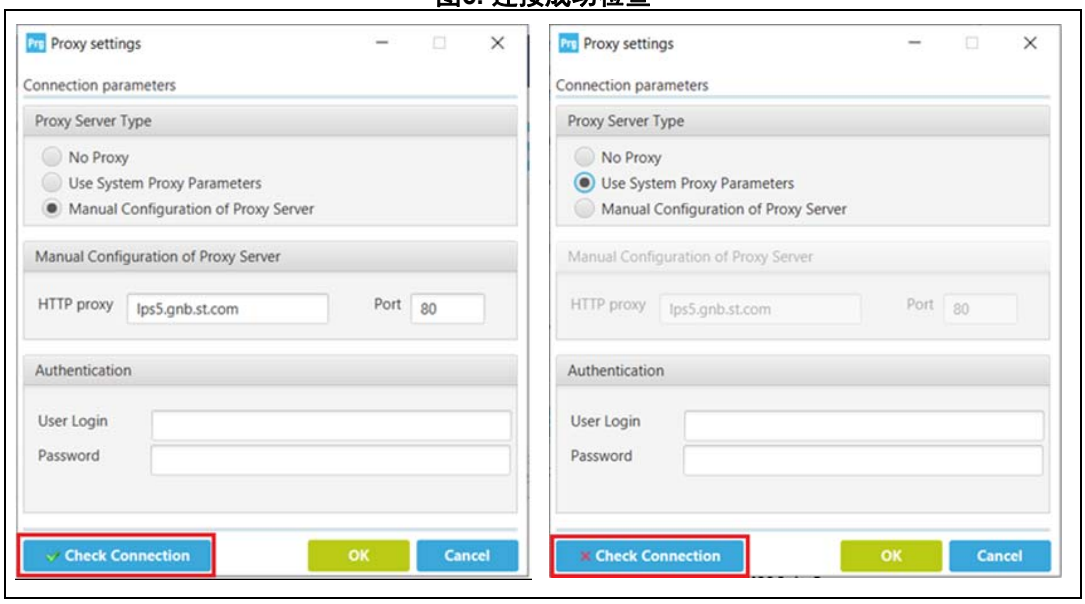
图5. 代理设置窗口



连接检查的状态显示在“检查连接”按钮上：

- 绿色图标表示成功（*图 6*的左侧）。
- 红色图标表示连接断开（*图 6*的右侧）。

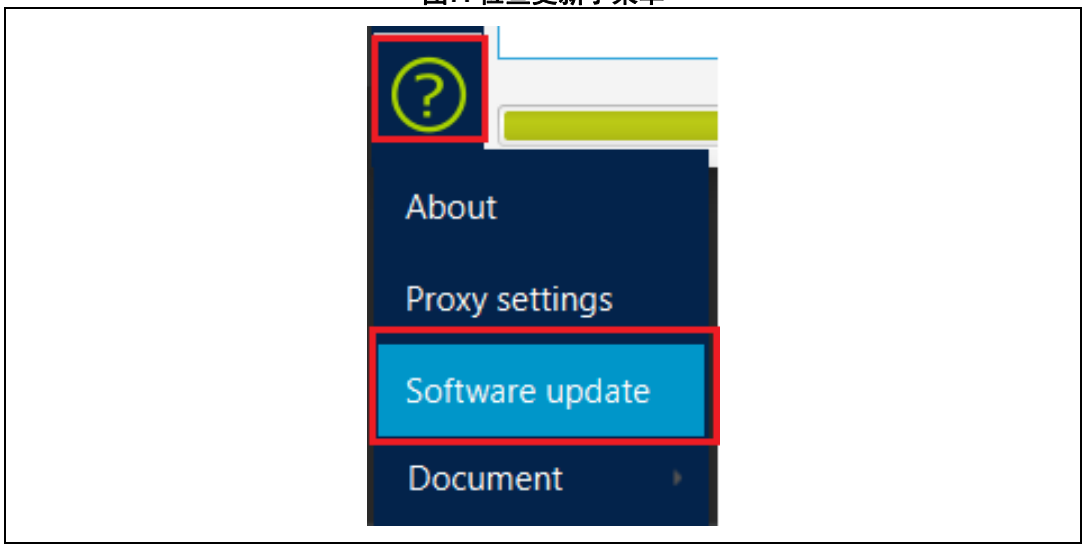
图6. 连接成功检查



1.3.3 检查更新

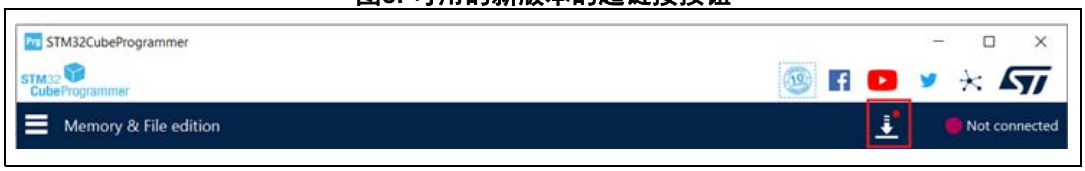
用户可使用“更新程序”窗口启动更新过程，该窗口可通过帮助按钮中增加的“软件更新”子菜单打开。

图7. 检查更新子菜单



如果有新版本可用，则主菜单中将显示更新按钮（*图 8*）。

图8. 可用的新版本的超链接按钮



注： *如果用户已经更新了STM32CubeProgrammer，则启动时将不再显示该超链接按钮。*

如果有新版本可用，则用户可使用更新程序窗口进行更新。

该窗口显示：

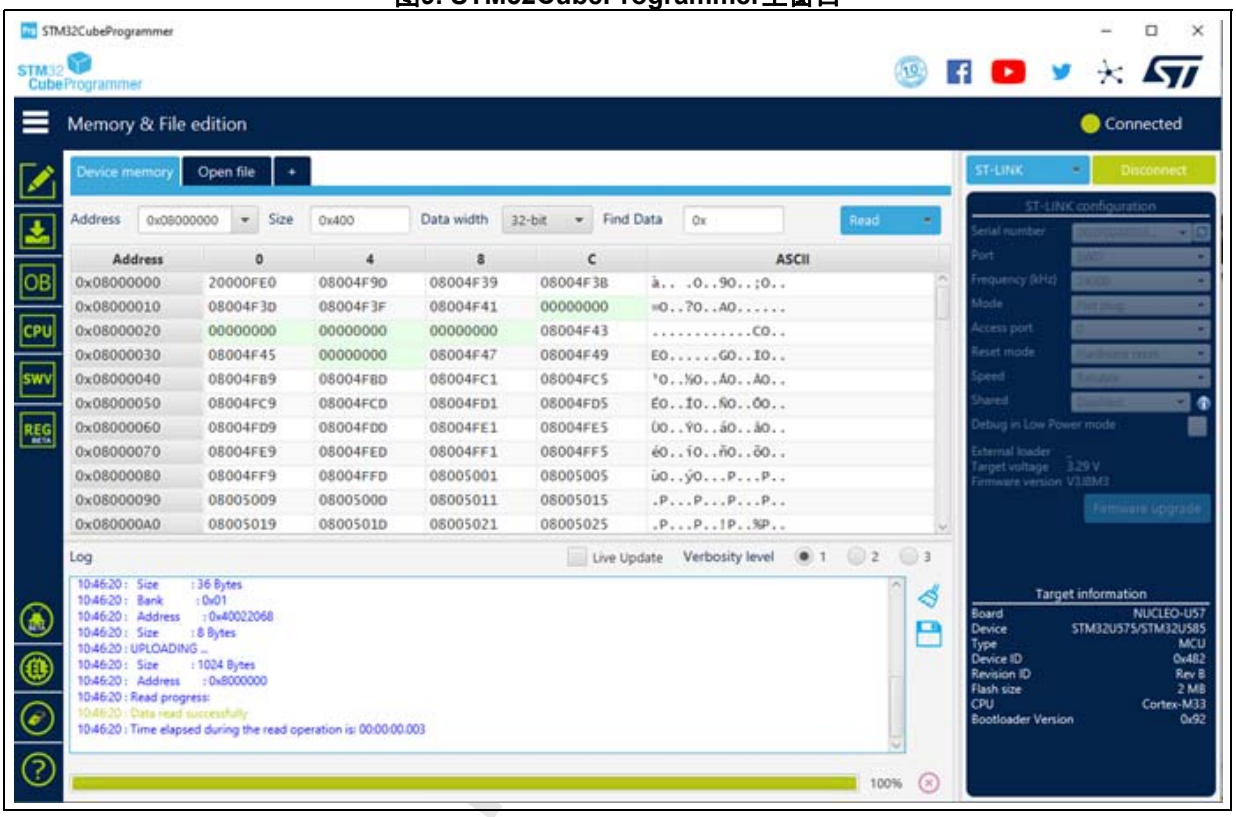
- STM32CubeProgrammer的当前版本
- STM32CubeProgrammer服务器中的可用版本
- 变更日志（包含新软件包中发布的主要变更）
- 授权
- 上一次更新（包含上一次更新的日期或消息“以前未执行更新”）
- 更新程序工具的当前版本
- 刷新按钮（用于检查是否有新版本）
- 关闭按钮（用于停止新版本的安装过程）

注： *需要管理员权限才能下载新软件包。更新完毕后，更新程序窗口只显示新版本。*

2 MCU的STM32CubeProgrammer用户界面

2.1 主窗口

图9. STM32CubeProgrammer主窗口

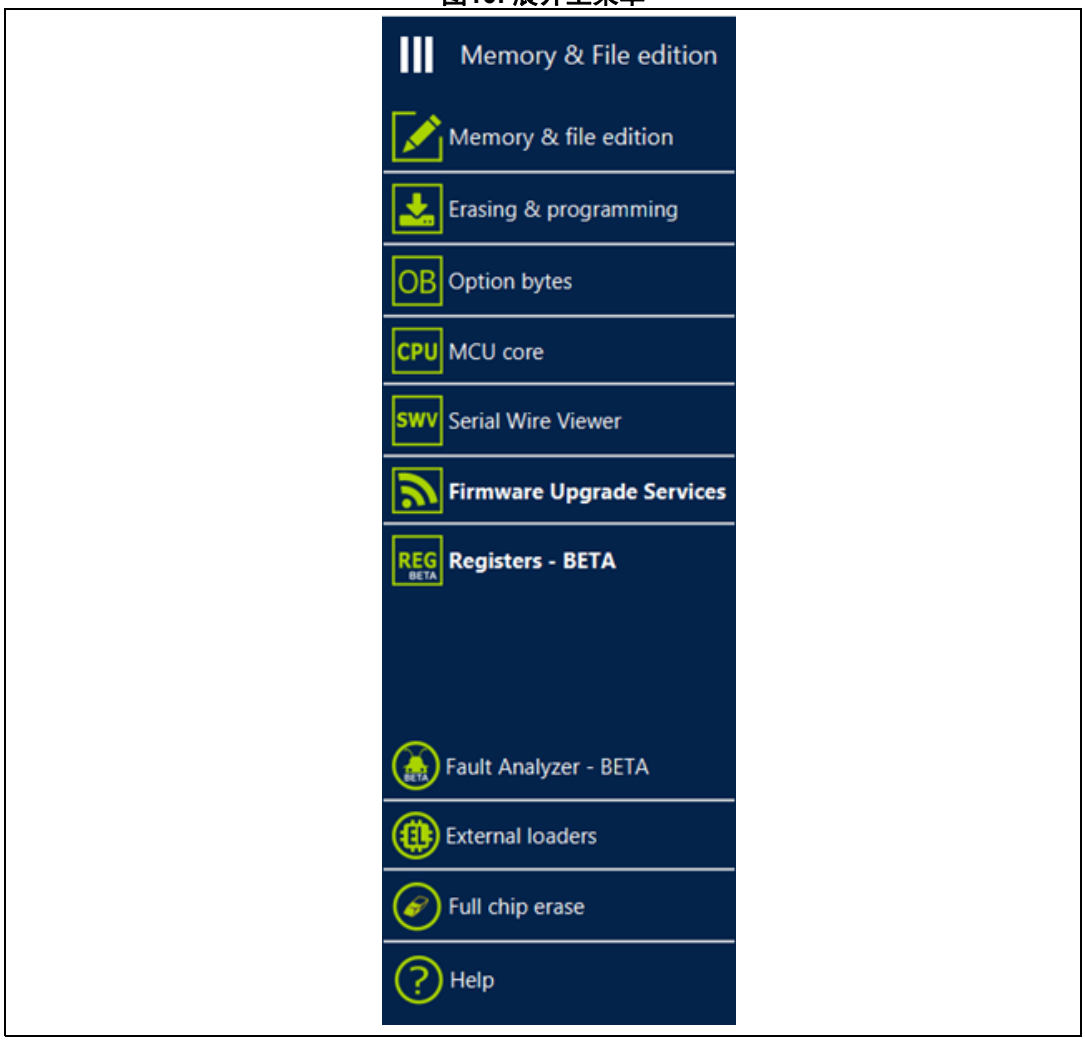


下面几节将描述主窗口的组成部分。

2.1.1 主菜单

用户可通过主菜单在“存储器和文件编辑”、“存储器编程和擦除”以及“选项字节工具”这三个主面板之间切换。将基于使用的器件显示其他面板。点击左上角的侧边栏菜单（三线按钮），菜单展开并显示文字说明，如*图 10*所示。

图10. 展开主菜单



2.1.2 日志面板

显示错误、警告和与该工具执行操作相关的信息性事件。使用日志文本区域上方的详细程度单选按钮可以细化显示消息的详细程度。最低详细级别为1，最高级别为3，此级别下可记录所选接口上的所有事务。所有显示的信息都使用格式“hh:mm:ss:ms”进行时间戳标记，其中“hh”为小时，“mm”为分钟，“ss”为秒，“ms”为毫秒（三位数字显示）。

日志面板的右侧有两个按钮，第一个用于清理日志，第二个用于将日志保存到日志文件。

2.1.3 进度条

进度条显示工具完成的任何操作或事务的进程（例如读取、写入、擦除）。点击进度条前的“停止”按钮即可中止任何正在进行的操作。

2.1.4 目标配置面板

这是连接目标之前要查看的第一个面板。它允许用户选择目标接口；可以使用ST-LINK调试工具的调试接口，或者使用经过UART、USB、SPI、CAN或I2C的引导装载程序接口。

刷新按钮允许您检查连接到PC的可用接口。在选择ST-LINK接口时按下此按钮，该工具将检查所连接的ST-LINK工具，并在序列号组合框中将其列出。如果选择了UART接口，它将检查PC的可用COM端口，并将它们在端口组合框中列出。如果选择了USB接口，它会检查连接到PC的DFU模式下的USB设备，并在端口组合框中将其列出。每个接口都有自己的设置，需要在连接之前进行设置。

ST-LINK设置

图11. ST-LINK配置面板

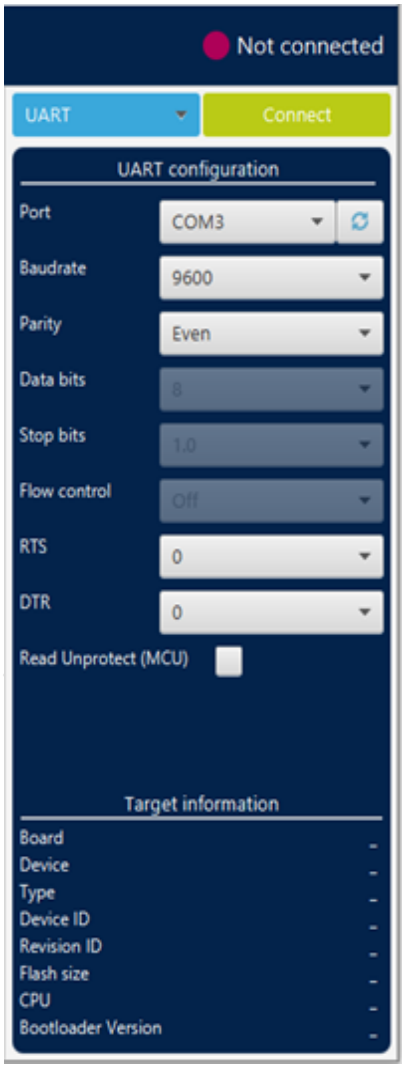
- **序列号**：该字段包含所有连接的ST-LINK工具的序列号。用户可以根据其序列号选择其中一个。
- **端口**：ST-LINK工具支持两种调试协议：JTAG和SWD。

注： *JTAG并不适用于安装在STM32 Nucleo或Discovery板上的所有嵌入式ST-LINK。*

- **频率**：JTAG或SWD时钟频率
- **访问端口**： 选择要连接的访问端口。大多数STM32设备只有一个访问端口，即Access端口0。
- **模式**：
  - **正常**：使用“正常”连接模式时，目标先复位然后停机。使用“复位模式”选项来选择复位的方式。
  - **复位下连接**：该模式允许在执行指令之前使用复位向量捕获连接到目标。这在很多情况下是很有用的，例如当目标包含了禁用JTAG/SWD引脚的代码时。
  - **热插拔**：可在不停机或不复位的情况下连接到目标。这对于在应用运行时更新RAM地址或IP寄存器非常有用。
  - **掉电**：允许将目标置于调试模式，即使自目标上电后应用尚未启动。硬件复位信号必须连接在ST-Link与目标之间。此特性在某些使用STMPS2141电源开关的板件（MB1360、MB1319、MB1361和MB1355）上可能不完全有效。
- **复位模式**：
  - **软件系统复位**：通过Cortex-M应用中断和复位控制寄存器（AIRCR）来复位除调试以外的所有STM32组件。
  - **硬件复位**：通过nRST引脚来复位STM32器件。JTAG连接器（引脚15）的RESET引脚须连接到器件复位引脚。
  - **内核复位**：通过AIRCR仅将内核Cortex-M复位。
- **速度**（仅Cortex- M33）：
  - **可靠**：允许用户以慢速模式连接。
  - **快速**：允许用户以快速模式连接。
- **共享**：启用共享模式可将STM32CubeProgrammer或其他调试器的两个或更多实例连接到同一ST-LINK工具。
- **在低功耗模式下调试**（仅STM32U5/WB/L4系列）：将DBGMCU_CR中的位置为1。
- **外部加载程序**：显示在“外部加载程序”面板（可从主菜单（侧边栏菜单）访问）中选择的外部存储器加载程序的名称。
- **目标电压**：在这里测量和显示目标电压。
- **固件版本**： 显示ST-LINK固件版本。固件升级按钮可以升级ST-LINK固件。

UART设置

图12. UART配置面板



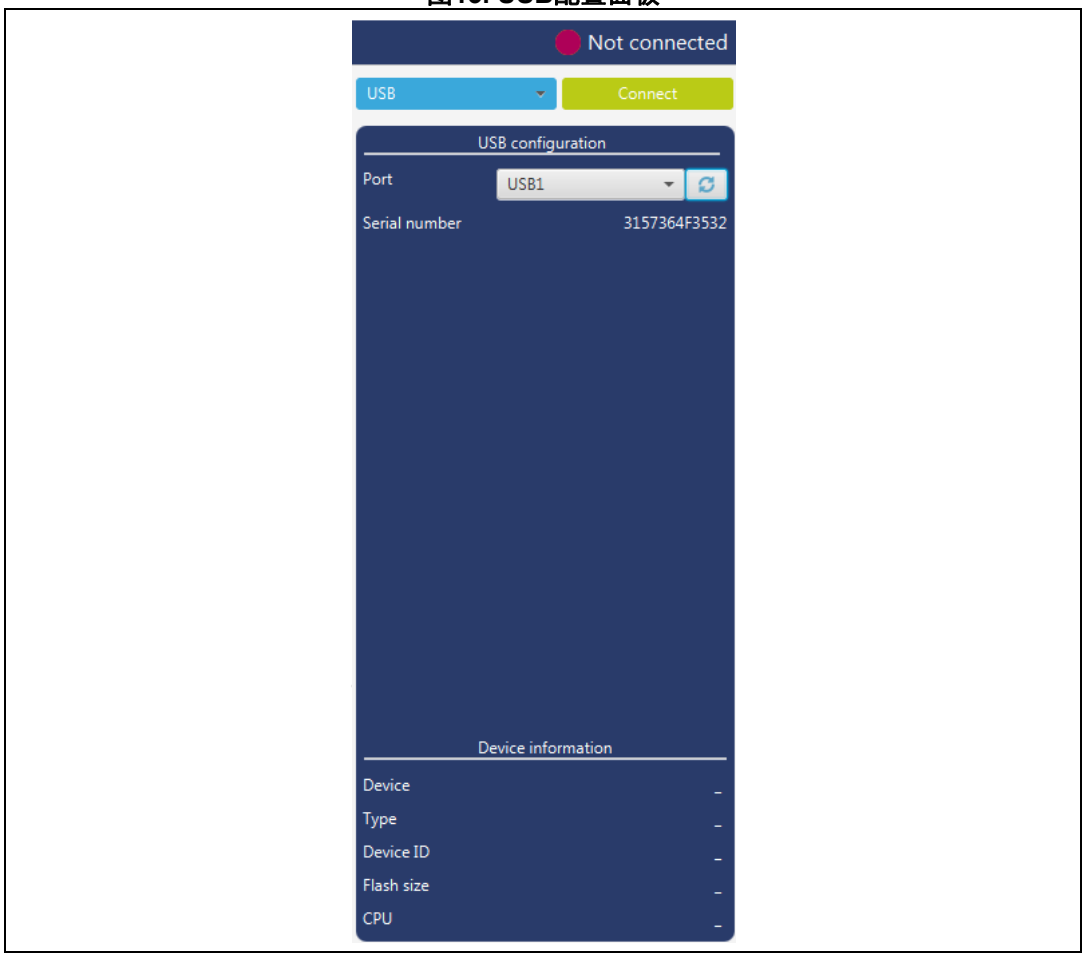
- **端口**：选择目标STM32所连接的COM端口。使用刷新按钮重新检查PC上可用的COM端口。

注： *必须使用启动引脚和/或选项位，使得STM32在引导加载程序模式下启动。请参阅"STM32微控制器系统存储器启动模式"（AN2606）（可从ST网站www.st.com获取）了解关于STM32引导加载程序的更多信息。*

- **波特率**： 选择UART波特率。
- **奇偶校验**： 选择奇偶校验（偶，奇，无）。对于所有的STM32器件，奇偶校验须为"偶"。
- **数据位**： 须始终为8。STM32仅支持8位数据。
- **停止位**： 须始终为1。STM32仅支持1位停止位。
- **流控制**： 须始终关闭
- **RTS**（请求发送）：将COM RTS引脚置为高或低电平。
- **DTR**（数据终端就绪）：将COM DTR引脚置为高或低电平。

USB设置

图13. USB配置面板



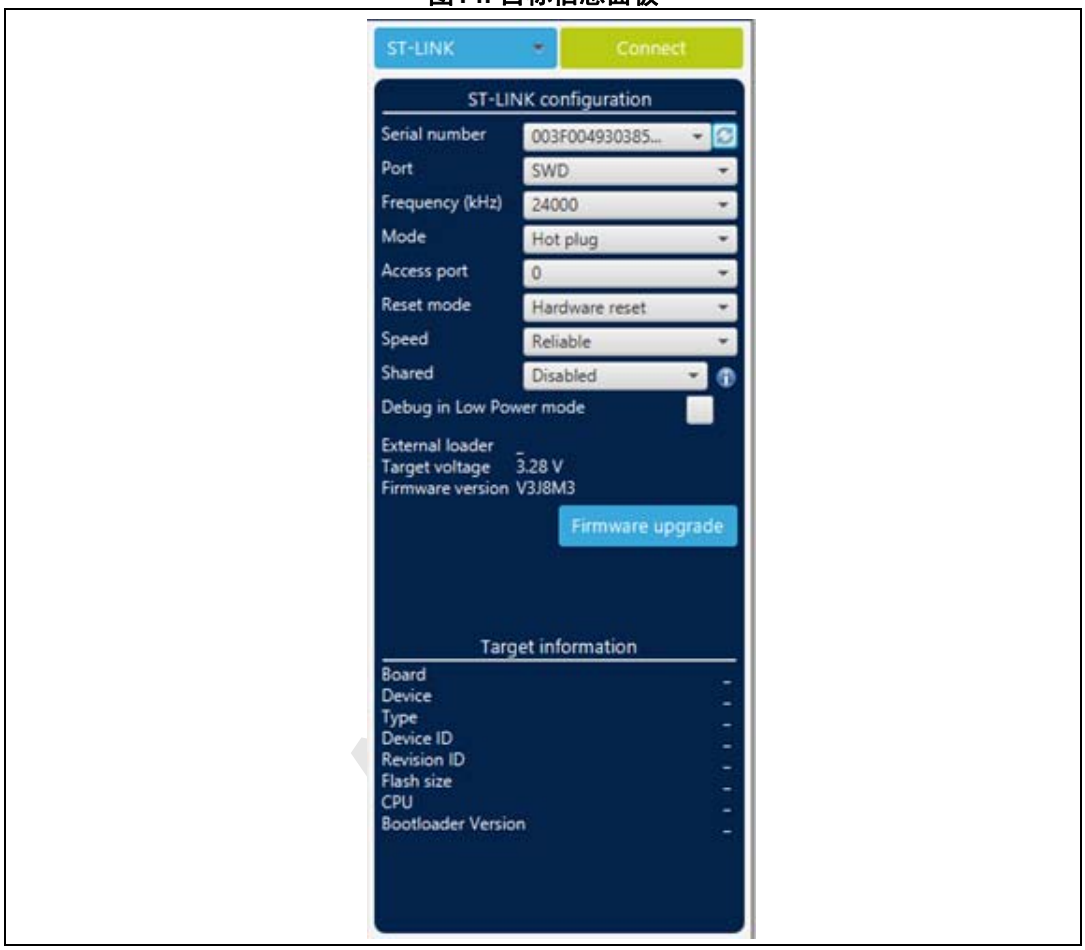
- **端口**：选择DFU模式下连接到PC的USB设备。您可以使用刷新按钮重新检查可用设备。

注： *必须使用启动引脚和/或选项位，使得STM32在引导加载程序模式下启动。请参阅AN2606（可从ST网站www.st.com获取）了解关于STM32引导加载程序的更多信息。*

设置了正确的接口设置后，点击“连接”按钮即可连接到目标接口。如果连接成功，按钮上方的指示灯会变成绿色。

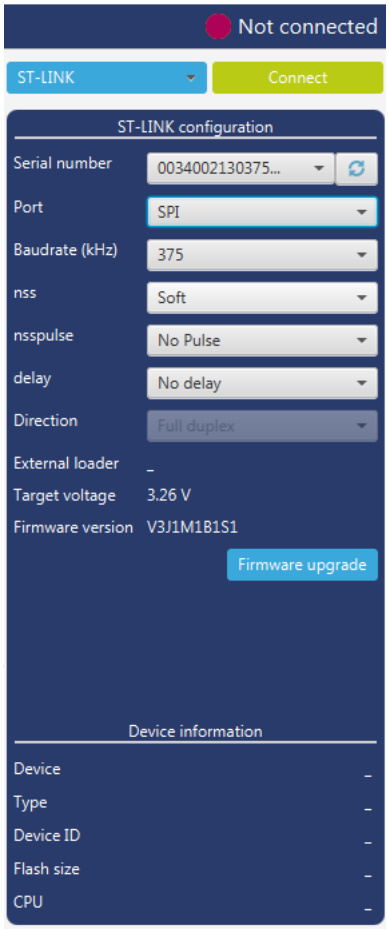
连接后，目标信息将在设置区下面的设备信息区中显示，然后该设置区被禁用，如*图 14*所示。

图14. 目标信息面板


ST-LINK
Connect
ST-LINK configuration
Serial number
003F004930385...
Port
SWD
Frequency (kHz)
24000
Mode
Hot plug
Access port
0
Reset mode
Hardware reset
Speed
Reliable
Shared
Disabled
Debug in Low Power mode
External loader
Target voltage
3.28 V
Firmware version
V3J8M3
Firmware upgrade
Target information
Board
Device
Type
Device ID
Revision ID
Flash size
CPU
Bootloader Version

SPI设置

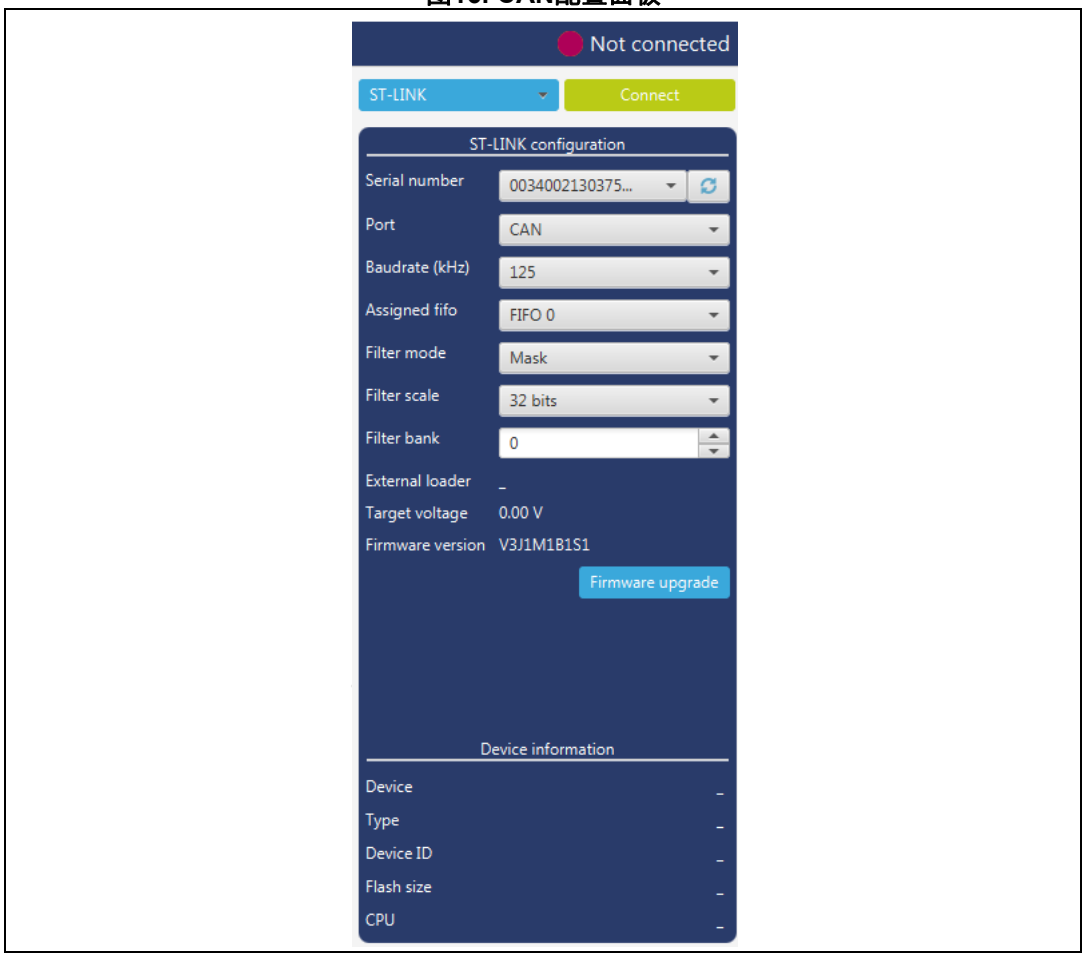
图15. SPI配置面板



- **序列号**：该字段包含使用SPI引导加载程序时连接的所有ST-LINK-V3工具的序列号。
- **端口**：选择连接到PC的USB设备。您可以使用刷新按钮重新检查可用设备。
- **波特率**：选择SPI波特率。
- **nss**：从设备选择软件或硬件。
- **nsspulse**：当有一段时间的连续传输时，从设备选择信号能以脉冲模式工作，主设备在数据帧之间基于nss输出信号生成脉冲，并持续一个SPI时钟周期。
- **延迟**：用于在数据之间插入若干微秒的延迟。
- **方向**：必须始终为双工，使用两条数据线路且数据同时沿两个方向流动。

CAN设置

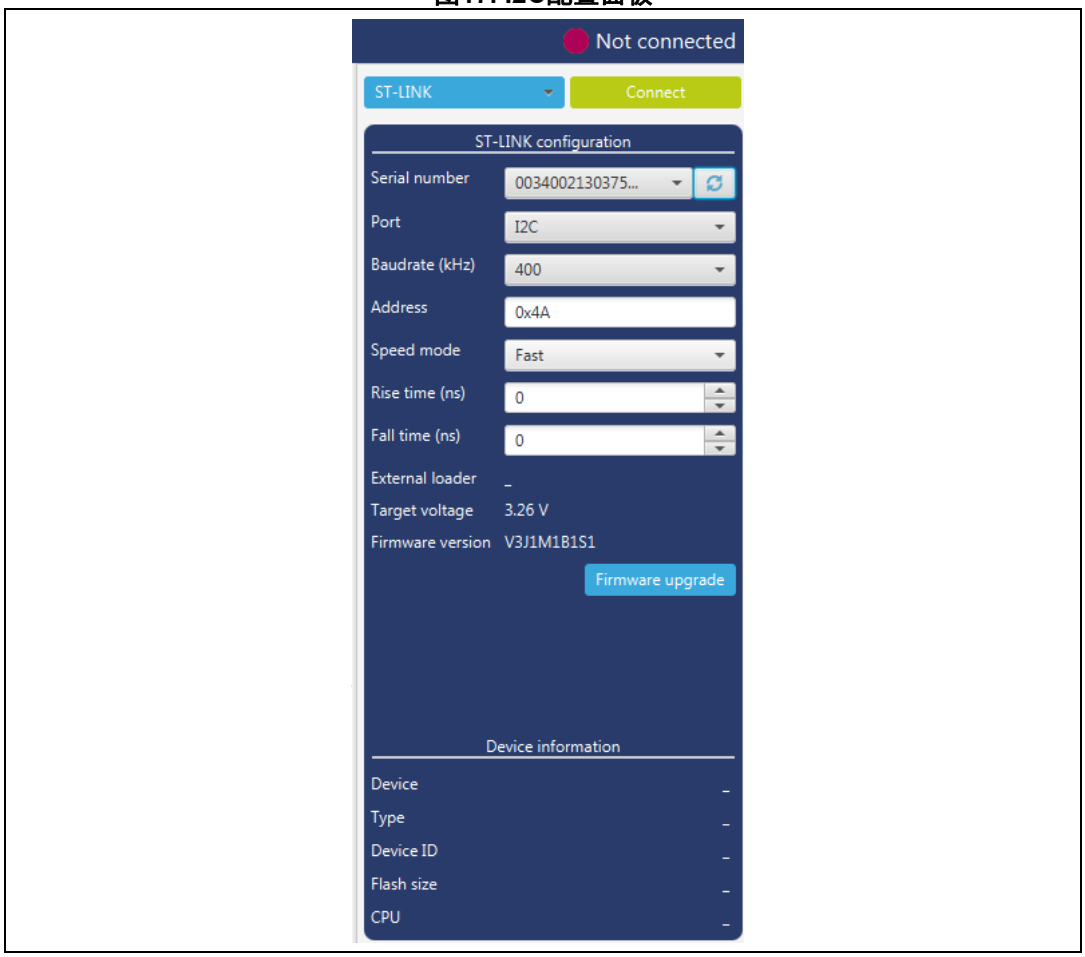
图16. CAN配置面板



- **序列号**：该字段包含使用CAN引导加载程序时连接的所有ST-LINK-V3工具的序列号。
- **端口**：选择连接到PC的CAN设备。您可以使用刷新按钮重新检查可用设备。
- **波特率**：选择CAN波特率。
- **分配的FIFO**：选择用于存储传入消息的接收FIFO存储器。
- **滤波器模式**：选择滤波器类型：屏蔽或倾斜。
- **滤波器范围**：选择滤波器组的宽度：16或32位。
- **滤波器组**：值为0至13，用于选择滤波器组编号。

I2C设置

图17. I2C配置面板



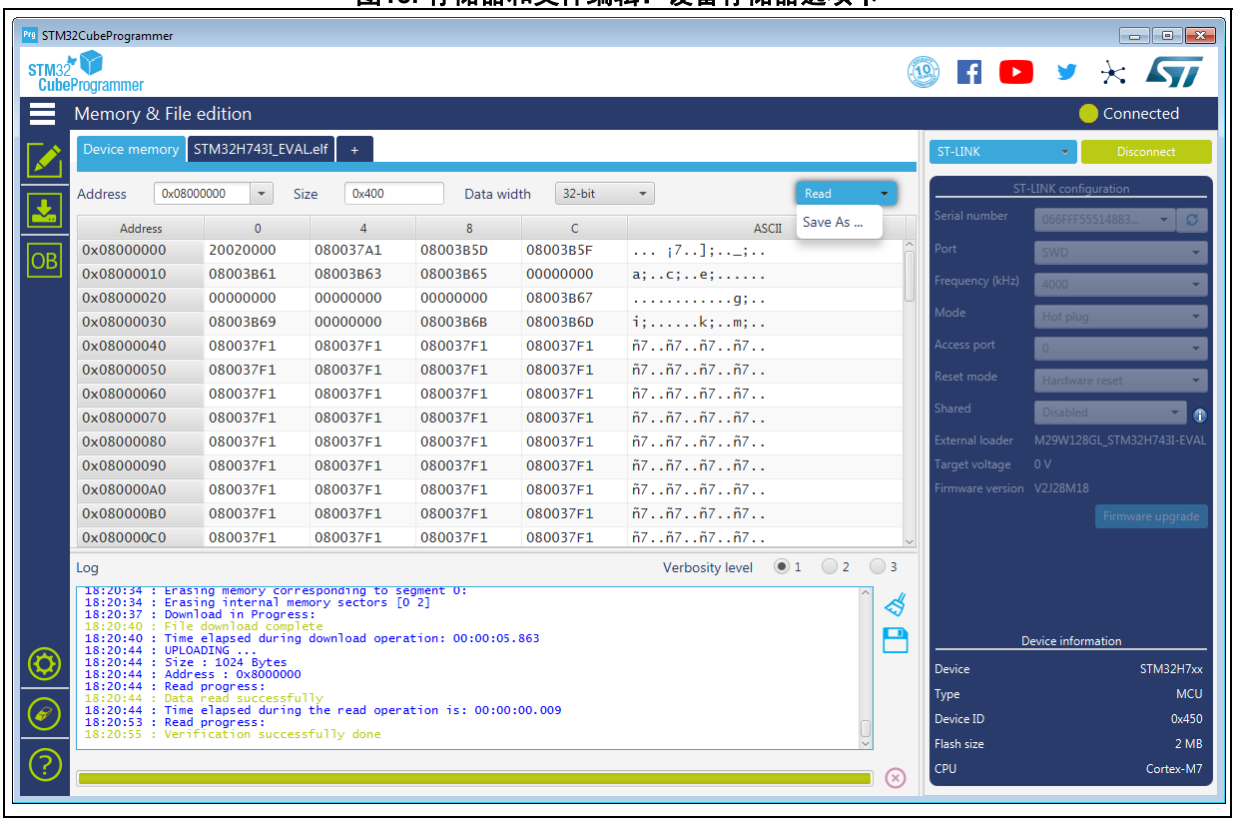
- **序列号**：该字段包含使用I2C引导加载程序时连接的所有ST-LINK-V3工具的序列号。
- **端口**：选择连接到PC的I2C设备。您可以使用刷新按钮重新检查可用设备。
- **波特率**：选择I2C波特率。
- **地址**：添加从设备引导加载程序的十六进制地址。
- **速度模式**：选择发送速度模式：标准或快速。
- **上升时间**：根据速度模式选择值：0-1000（标准）、0-300（快速）。
- **下降时间**：根据速度模式选择值：0-300（标准）、0-300（快速）。

2.2 存储器和文件编辑

存储器和文件编辑面板用于读取和显示目标存储器内容和文件内容。

2.2.1 读取和显示目标存储器

图18. 存储器和文件编辑：设备存储器选项卡

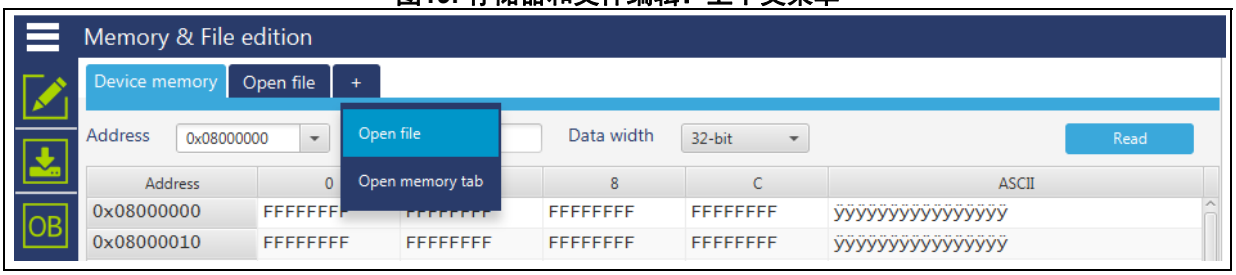


目标连接后，可以使用此面板读取STM32目标存储器。要实现这一点，需指定要读取的数据地址和大小，然后单击左上角的读取按钮。可使用“数据宽度”组合框以不同格式（8、16和32位）显示数据。

还可使用选项卡上下文菜单或操作按钮中的“另存为...”菜单，将设备存储器内容存储在.bin、.hex或.srec文件中。

您可以打开多个设备存储器选项卡来显示目标存储器的不同位置。要做到这一点，只需单击“+”选项卡来显示一个上下文菜单，该菜单允许您添加新的“器件存储器”选项卡，或打开文件并将其显示在“文件”选项卡中：

图19. 存储器和文件编辑：上下文菜单



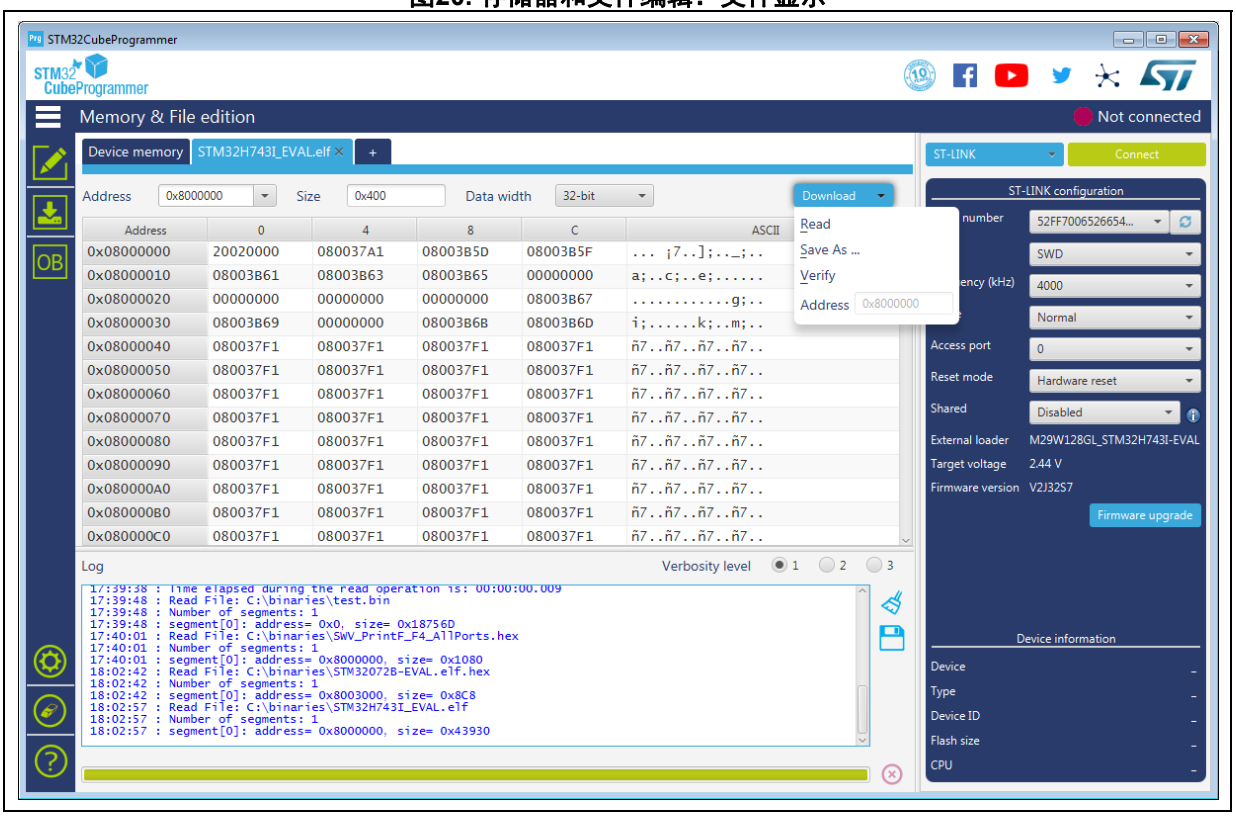
2.2.2 读取并显示文件

要打开并显示文件，只需点击“+”并选择“打开文件”菜单，如*图 19*所示。

支持的文件格式为二进制文件（.bin）、ELF文件（.elf，.axf，.out）、Intel十六进制文件（.hex）和Motorola S-record文件（.Srec）。

文件打开并解析后，会在专用选项卡中显示其文件名，如*图 20*所示。文件大小显示在“大小”字段中，hex、srec或ELF文件的起始地址显示在“地址”字段中，二进制文件的起始地址为0。

图20. 存储器和文件编辑：文件显示



可以修改地址字段，以便从某个偏移量开始显示文件内容。使用选项卡上下文菜单或操作按钮时，可以使用“下载”按钮/菜单下载文件。对于二进制文件，您需要在“地址”菜单中指定下载地址。用户可使用“验证”菜单确认是否已下载文件，并将文件保存为另一种格式（.bin、.hex或.srec）。

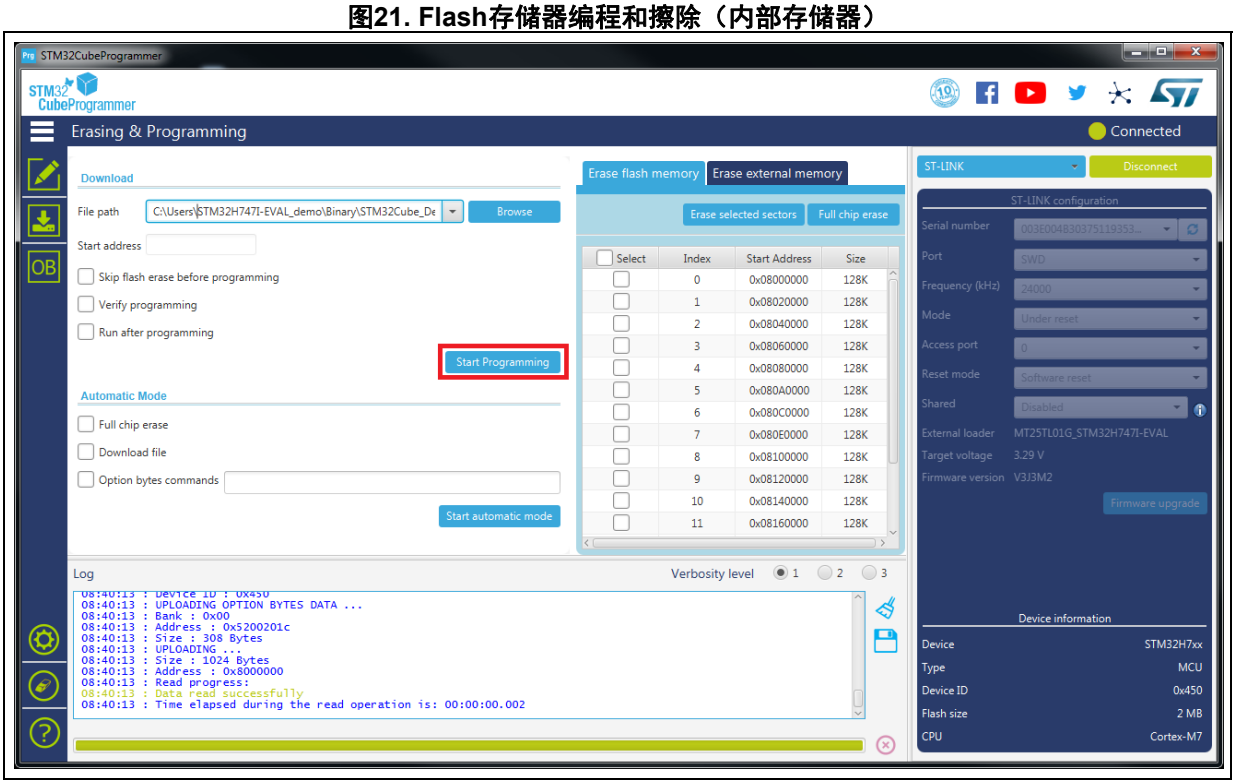
对于“器件存储器”选项卡，用户可以使用“数据宽度”组合框以不同格式（8、16和32位）显示文件存储器内容。

2.3 存储器编程和擦除

该面板专用于Flash存储器存编程和擦除操作。

2.3.1 内部Flash存储器编程

图21. Flash存储器编程和擦除（内部存储器）



存储器擦除

连接到目标后，存储器扇区将显示在右侧面板中，显示每个扇区的起始地址和大小。要擦除一个或多个扇区，请在第一列中进行选择，然后单击“Erase selected 擦除所选扇区”按钮。

“全片擦除”按钮用于擦除整个Flash存储器。

存储器编程

为了进行存储器编程，执行以下步骤：

1. 点击浏览按钮并选择要编程的文件。支持的文件格式为二进制文件（.bin）、ELF文件（.elf，.axf，.out）、Intel十六进制文件（.hex）和Motorola S-record文件（.Srec）。
2. 在编程二进制文件的情况下，须设置编程的地址。
3. 选择编程选项：
   - 编程后验证：读回所编程的内存并逐个字节地与文件进行比较。
   - 编程前跳过Flash擦除：若选中，在编程前不擦除存储器。只有当您确定目标内存已被擦除时才须选中该选项。
   - 编程后运行：编程后立即启动应用程序。
4. 点击“开始编程”按钮开始编程。

窗口底部的进度条显示擦除和编程操作的进度。

2.3.2 外部Flash存储器编程

为了对通过任意可用接口（如SPI、FMC、FSMC、QSPI和OCTOSPI）连接到微控制器的外部存储器进行编程，您需要一个外部加载程序。

对于大多数带有外部存储器的STM32评估和探索板，STM32CubeProgrammer一并提供了对应的外部加载程序，可在“bin / ExternalLoader”目录下找到。如需创建新的外部加载程序，参见*第 2.3.3 节*获取更多信息。

为了进行外部存储器编程，从“外部加载程序”面板选择一个或多个外部加载程序，工具将使用加载程序读取、编程或擦除外部存储器，如*图 22*所示。选定之后，使用外部加载程序在其存储器范围内进行任何存储器操作。

“擦除和编程”面板右侧的“外部Flash擦除”选项卡显示每个选定加载程序的存储器扇区，可以进行扇区或整个芯片的擦除，如*图 23*所示。

图22. Flash存储器编程（外部存储器）

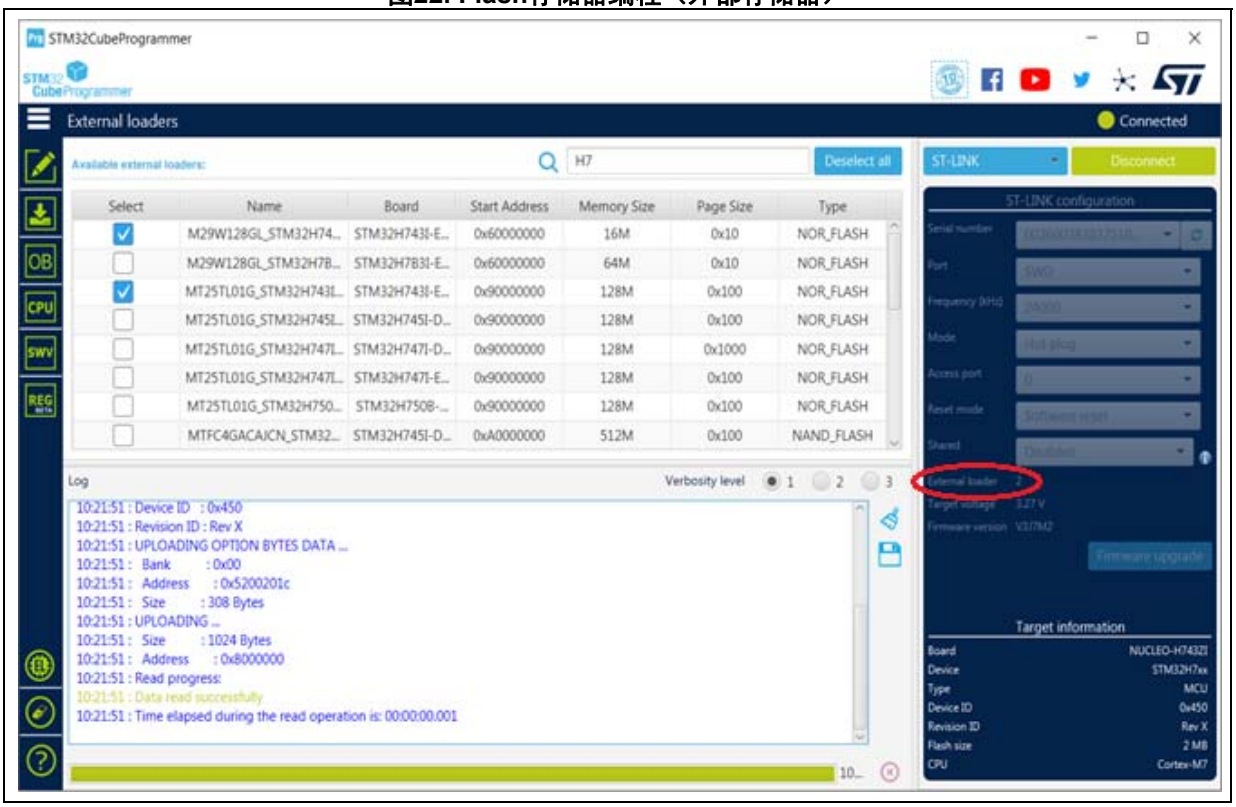
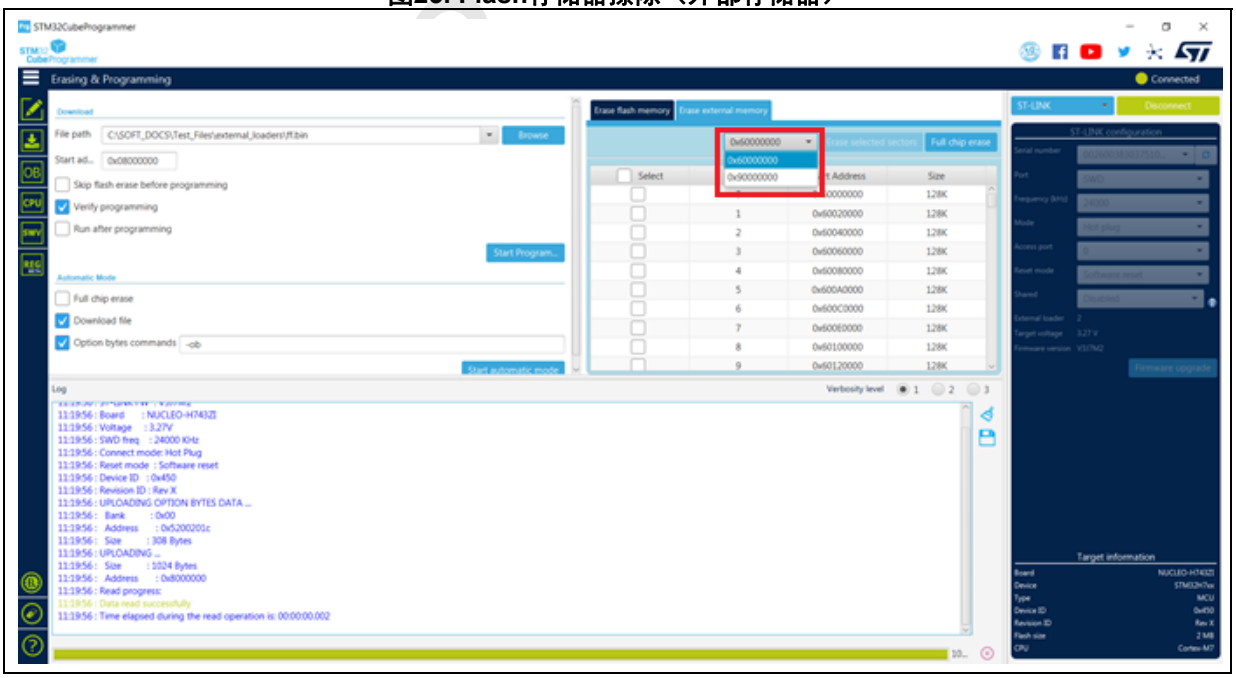


图23. Flash存储器擦除（外部存储器）



2.3.3 为外部存储器开发自定义加载程序

基于*"bin/ExternalLoader"*目录下的示例，用户可以为给定的外部存储器开发自定义加载程序。为三种工具链提供了这些示例：Keil® MDK、EWARM和TrueSTUDIO®。自定义加载程序的开发可以使用上述工具链之一来执行，能够保持相同的编译器/链接器配置，如示例中所示。

外部Flash编程机制与STM32 ST-LINK实用工具所用的机制相同。为用于ST-LINK实用程序所开发的任何Flash加载程序都能与STM32CubeProgrammer工具兼容，并且可以不加任何修改地使用。

要创建一个新的外部存储器加载程序，请按照以下步骤操作：

1. 使用外部存储器相关的正确信息，来更新*Dev_Inf.c* 文件的*StorageInfo* 结构中的设备信息。
2. 在*Loader_Src.c* 文件中重写相应的函数代码。
3. 更改输出文件名。

注： *一些函数是强制性的，不能省略（参见Loader_Src.c文件中的函数说明）。*

不得修改链接文件（linker files）或分散链接描述文件（scatter files）。

在构建外部加载程序项目之后，会生成一个ELF文件。ELF文件的扩展名取决于所用工具链（对于Keil为.axf，对于EWARM为.out，以及对于TrueSTUDIO或任何基于gcc的工具链为.elf）。

必须将ELF文件的扩展名更改为".stldr"，且必须将该文件复制到"*bin/ExternalLoader*"目录下。

Loader_Src.c文件

基于特定IP为内存开发外部加载程序需要以下函数：

- **Init**函数

  Init函数定义将外部存储器连接到设备所用的GPIO引脚，并初始化所用IP的时钟。

  如果成功则返回1，失败则返回0。

  int Init (void)

- **Write**函数

  Write函数将一块RAM范围中的缓冲区数据写入到指定的地址上去。

  如果成功则返回1，失败则返回0。

  int Write (uint32_t Address, uint32_t Size, uint8_t* buffer)

- **SectorErase**函数

SectorErase函数擦除指定扇区的存储器。

如果成功则返回1，失败则返回0。

int SectorErase (uint32_t StartAddress, uint32_t EndAddress)

其中"**StartAddress**"等于要擦除的第一个扇区的地址，"**EndAddress**"等于要擦除的扇区末尾地址。

注： *该函数在外部SRAM加载程序中不能使用。*

在外部加载程序中定义上述函数是必要的。工具用其来擦除和编程外部存储器。例如，如果用户从外部加载程序菜单中单击程序按钮，该工具将执行以下操作：

- 自动调用**Init**函数来初始化接口（QSPI、FMC……）和Flash存储器
- 调用**SectorErase()**来擦除所需的Flash存储器扇区
- 调用**Write()**函数来编程存储器

除了这些函数，您还可以定义以下函数：

- **Read**函数

  Read函数用来读取指定范围的存储器，并将读取的数据返回到RAM里的缓冲区中。

  如果成功则返回1，失败则返回0。

  int Read (uint32_t Address, uint32_t Size, uint16_t* buffer)

  其中“**Address**” = 读取操作起始地址，“**Size**”为读取操作的大小，“**buffer**”为指向读取后的数据的指针。

注： *对于QSPI/OSPI（Quad-SPI/Octo-SPI）存储器，可以在Init函数中定义存储器映射模式；这种情况下，Read函数无用，因为数据可以直接从JTAG/SWD接口读取。*

- **Verify**函数

  选择“verify while programming”模式时会调用**Verify**函数。该函数检查编程的存储器是否与RAM中定义的缓冲区保持一致。它返回一个uint64，定义如下：

  返回值 = ((checksum<<32) + AddressFirstError)

  其中“**AddressFirstError**”为第一次失配的地址，“**checksum**”所编程缓冲区的校验和值

  uint64_t Verify (uint32_t FlashAddr, uint32_t RAMBufferAddr, uint32_t Size)

- **MassErase**函数

  MassErase函数擦除整个存储器。

  如果成功则返回1，失败则返回0。

  int MassErase (void)

- 校验和函数

所有上述函数在成功操作的情况下返回1，在失败的情况下返回0。

Dev_Inf.c文件

此文件中定义的StorageInfo结构提供有关外部存储器的信息。该结构定义的信息类型示例如下所示：

```
#if defined (__ICCARM__)
  __root struct StorageInfo const StorageInfo = {
#else
  struct StorageInfo const StorageInfo = {
#endif
  "External_Loader_Name", // Device Name + version number
  MCU_FLASH, // 器件类型
  0x08000000, // 器件起始地址
```

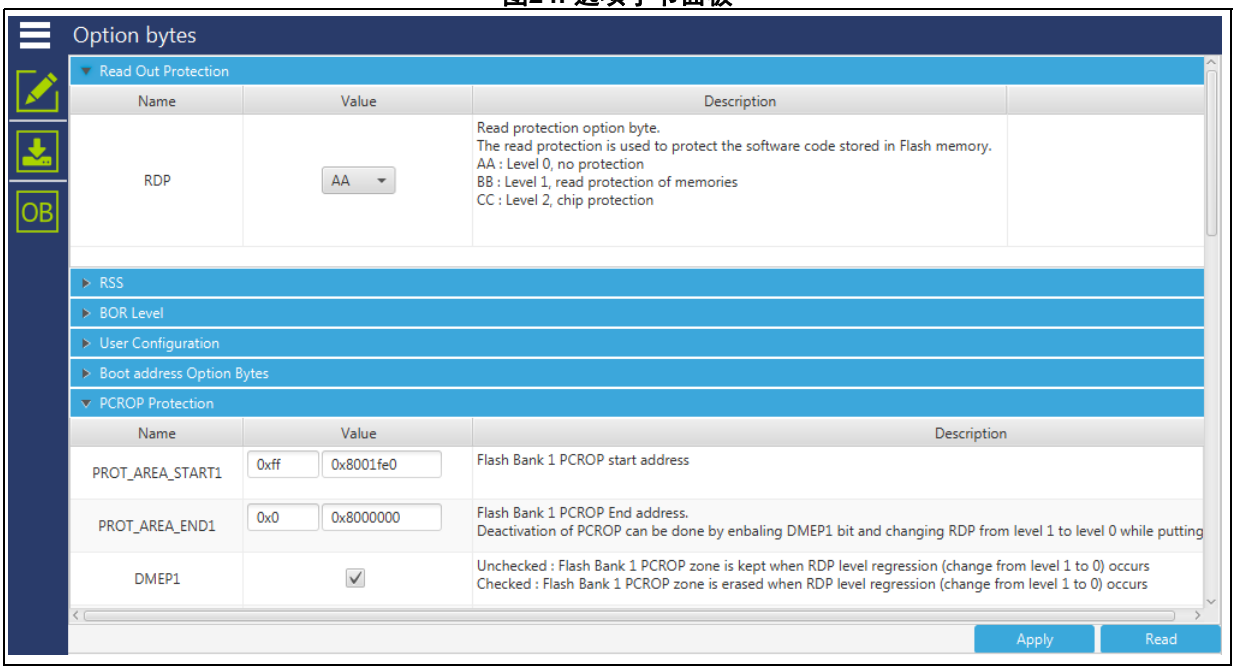
```
        0x00100000, // 器件以字节计的大小 (1MBytes/8Mbits)
        0x00004000, // 页编程大小 16KBytes
        0xFF, // 被擦除的存储器初始值
// 指定扇区的大小和地址（查看下面的示例）
        0x00000004, 0x00004000, // Sector Num : 4 ,Sector Size: 16KBytes
        0x00000001, 0x00010000, // Sector Num : 1 ,Sector Size: 64KBytes
        0x00000007, 0x00020000, // Sector Num : 7 ,Sector Size: 128KBytes
        0x00000000, 0x00000000,
};
```

2.4 选项字节

选项字节面板允许用户按类别分组读取和显示目标选项字节。选项位在表格中显示，有三列内容，其中包含位名、值以及对设备的影响的描述。

用户可通过更新值字段来修改这些选项字节的值，然后点击应用按钮，以此执行编程并确认修改过的选项字节的正确编程。用户可以随时点击读取按钮，读取并刷新显示的选项字节。

图24. 选项字节面板

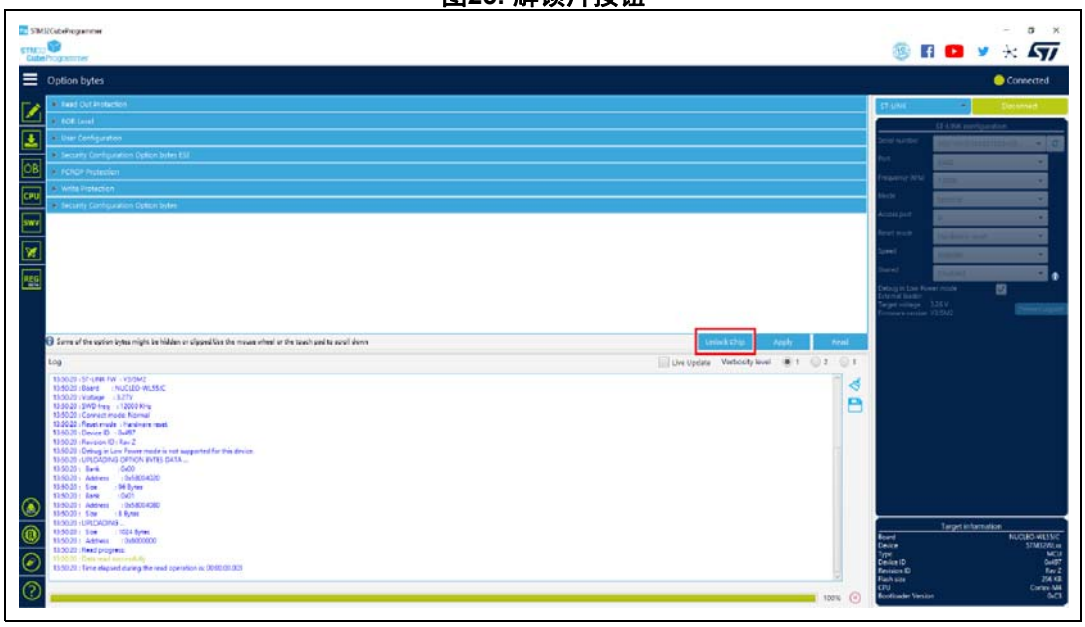


请参考ST网站*www.st.com*上提供的Flash存储器编程手册和参考手册中的选项字节部分了解更多信息。

2.4.1 MCU解锁（特定于STM32WL系列）

如果已经设定了不良选项字节，则用户可通过单击“解锁片”按钮解锁器件（只可用于STLink连接）。解锁后需执行掉电重启。

图25. 解锁片按钮

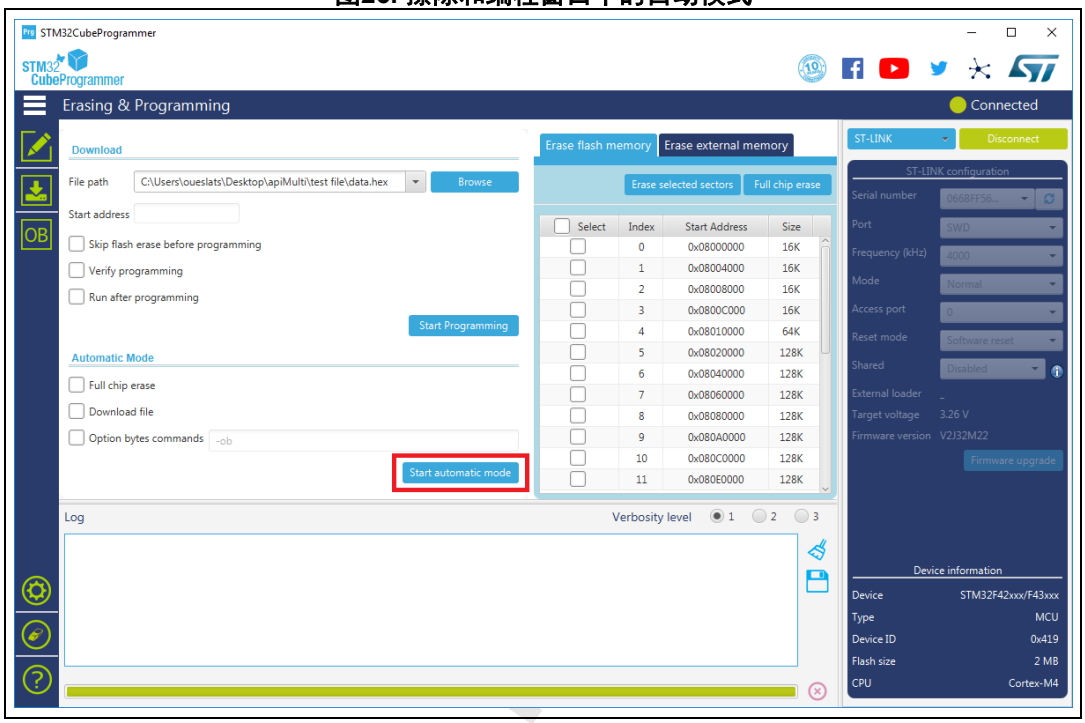


2.5 自动模式

擦除和编程窗口中显示的自动模式（参见*图 26*）用于环路中STM32器件的编程和配置。支持的操作：

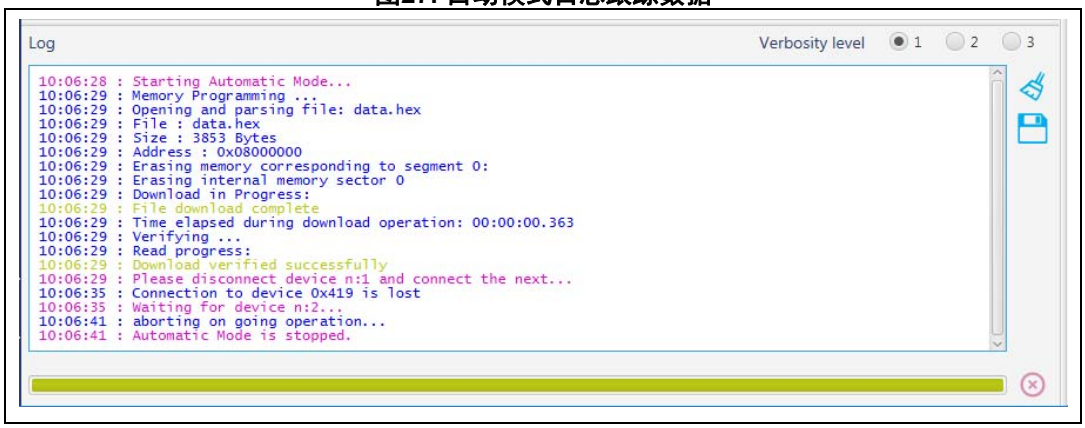
- 全片擦除：擦除整个Flash存储器
- 下载文件：激活和设置“下载”部分的编程选项：
  - 文件路径
  - 起始地址
  - 编程前跳过擦除
  - 编程验证
  - 编程后运行
- 选项字节指令：通过设置选项字节命令行配置器件

图26. 擦除和编程窗口中的自动模式



日志面板中的所有自动模式跟踪数据（参见*图 27*）显示了过程演变和用户干预消息。

图27. 自动模式日志跟踪数据



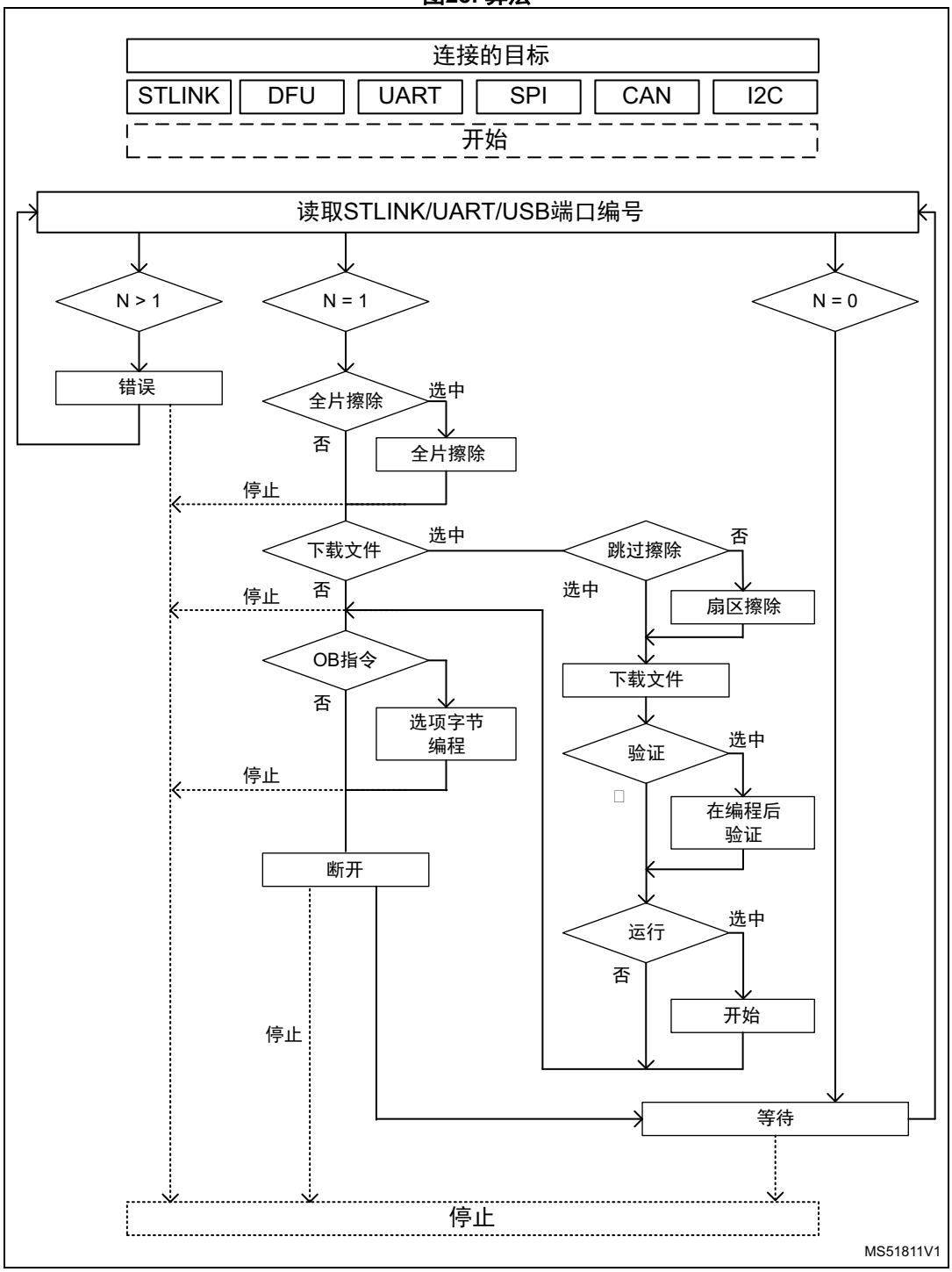
图形指南

- 在执行自动模式收集与后续所有器件相关的连接参数值之前，必须与第一个目标建立连接。
- 如果选中了“下载文件”，则系统将执行所有“下载文件”选项，否则执行选中的“下载”选项。
- 如果选中了“选项字节指令”，则文本字段会被激活，然后用户可以插入选项字节指令（如CLI指令），请确保开头无空格：
  -ob [OptionByte=value] [OptionByte=value] [OptionByte=value] …
- 选项字节指令的示例：“**–ob BOR_LEV=0 nBOOT0=1**”
- 如果按下“启动自动模式”按钮，则系统将进入循环，直至系统停止被调用。
- 当自动模式处于执行状态时，所有图形对象均被禁用。
- 用户可在任何需要的时间通过按下取消按钮或停止自动模式按钮停止此过程。

日志消息

- “启动自动模式…”
  表示系统成功进入自动处理。
- “检测到一个以上的ST-LINK工具！只保留一个ST-LINK工具！“
  如果使用JTAG/SWD接口时计算机连接了一个以上的ST-LINK工具，则不能使用自动模式。系统会显示一条消息，阻止并要求用户只保留一个连接的ST-LINK探头以便继续使用该模式。
- “检测到一个以上的ST-LINK Bridge！只保留一个ST-LINK Bridge！”
  如果使用引导加载程序接口SPI/CAN/I^2C时计算机连接了一个以上的ST-LINK Bridge，则不能使用自动模式。系统会显示一条消息，阻止用户并要求其只保留一个连接的ST-LINK Bridge以便继续使用该模式。
- “检测到一个以上的ST-LINK USB DFU！只保留一个USB DFU！”
  如果使用USB引导加载程序接口时计算机连接了一个以上的USB DFU，则不能使用自动模式。系统会显示一个对话框，阻止并要求用户只保留一个连接的USB DFU以便继续使用该模式。
- “检测到的UART端口多于上一次连接！”
  在第一次连接时，自动模式计算可用串行端口的数量并将其作为参考，以便正确地检测我们只对STM32器件使用了一个端口UART。
- “请断开器件并连接下一个…”
  如果系统完成了第一个过程，则无论结果如何，断开当前器件并准备连接第二个器件。
- “等待器件…”
  在与上一个器件的连接正常断开后，系统持续搜索新器件。
- “自动模式停止。”
  表示需要取消，系统停止此过程。

图28. 算法


连接的目标
STLINK
DFU
UART
SPI
CAN
I2C
开始
读取STLINK/UART/USB端口编号
N > 1
N = 1
N = 0
错误
全片擦除
选中
否
全片擦除
停止
下载文件
选中
跳过擦除
否
停止
否
选中
扇区擦除
OB指令
下载文件
否
选项字节
编程
验证
选中
停止
在编程后
验证
断开
运行
选中
否
开始
停止
等待
停止
MS51811V1

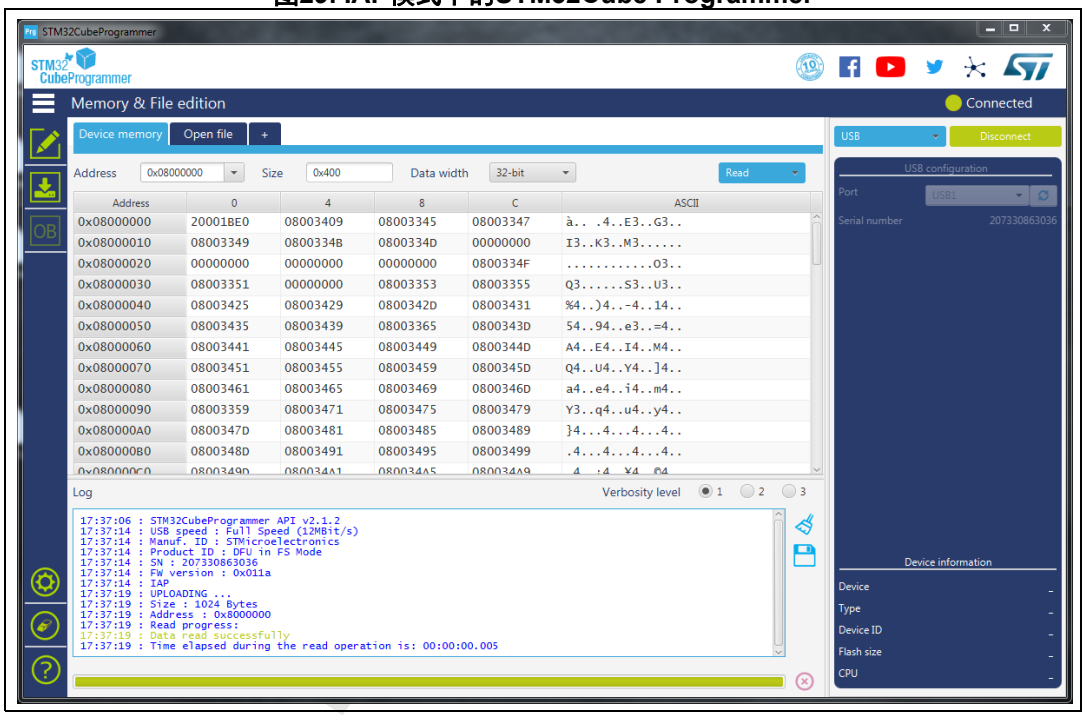
2.6 应用内编程（IAP/USBx）

只有在USB连接模式下，STM32CubeProgrammer才支持IAP/USBx。当选择USB连接且从Flash存储器启动时，STM32CubeProgrammer检测到IAP/USBx（如DFU引导加载程序），连接后，日志面板上显示IAP/ USBx消息。

注： *使用IAP/USBx时选项字节和扇区擦除不可用。*

ST网站（*www.st.com*）上的CubeFW/ CubeAzure中提供了IAP/USBx示例。

图29. IAP模式下的STM32Cube Programmer



2.7 使用图形界面刷写协处理器二进制文件

2.7.1 FUS/栈升级

1. 使用STM32CubeProgrammer（2.4或更高版本），参见*图 30*
2. 访问SWD/引导加载程序USB接口，参见*图 31*
3. 删除当前无线栈，参见*图 32*
4. 无更新版FUS时，用下载栈的方式升级FUS版本
5. 下载新的FUS
6. 下载新的无线栈（必须显示弹出窗口，才能确保成功升级），参见*图 33*

注： *在STM32CubeProgrammer（2.7或更高版本）中，用户可以只安装新固件（v1.11.0或更高版本的栈）。使用STM32CubeProgrammer v2.6.0安装旧版固件。*

图30. STM32CubeProgrammer API SWD连接

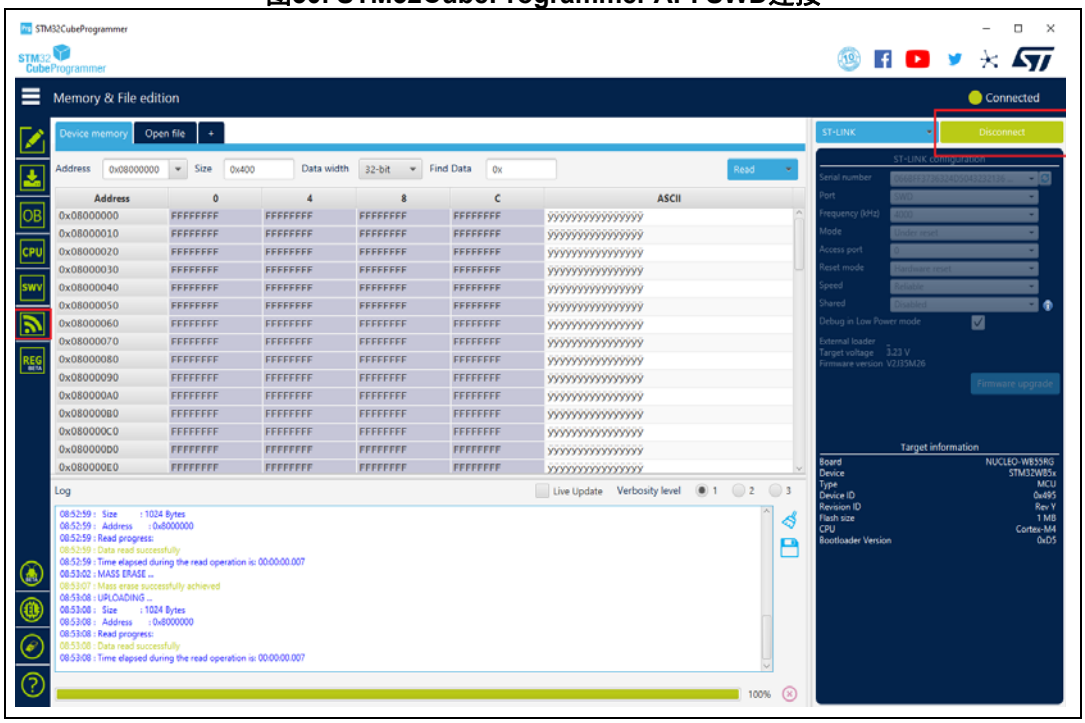


图31. 固件升级步骤

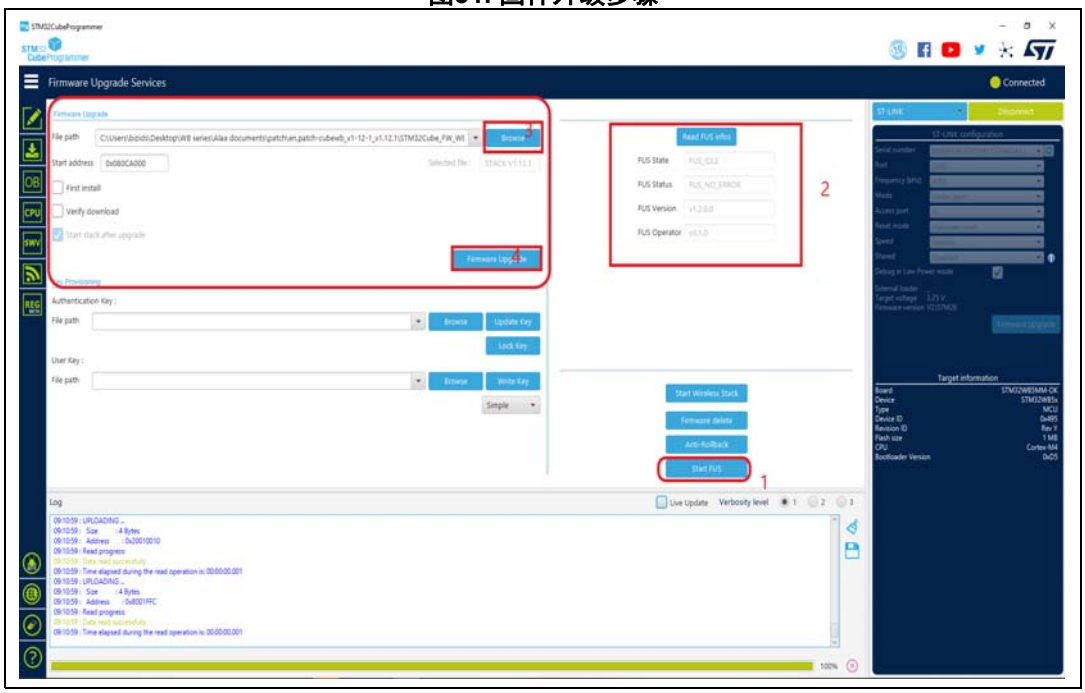


图32. 确认固件成功删除的弹出窗口

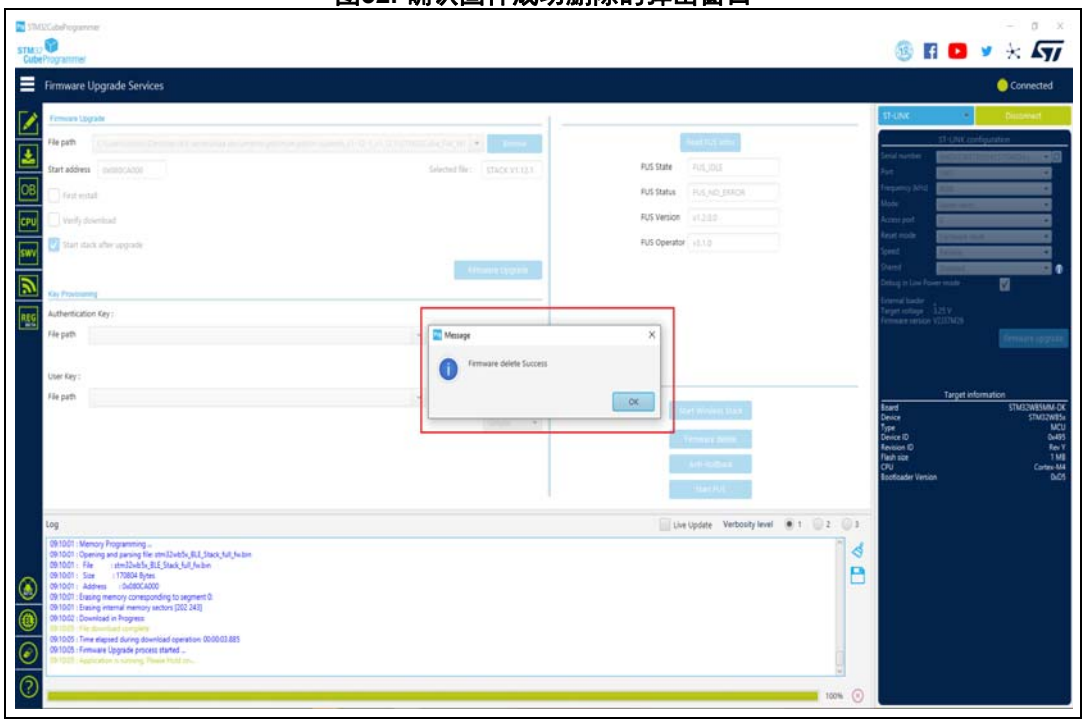
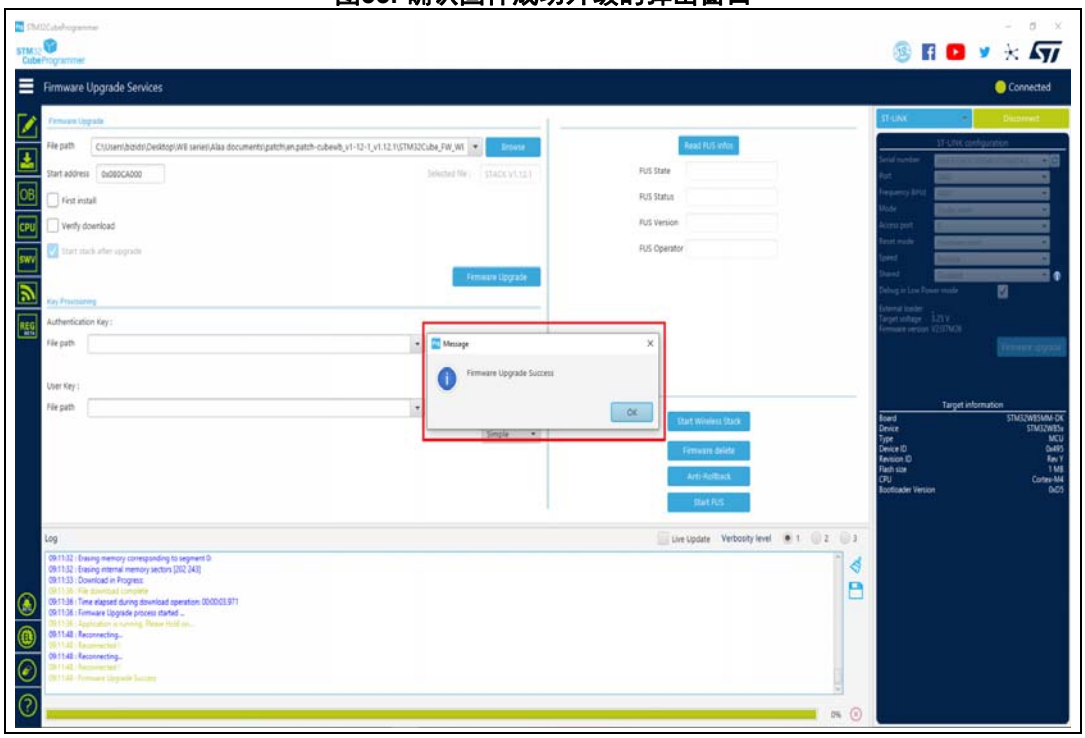


图33. 确认固件成功升级的弹出窗口



2.7.2 密钥配置

在STM32CubeProgrammer中，用户可以为任何加载文件添加自定义签名（由意法半导体加密并签名）。

用户验证

在FUS窗口中，可通过更新密钥按钮存储用户验证密钥（*图 34*）。

在安装用户验证密钥后，可以进行修改，除非选择了锁定用户验证密钥按钮（参见*图 35*）。在安装验证密钥后，必须通过双重签名FUS/栈执行安装或升级服务，否则会被拒绝。

图34. 升级验证密钥

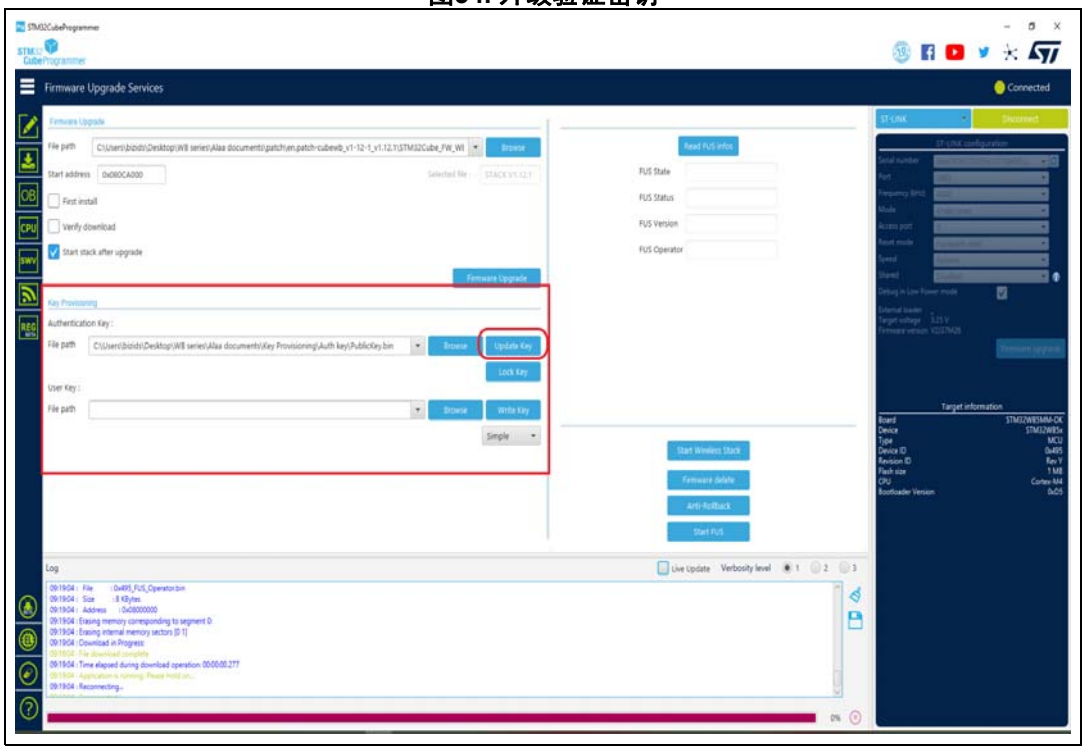
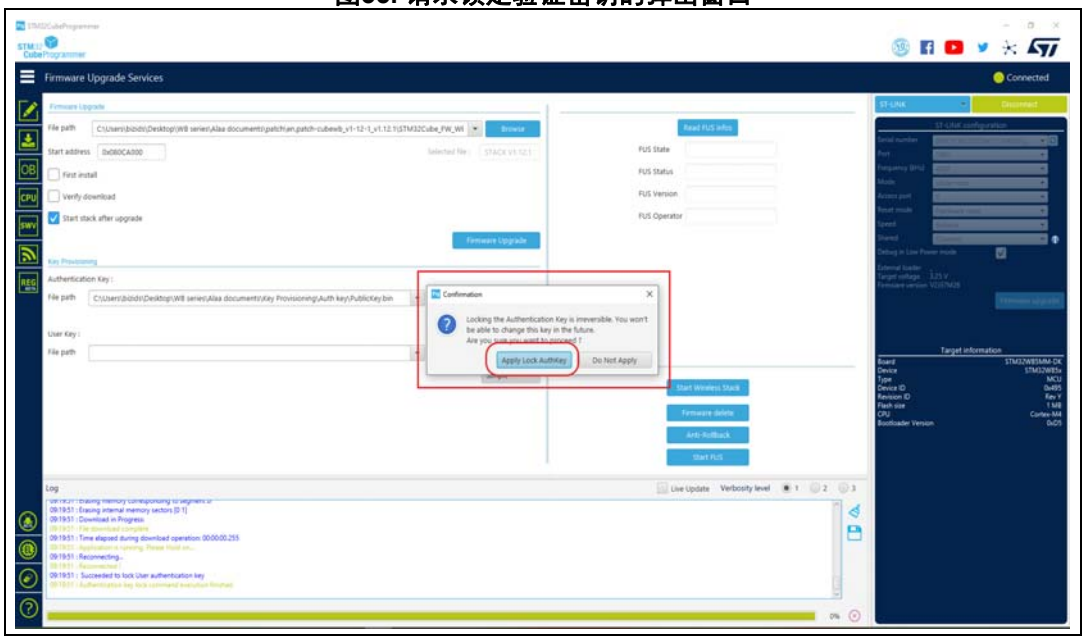


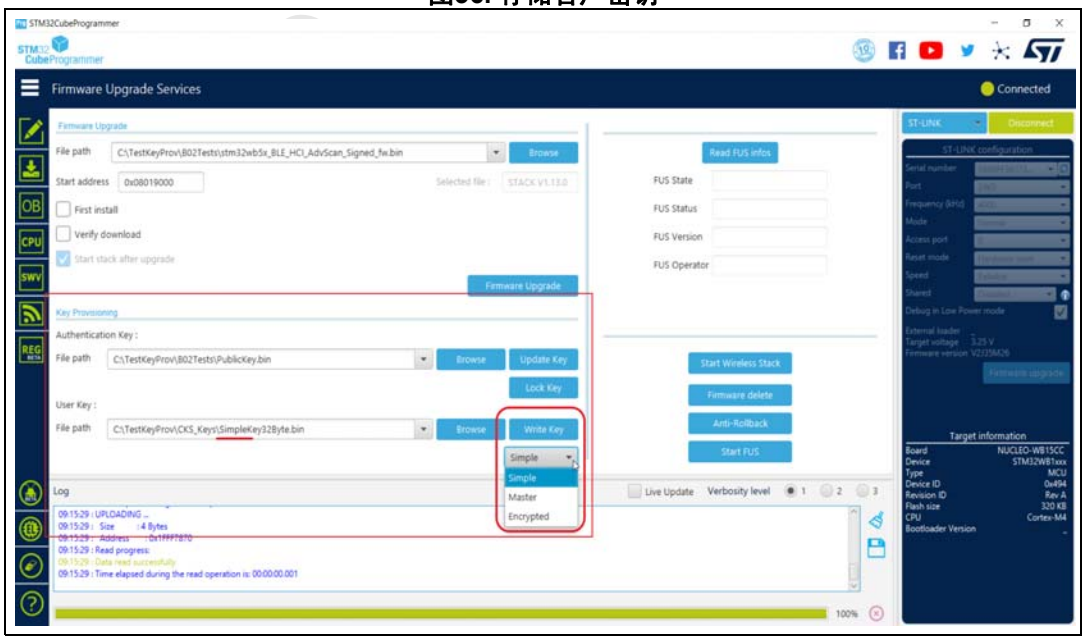
图35. 请求锁定验证密钥的弹出窗口



客户密钥存储

在STM32CubeProgrammer中，可将客户密钥以二进制格式存储在专用FUSFlash存储器区中（用户密钥类型：简单、主要或加密），参见*图 36*。

图36. 存储客户密钥



关于客户密钥存储的更多信息，请参考AN5185 “*STM32WB系列的ST固件升级服务*”。关于STM32WBxx产品的完整文档，请访问ST网站*www.st.com*上的专用网页。

2.8 串行线查看器（SWV）

串行线查看器窗口（参见*图 37*）显示目标通过SWO发送的printf数据。它显示关于正在运行的固件的有用信息。

注： *只有使用SWD接口时串行线查看器才可用。*

在开始接收SWO数据之前，为了让工具正确地配置ST-LINK，用户必须指定准确的目标系统时钟频率（以MHz为单位），以及正确的SWO频率的目标。“激励端口”组合框允许用户选择给定ITM激励端口（从端口0至31）或从所有ITM激励端口同步接收数据。

用户可选择使用“浏览”按钮指定“.log”文件，以便保存SWV跟踪日志，默认位置为“$USER_HOME/STMicroelectronics/STM32CubeProgrammer/SWV_Log/swv.log”。

用户可选择选中“激活色彩”复选框，以便以彩色显示跟踪信息输出。此特性要求原始跟踪信息包含下列颜色编码：

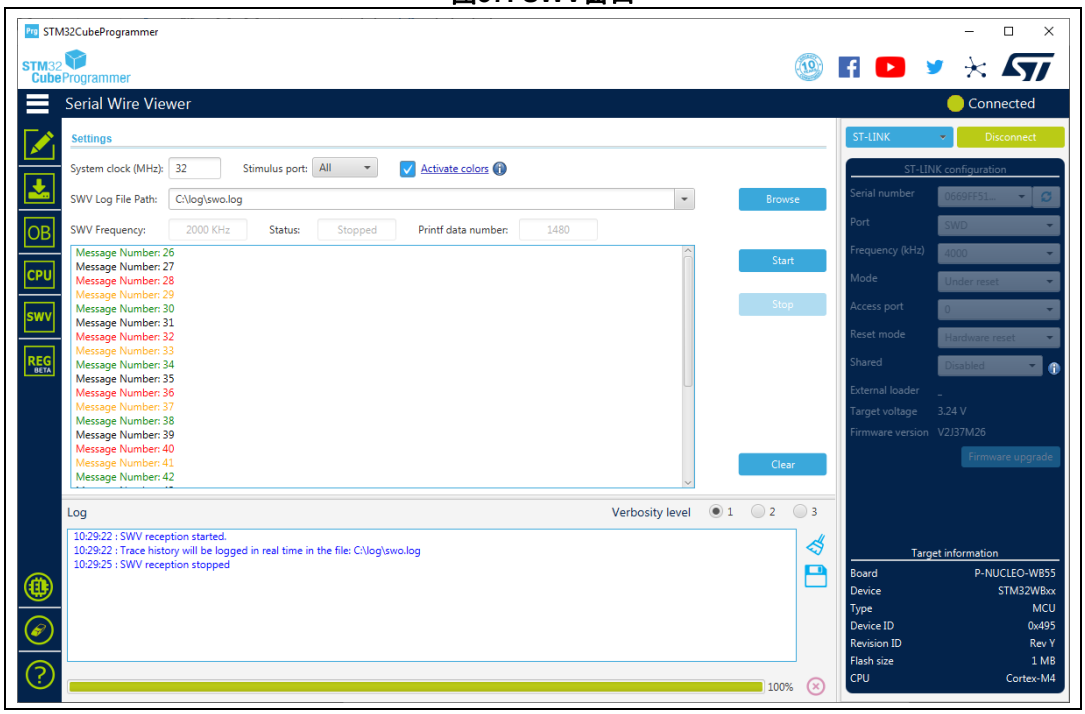
- #GRN#代表绿色
- #RED#代表红色
- #ORG#代表橙色

示例：

printf(“#GRN#This outputs a green message!”);

通过单击“激活色彩”复选框旁边的“信息图标”按钮，可访问演示特性和显示使用方法的帮助窗口。

图37. SWV窗口



在指定SWV配置后，可使用“开始”和“停止”按钮开始或停止SWV接收。SWO数据显示在专用区域，可使用“清除”按钮将其清除。

SWV信息栏显示关于当前SWV传输的有用信息，如SWO频率（从系统时钟频率推导得出）和接收的printf数据量（以字节为单位）。

注： *由于ST-LINK硬件缓冲区的容量有限，传输期间可能会丢失一些SWV字节。*

2.9 MCU的STM32CubeProgrammer Script Manager平台

2.9.1 Script Manager的使用场景介绍

在ScriptManager平台上，可以自动化STM32CubeProgrammerCLI指令，并添加宏以便操作从STM32 MCU读取的数据。

2.9.2 Script Manager的使用

创建扩展名为prg的文件，然后开始写入命令行界面（CLI），所有STM32MCU和特定脚本宏均支持CLI。在完成脚本填充后，在CLI模式下用*-script*指令连接STM32板件并开始执行。

用法示例：*STM32_Programmer_CLI -script myScript.prg*

Script Manager可以使用数学和逻辑运算，参见*表 1*。

表1. Script Manager支持的运算

	逻辑
– +（加） – -（减） – *（乘） – /（除）	– &&（逻辑与） – ||（逻辑或） – &（按位与） – |（按位或） – ^ (XOR) – << >>（左移和右移）

使用命令行界面（CLI）：在该脚本中，我们可以使用STM32MCU支持的所有CLI（参见*第 3 节*）。

使用特定的Script Manager宏分析、显示和修改数据，每个宏均以#开头。支持以下宏。

#Write宏：

#Write32(Address,data)
#Write16(Address,data)
#Write8(Address,data)
#WriteX(Address,#var) （其中的X为8/16/32）

说明： 将指定的32/16/8位数据从指定地址开始下载到Flash存储器中。

#Read宏：

#Read(Address)
#variable=#Read(Address)

说明：从指定地址读取32位数据存储器或从指定宏地址读取32位数据存储器，并将其放入用户使用的变量。

#Display宏：

#Display(“message”)
#Display(#errorLevel)
#Display(#variable)

说明：显示脚本中已使用的任何消息、数据、错误级别和变量内容。

#Delay宏：

#Delay(Time)

说明：使用户能够将系统置于待机状态一段时间（单位：ms）。

计算宏：

#variable=[var1] op [var2]
#variable=var1 shift（移动的位数）

说明：在Script Manager中用数学和逻辑运算进行计算。

断开连接指令
--scriptdisconnect

说明：使用户能够断开器件并重新连接同一脚本中的另一个端口。

注： *您可以在Script Manager中使用“//”添加命令，如示例中所示。*

Script Manager示例1（CLI和脚本宏），参见*图 38*

```
-c port=swd
-e 0 1
#Write32(0x08000000,0xAAAABBBB)
#var0=#Read(0x08000000)
#Display(#var0)
```

Script Manager示例2，参见*图 39*

```
-c port=swd
#Write32(0x08000000,0xAAAABBBB)
--scriptdisconnect
#Delay(5000)
-c port=COM17
#Write16(0x08000004,0xCCCC)
```

Script Manager示例3

```
-c port=swd
#Display ("Hello World!")
-e 0 1
#Write32(0x08000000,0xAAAABBBB)
#Read(0x08000000)
-r32 0x08000000 0x50
#var0=#Read(0x08000000)
#Display(#errorLevel)
#Display(#var0)
#Write32(0x08000004,#var0)
#Delay(3000)
#Write16(0x08000008,0xCCCC)
#Read(0x08000004)
#Display(#errorLevel)
#var1=#Read(0x08000008)
#Display(#var1)
#Write8(0x08000010,0xDD)
#Delay(5000)
#var2=#Read(0x08000010)
#Display(#var2)
#var3=(((0xbb*1)+(1-1))/1)
#Display(#var3)
#Write8(0x08000014,#var3)
#var4=((0xbb & 0xaa)| 0xbb )
#Display(#var4)
#var5=((0xbb && 0xaa) || 0xbb )
#Display(#var5)
#var6=(0xbb >>1)
#Display(#var6)
-e 0 1
-w32 0x08000000 0xAAAAAAAA
-r32 0x08000000 0x50
```

图38. Script Manager示例1的输出

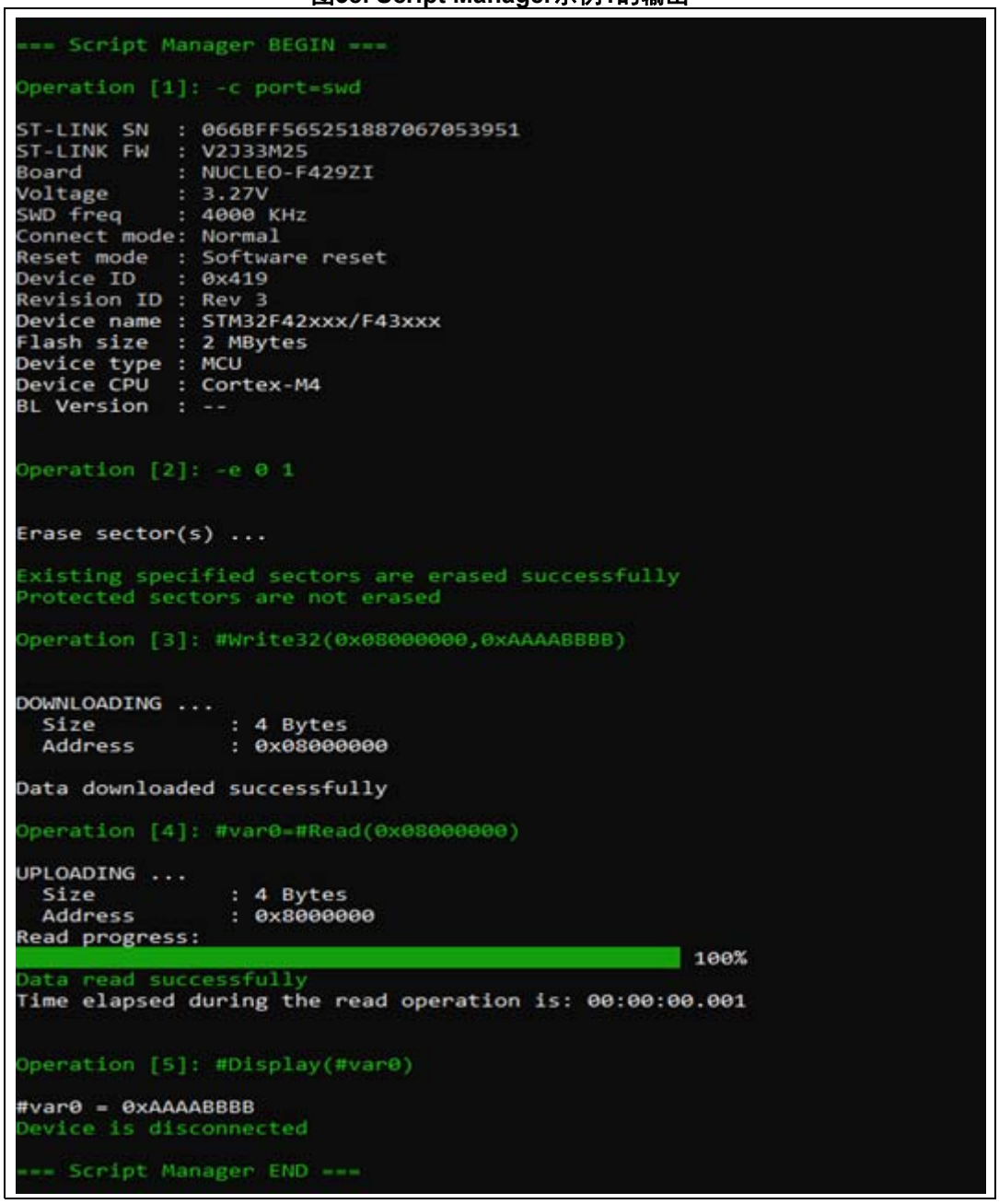
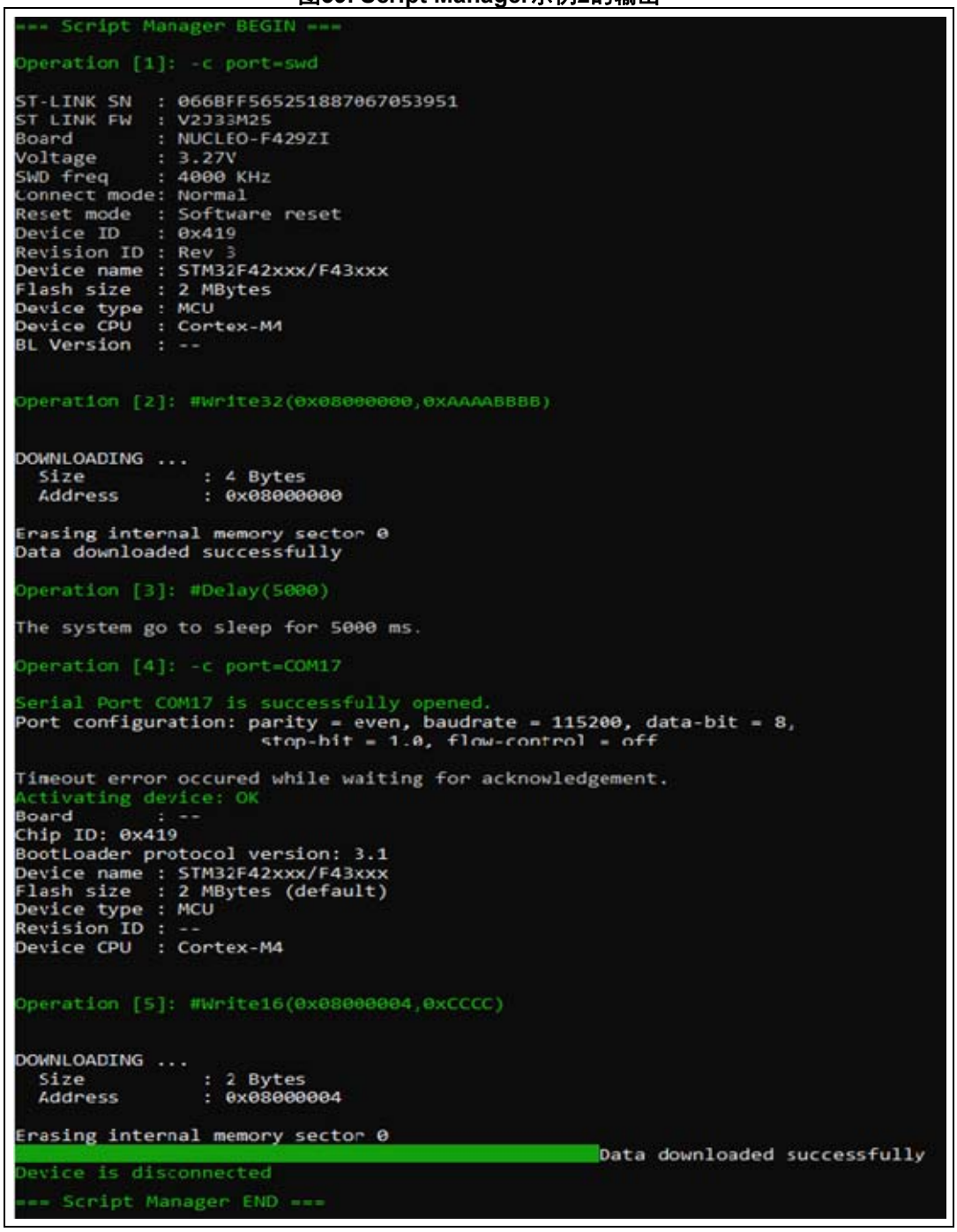


图39. Script Manager示例2的输出



2.10 具有自定义PID和VID的DFU IAP/USBx

在通过DFUIAP进行连接时，STM32CubeProgrammerDFUIAP/USBx并非只支持ST产品ID。

在使用新的产品ID开始DFU连接之前，为USB驱动程序签名（请访问http://woshub.com获取更多信息）。

当选择USB连接（具有新的产品ID）且从Flash存储器启动时，STM32CubeProgrammer检测到IAP/ USBx（如DFU引导加载程序），连接后，日志面板上显示IAP消息。

如需通过新的USB DFU进行连接，按照以下流程操作：

1. 修改默认产品ID
2. 修改默认供应商ID
3. 点击刷新按钮，然后点击连接按钮

注： *如果用户不输入PID或VID值，则STM32CubeProgrammer将采用ST产品的默认PID和VID（PID=0XDF11，VID=0X0483）。*

*图 40*显示了通过新的USB DFU面板进行连接的步骤，*图 41*显示了连接后的STM32CubeProgrammer主窗口。

图40. 通过USB DFU面板连接

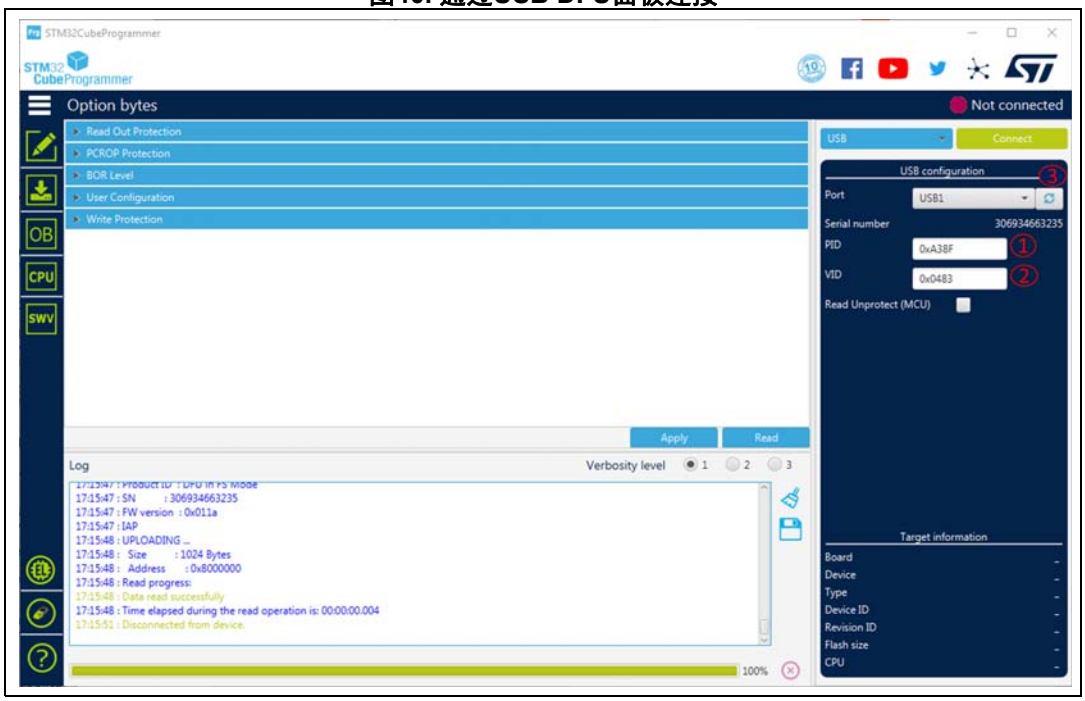
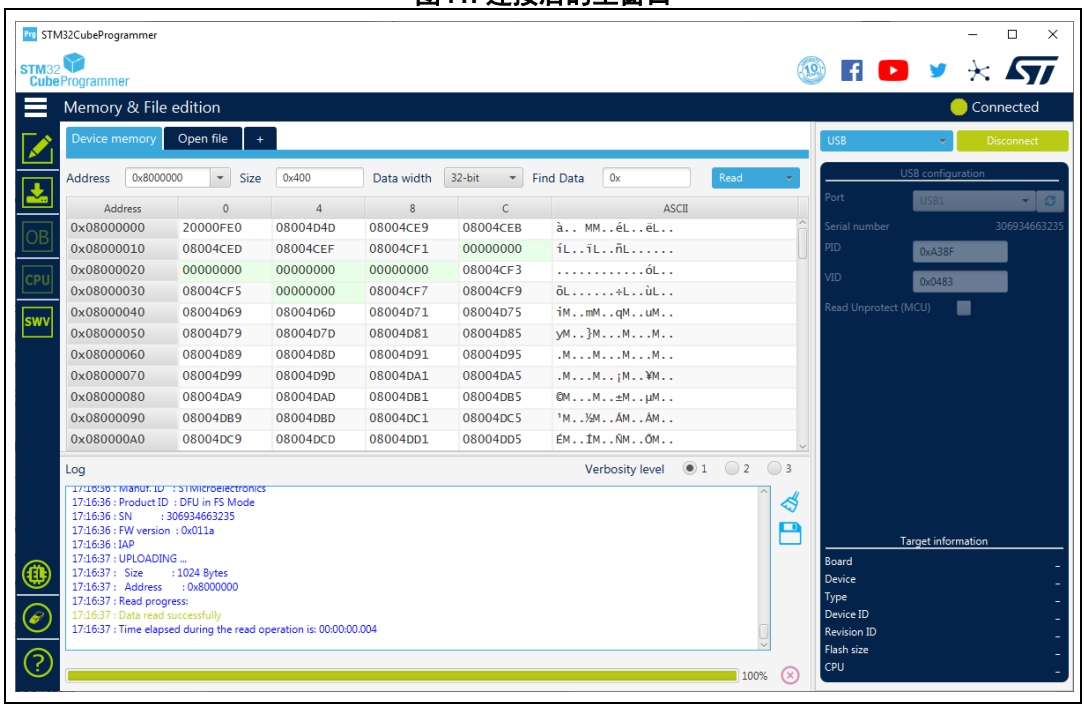


图41. 连接后的主窗口



注： *对于CLI模式，请参考第 3.2.1 节：连接指令。*

2.11 SigFox™凭证

在连接STM32WL器件后，将显示*图 42*中所示窗口。

此窗口显示芯片证书，其大小为136字节。用户可将其保存在二进制文件中，并将数据复制到剪贴板。

在提取芯片证书后，后端网络服务验证数据并返回两个SigFox凭证：二进制文件和头文件。

情况1：二进制原始文件

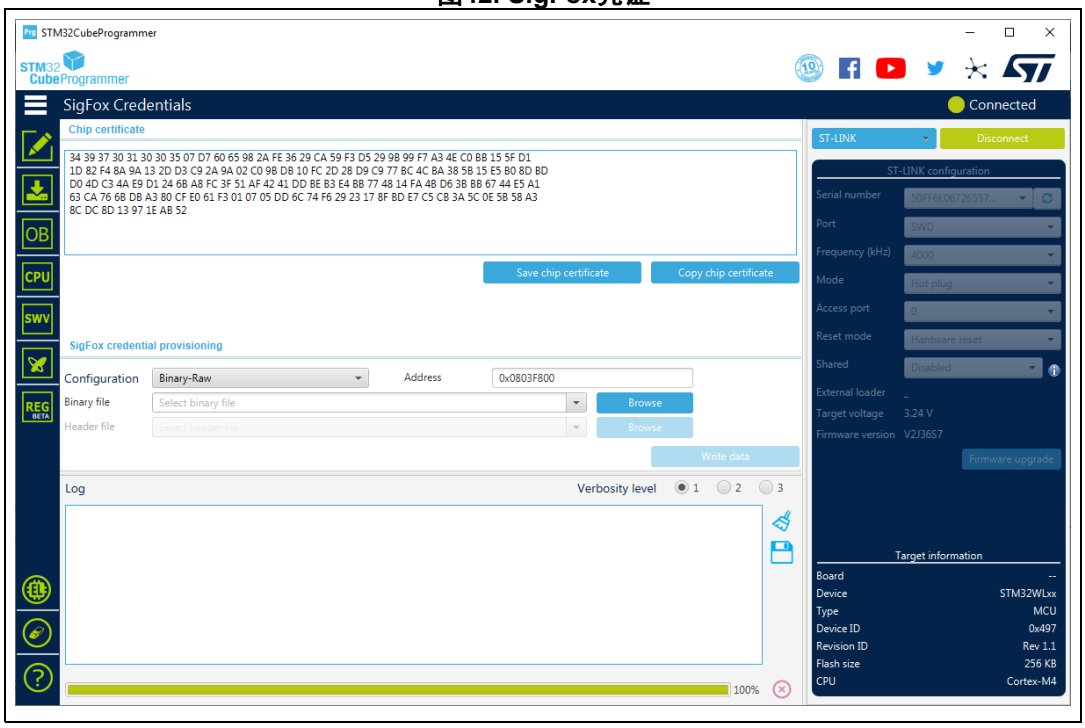
使用后端网络服务返回的二进制文件。此文件的大小必须等于48字节，在默认地址0x0803E500写入。

情况2：二进制KMS

使用后端网络服务返回的头文件。在默认地址0x0803E500写入。

注： *为了使用STM32CubeProgrammer访问ST SigFox服务器，用户必须点击“打开Sigfox页面”。一个网页随即打开，用户必须手动复制证书，然后生成SigFox凭证（二进制文件和头文件）。*

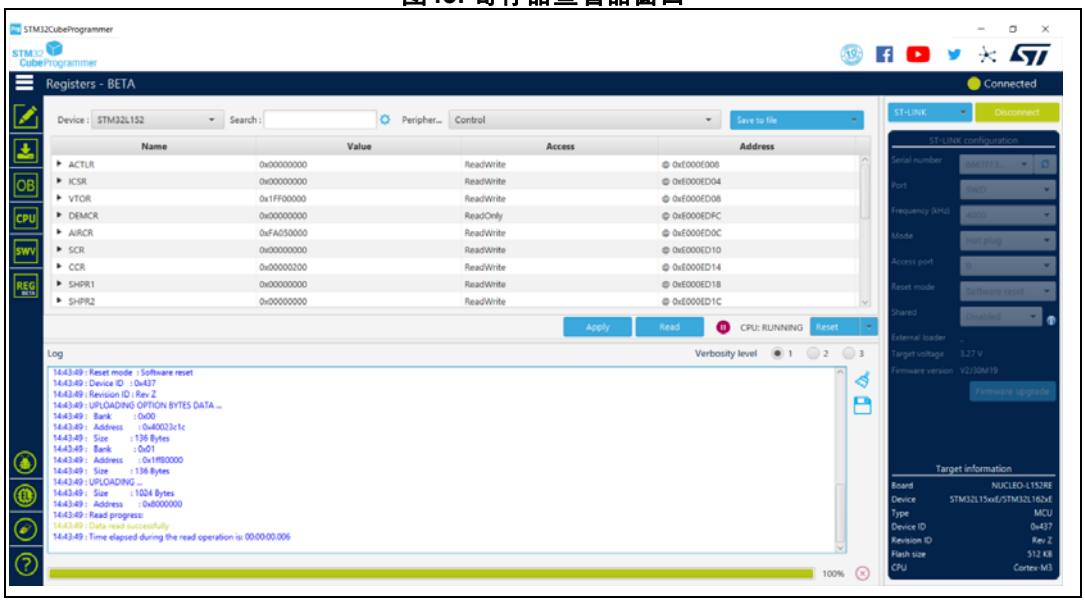
图42. SigFox凭证



2.12 寄存器查看器

STM32CubeProgrammer支持寄存器查看器功能（参见*图 43*），可在运行应用的同时实现所有MCU和核心寄存器的实时可视化。它还支持修改MCU寄存器值或将其保存到日志文件中。

图43. 寄存器查看器窗口



注： *只有使用SWD/JTAG接口时寄存器查看器才可用。*

寄存器查看器具有一个作为输入的文件列表，这些文件包含描述内核和STM32寄存器映射的数据（“svd”文件）。

2.13 硬性故障分析器

2.13.1 说明

STM32CubeProgrammer故障分析器解读从基于Cortex-M的器件提取的信息，以便找出故障原因。

此信息在GUI模式或CLI模式下的故障分析器窗口中实现可视化。它帮助识别应用软件驱动CPU进入故障状态时发生的系统故障。

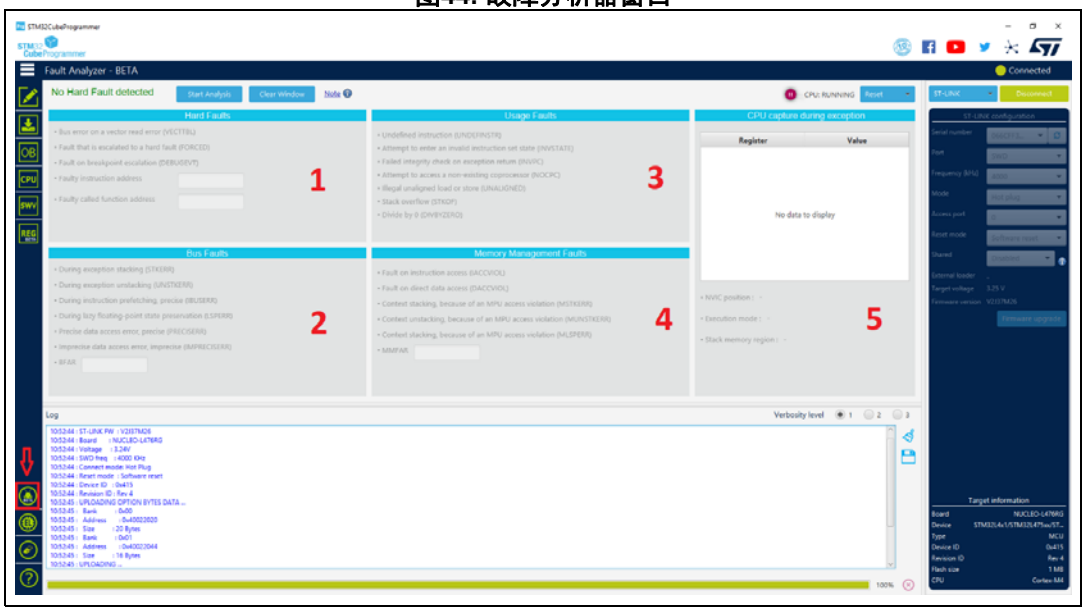
可能检测到的故障异常：

- 硬性故障：默认异常，可由异常处理期间由总线故障、存储器管理故障或使用故障（如果其处理程序无法执行）导致的错误触发。
- 存储器管理故障：检测对存储器管理单元（MPU）定义的区域的存储器访问违规，如从只允许读/写访问的存储器区域执行代码。
- 总线故障：在发生中断（进入/退出）时，检测指令提取、数据读取/写入、中断向量提取和寄存器入栈（保存/恢复）时的存储器访问错误。
- 使用故障：为加载/存储多个指令检测未定义指令和未对齐存储器访问的执行。启用后，将检测除零错误和其他未对齐存储器访问。
- 安全故障：提供基于Cortex-M33的器件的安全相关故障的信息。

注： *故障分析器只对ST-LINK接口可用。*

如*图 44*所示，故障分析器窗口由五个主要部分组成。

图44. 故障分析器窗口



1. 硬性故障详情：显示发生的故障类型，定位指令和调用的函数地址。
2. 总线故障详情：显示指令获取和数据访问导致的总线错误状态，表示在总线工作期间检测到存储器访问故障。地址应显示在BFAR文本字段。
3. 使用故障详情：包含一些指令执行故障和数据访问的状态。
4. 存储器管理故障详情：表示MPU检测到存储器访问违规。如果此故障由错误地址触发，则将在MMFAR文本字段中显示访问。
5. 异常期间的CPU捕获：显示生成异常时的CPU状态，以便获得CPU寄存器的概况和一些帮助信息。
   a) NVIC位置：表示引发错误的中断的编号，如果是“-”，则中断/异常向量无特定位置。
   b) 执行模式：表示工作模式“Handler”/“Thread”。
   c) 栈存储区：表示故障期间使用的栈存储区“Main”或“Process stack”。

2.13.2 示例

开发一个生成使用故障的简单应用程序，在程序主函数中设置一个执行除零（不允许的操作）的指令。

int a = 4, b = 0, c = 0;

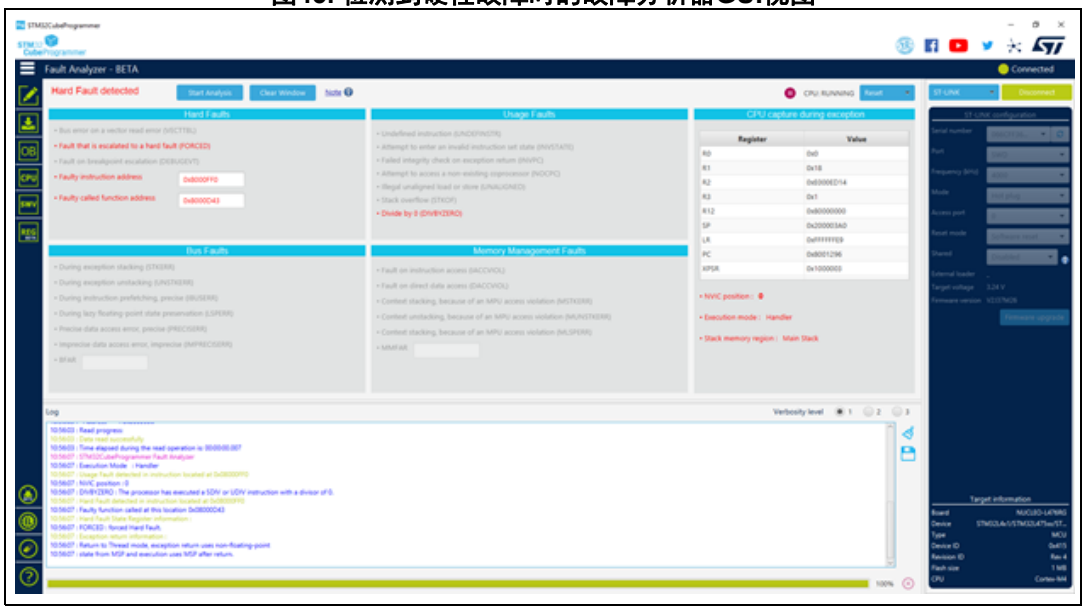
c = a / b;

打开故障分析器窗口，按下“开始分析”按钮启动故障检测算法，窗口上显示错误原因。

在本例中，窗口上显示“检测到硬性故障”，标签“除零（DIVBYZERO）”高亮显示，并显示附加信息：

- 故障指令地址：0x8000FF0
- 故障调用的函数地址：0x8000D40，表示调用故障指令的地址。
- NVIC位置：0，窗口看门狗中断
- 执行模式：Handler
- 栈存储区：主栈

图45. 检测到硬性故障时的故障分析器GUI视图



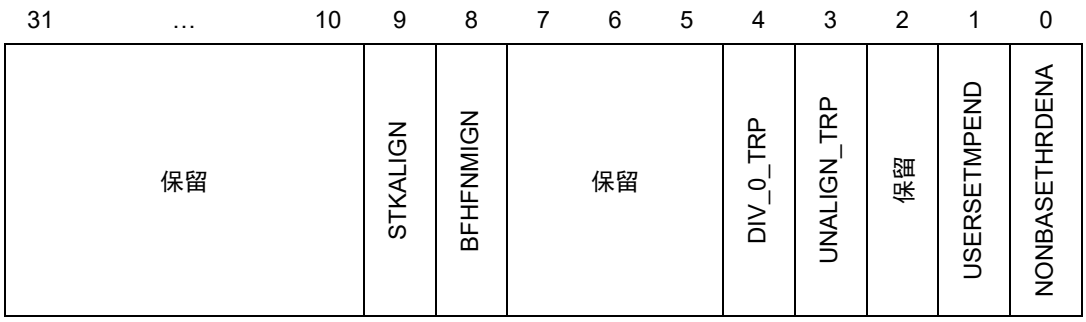
2.13.3 故障分析器注意事项

由于未被软件启用，故障分析器可能无法检测未追踪故障。

配置和控制寄存器（CCR）控制除零和未对齐存储器访问的使用故障的行为，主要用于控制可自定义的故障异常。

CCR的以下比特位控制使用故障的行为：

图46. CCR位



- DIV_0_TRP：在处理器执行除数为0的SDIV或UDIV指令时使能使用故障。
  - 0 = 不捕捉除零；除零返回商数0。
  - 1 = 捕捉除零。
- UNALIGN_TRP：在执行对未对齐地址的存储器访问时使能使用故障。
  - 0 = 不捕捉未对齐半字和字访问
  - 1 = 捕捉未对齐半字和字访问；未对齐访问生成使用故障。

注意，使用LDM、STM、LDRD和STRD指令的未对齐访问总是生成使用故障，即使是在UNALIGN_TRP置为0时。

STM32CubeProgrammer在分析开始时使能所需比特位，如果未检测到错误，则将显示描述性弹出窗口，指出您必须再现场景并重新开始故障分析。

2.13.4 Cortex-M33的安全故障分析器

STM32CubeProgrammer提供基于Cortex-M33的器件的安全相关故障的信息（CLI和GUI界面）。

在连接基于Cortex-M33的器件（如STM32L5系列的MCU）时，故障分析器窗口中添加了名为“安全故障”的新字段。

结果分析基于安全故障状态寄存器（SFSR）设置，如果发生错误，则将触发故障：

- INVEP：如果来自非安全状态或异常的函数调用以安全状态下的非SG指令为目标，则此位置位。如果目标地址是SG指令，但NSC标记置位时无匹配的SAU/IDAU区域，则此位也将置位。
- INVIS：如果出栈操作期间发现异常栈帧的完整性签名无效，则此位置位。
- INVER：当在非安全状态下从异常返回时，置为1。
- AUVIOL：为了设置为非安全的事务，做出了访问地址空间（用NS-Req标记为安全）中某些部分的尝试。如果在惰性状态保持期间发生了违规，则此位不置位。
- INVTRAN：表示未被标记为跨域的分支导致了从安全到非安全存储器的过渡，由此引发了异常。
- LSPERR：表示在浮点状态的惰性保持期间发生了SAU或IDAU违规。
- SFARVALID：当SFAR寄存器包含有效值时，此位置位。
- LSERR：表示在惰性状态激活或取消激活期间发生了错误。
- SFAR：表示发生安全故障时的地址值。

2.14 填充存储器指令

-fillmemory

说明：该指令用于从选定地址以给定模式填充存储器。

语法： -fillmemory <start_address> [size=<value>] [pattern=<value>] [datawidth=8|16|32]

<start_address>：	写访问的起始地址。 默认使用地址0x08000000。
[size=<value>]：	要写入的数据的大小。
[pattern=<value>]：	要写入的模式值。
[datawidth=8|16|32]：	填充数据大小，可以是8、16或32位。 默认值为8位。

- 示例1：STM32_Programmer_CLI.exe -c port=swd -fillmemory 0x08000000 size=0x10 pattern=0XAA datawidth=16（*图 47*）
- 示例2：STM32_Programmer_CLI.exe -c port=swd -fillmemory 0x08000000

图47. 示例 1

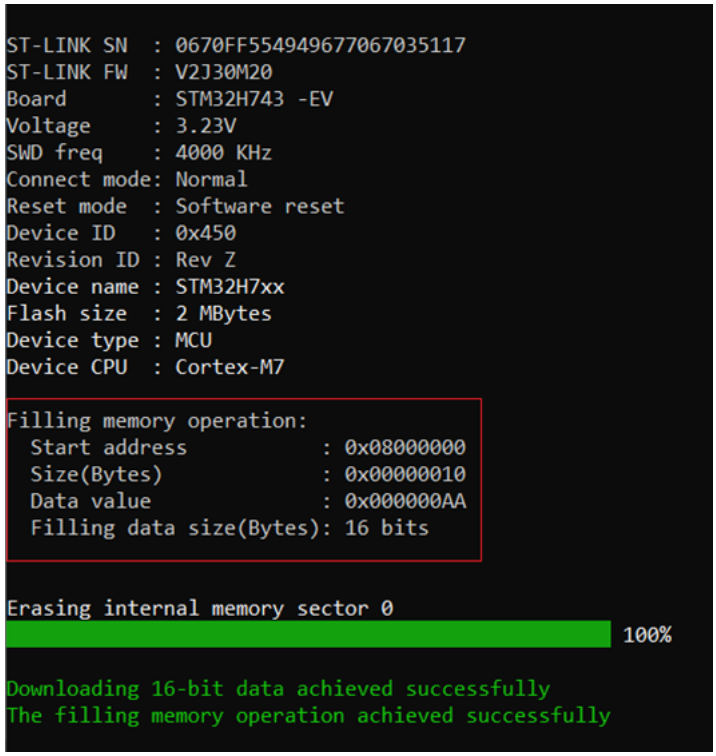
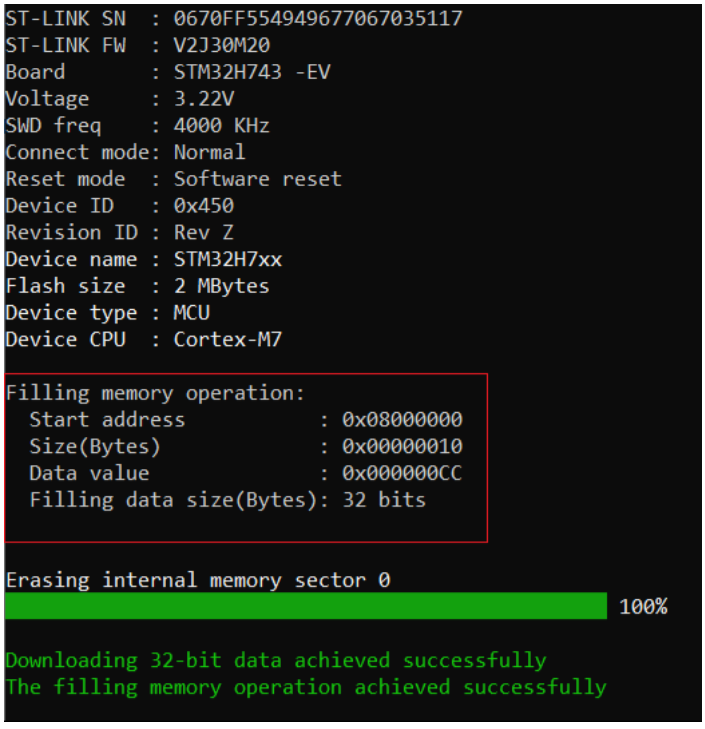


图48. 示例 2



2.15 填充存储器操作

用户可以从不同的子菜单打开填充存储器窗口。

图49. 从“读取”组合框显示子菜单

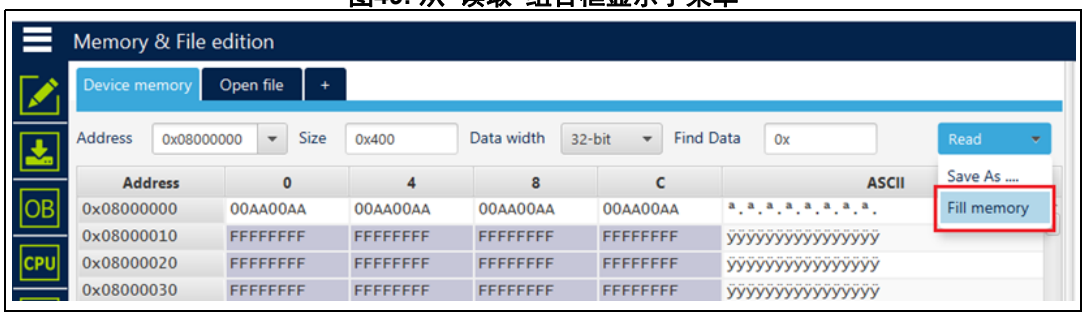


图50. 右键单击“器件存储器”选项卡显示子菜单

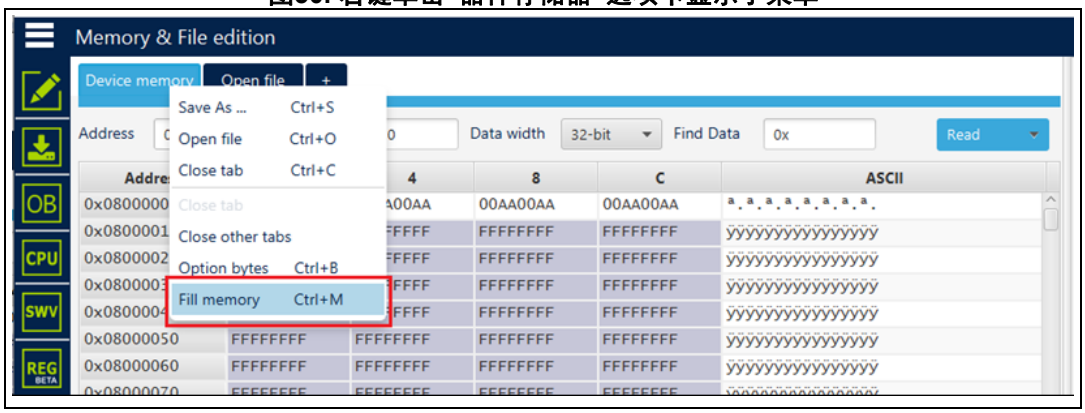
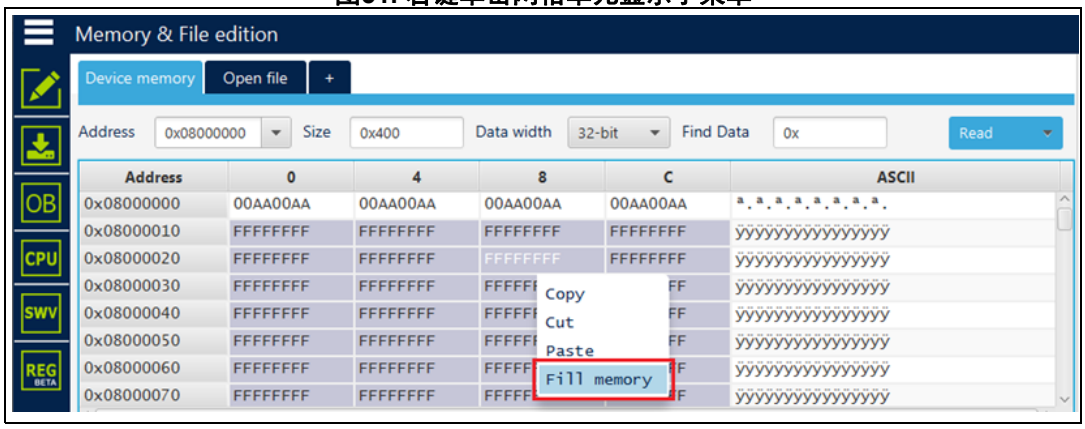


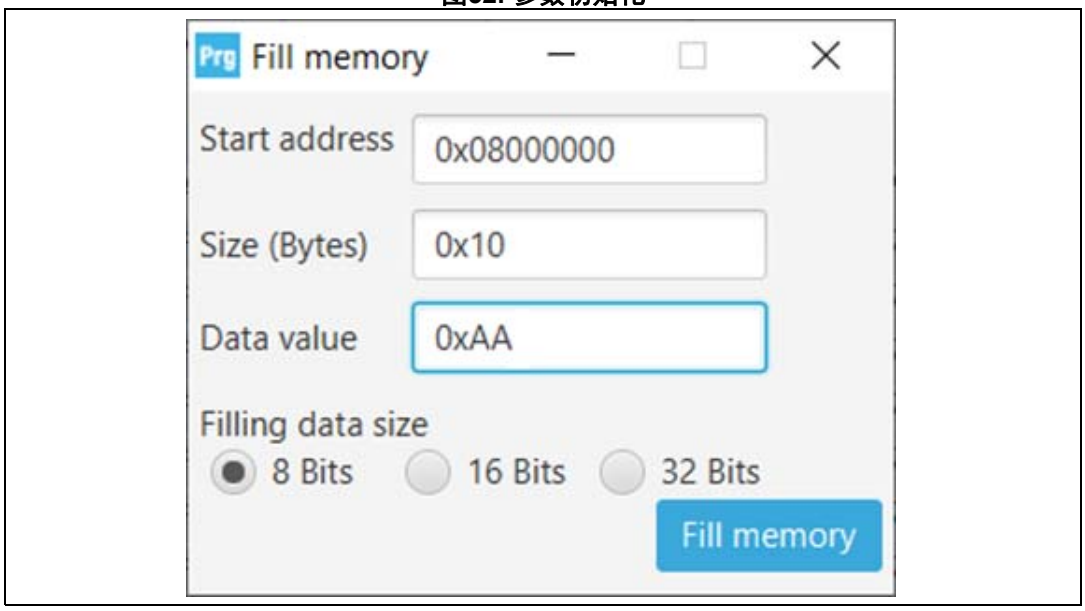
图51. 右键单击网格单元显示子菜单



注： *除了通过子菜单显示此窗口，用户还可以使用按键组合“Ctrl+M”直接打开它。*

在点击“填充存储器”选项后，将显示一个窗口，用户可以在这里初始化操作参数（参见 *图 52*）。

图52. 参数初始化



2.16 空白检查指令

-blankcheck

说明：该指令用于确认STM32Flash存储器为空白状态。如果不是，则将在消息中高亮显示第一个包含数据的地址。

语法：-blankcheck

示例： STM32_Programmer_CLI.exe -c port=swd –blankcheck

图53. 示例1：Flash存储器地址0x08000014处非空白

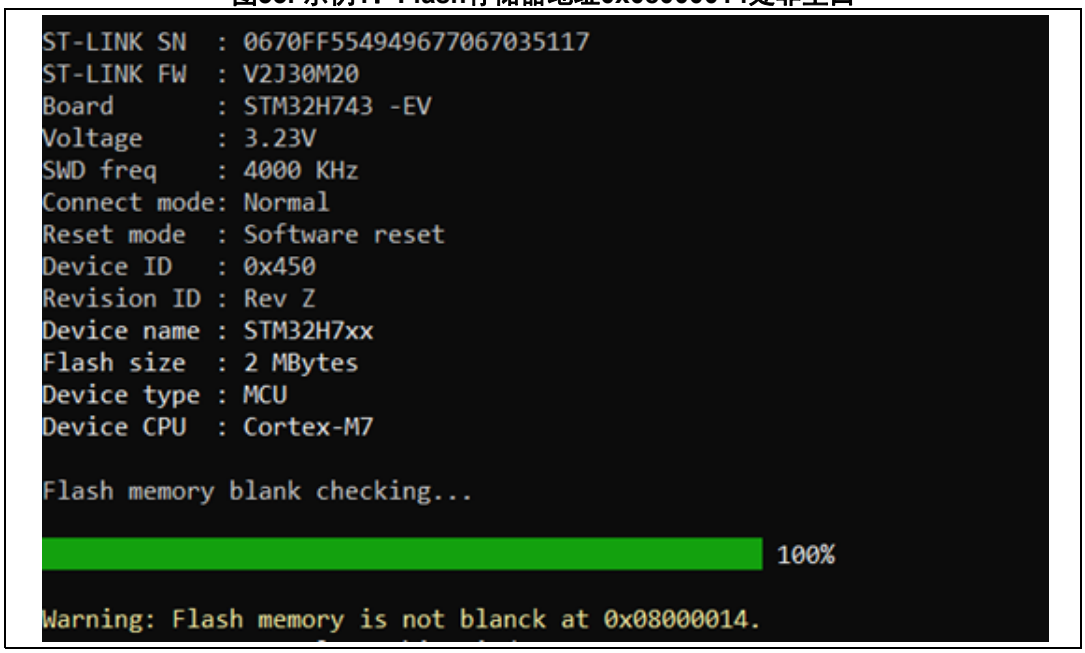
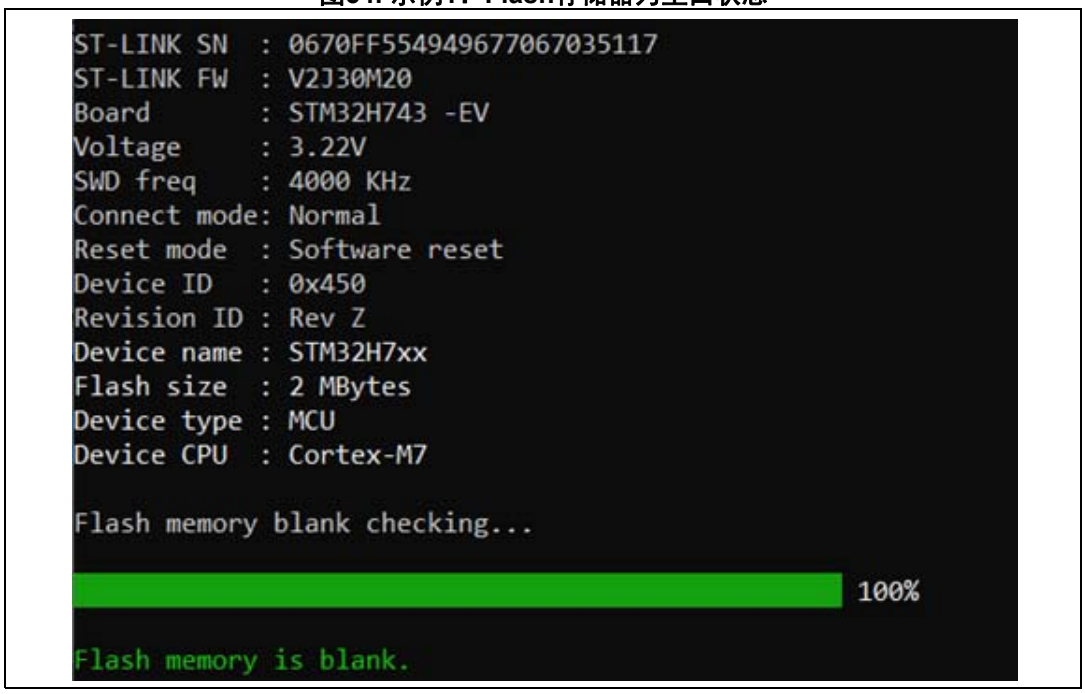


图54. 示例1：Flash存储器为空白状态



2.17 空白检查操作

用户可以从不同的子菜单打开填充存储器窗口。

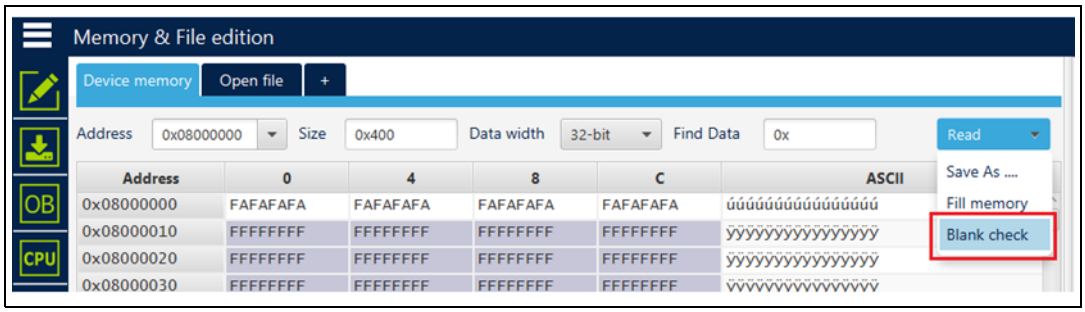


图56. 右键单击“器件存储器”选项卡显示子菜单

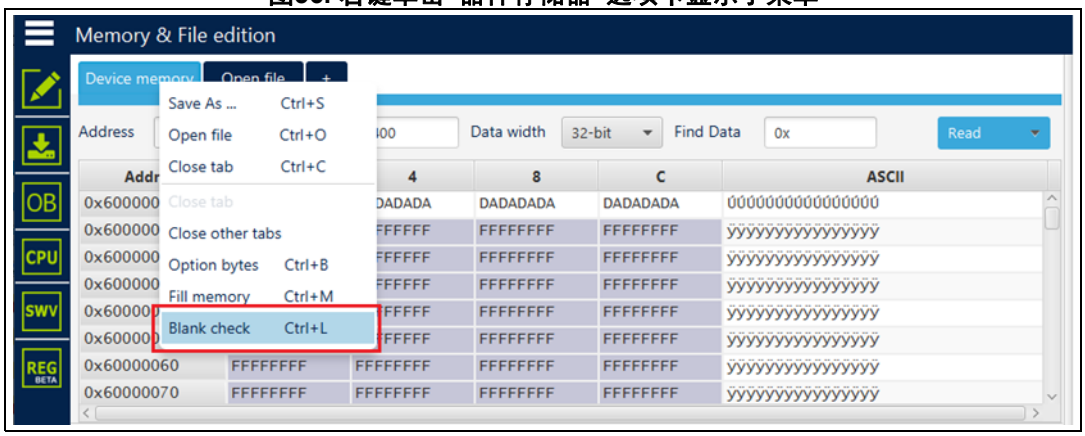


图57. 右键单击网格单元显示子菜单

注： *除了通过子菜单显示此窗口，用户还可以使用按键组合“Ctrl+L”直接启动操作。*

在点击“空白检查”子菜单后，过程开始确认STM32Flash存储器是否为空白状态。如果Flash存储器不是空白状态，则将在消息中高亮显示第一个包含数据的地址，如*图 58*所示。

预期结果如*图 59*和*图 60*所示。

图58. 第一个包含数据的地址

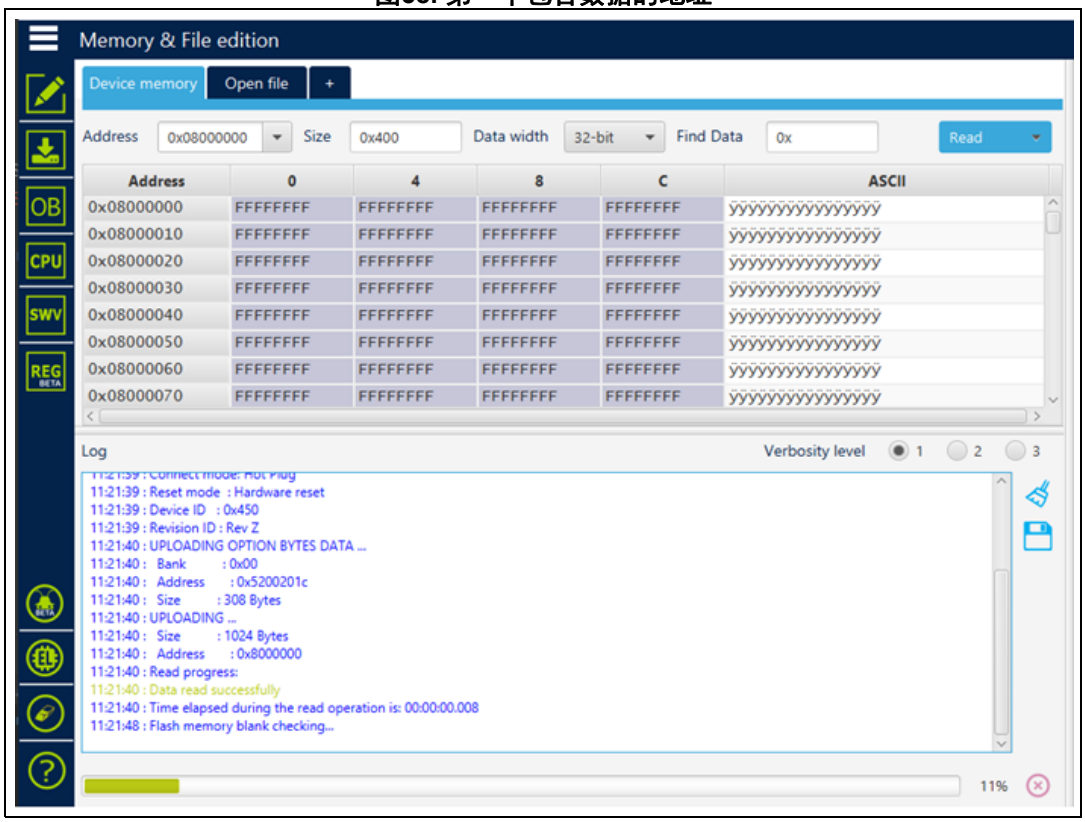


图59. 示例1：Flash存储器为空白状态

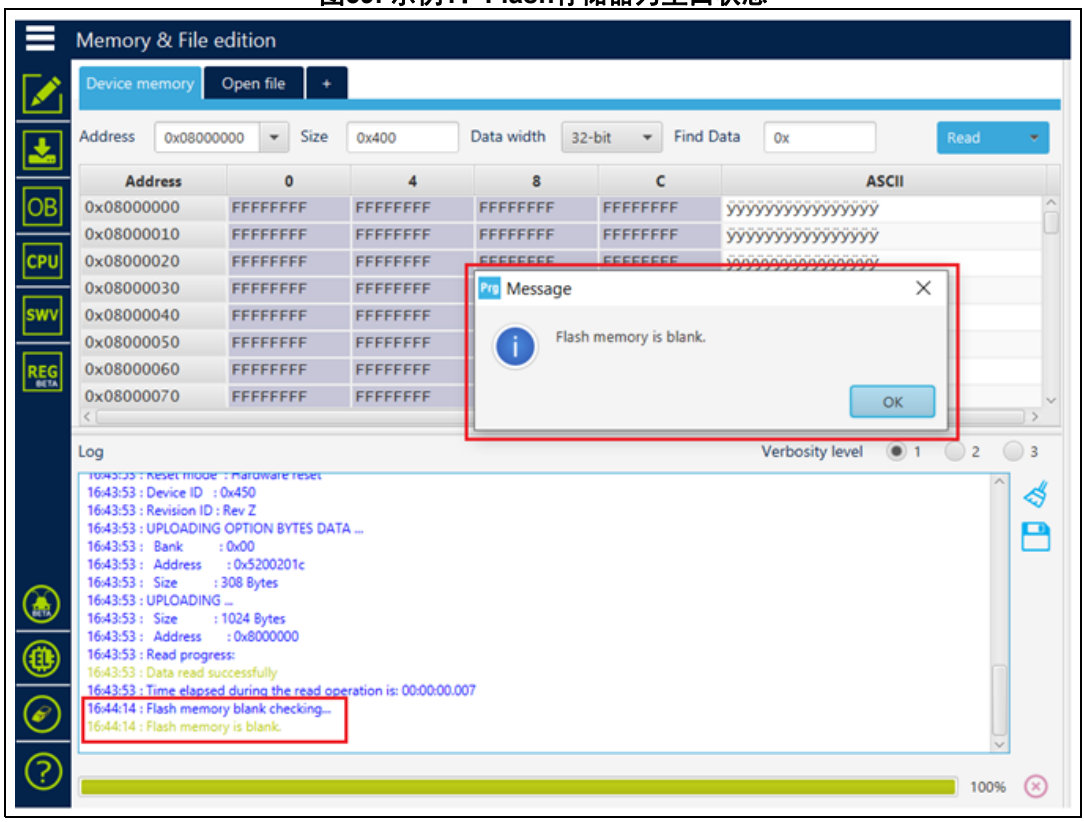
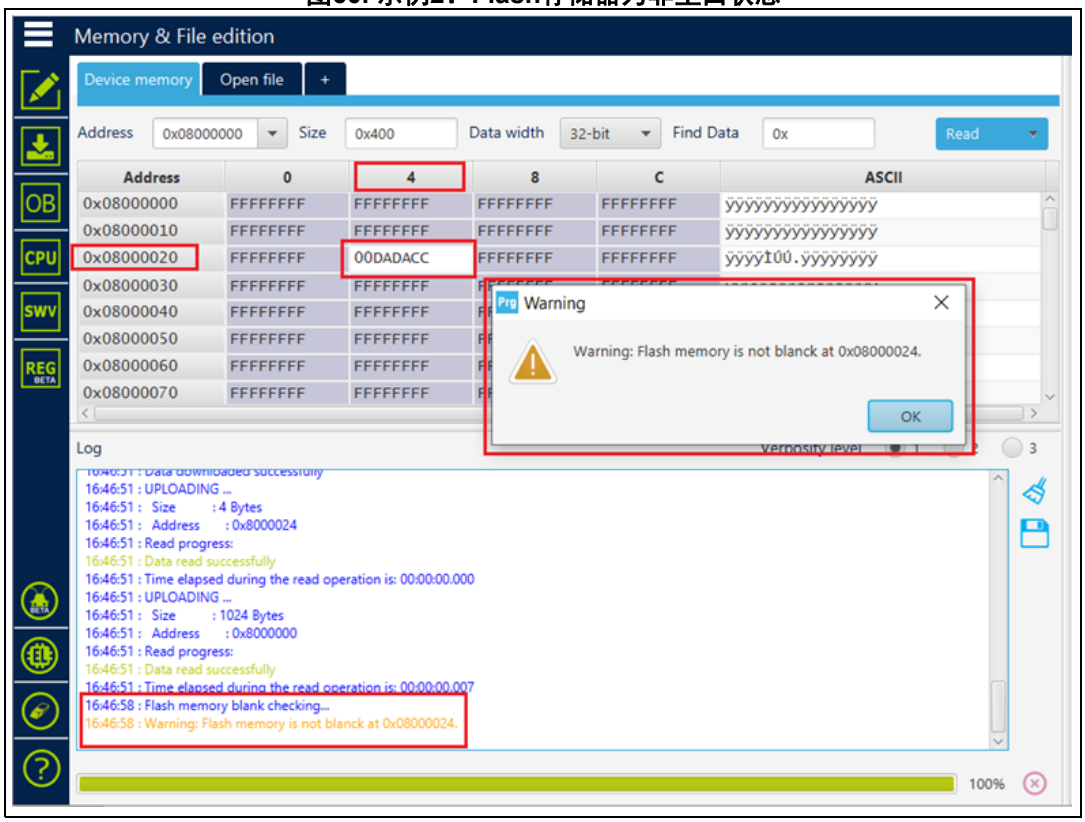


图60. 示例2：Flash存储器为非空白状态



2.18 比较Flash存储器和文件

说明：将MCU器件存储器内容与二进制、hex、srec、elf、out和axf文件进行比较。不同之处以红色显示在文件和Flash面板中。

用户可以从不同的子菜单打开比较窗口。

图61. 从“读取”组合框显示子菜单

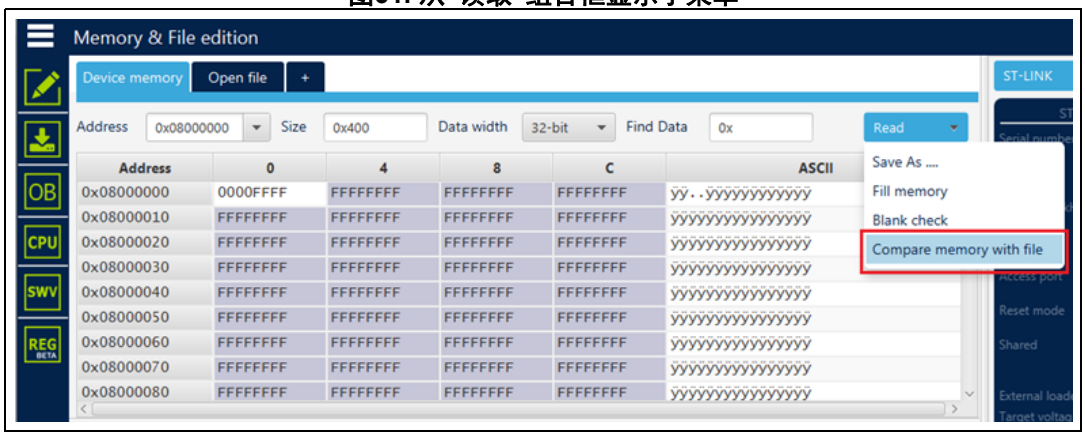


图62. 右键单击“器件存储器”选项卡显示子菜单

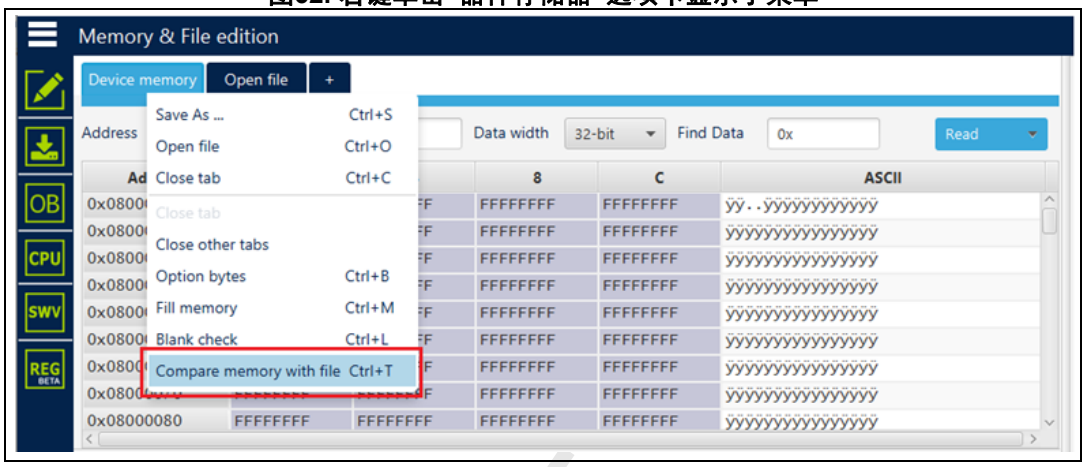


图63. 右键单击网格单元显示子菜单

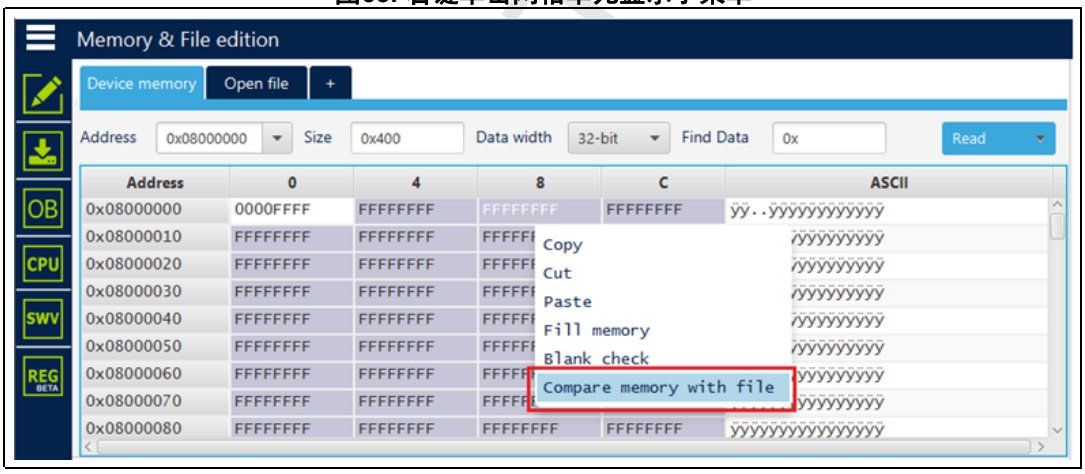


图64. 点击加号选项卡按钮显示子菜单

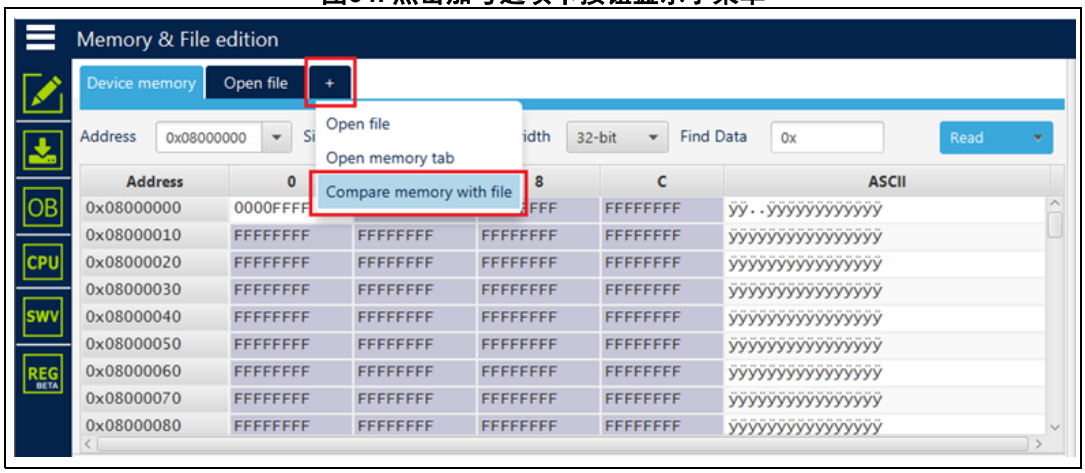


图65. 右键单击打开的文件选项卡显示子菜单

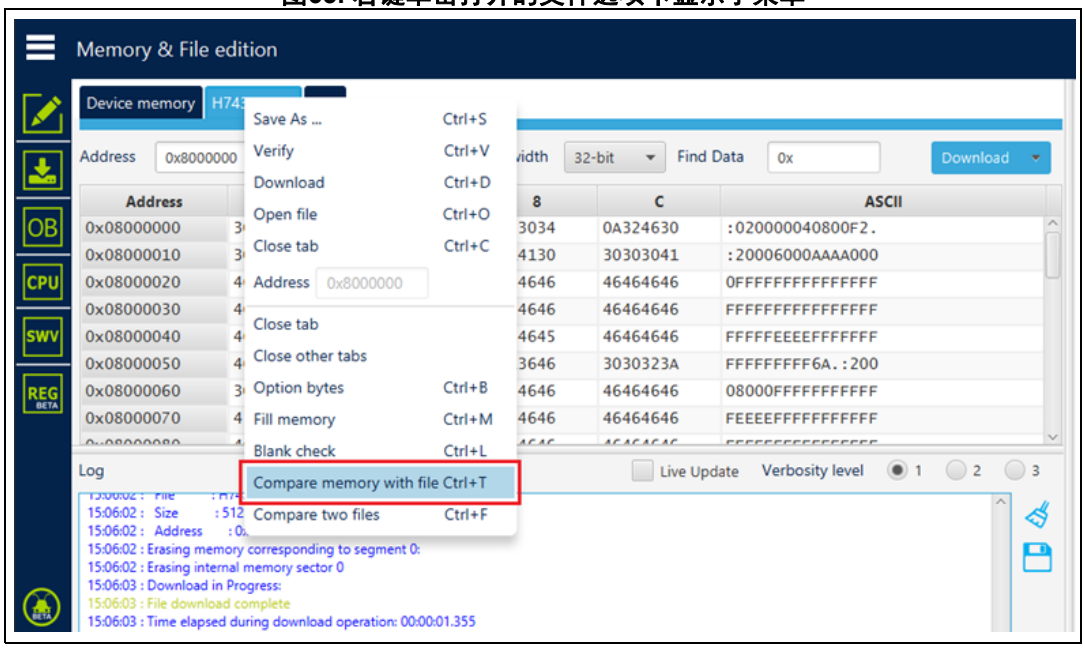
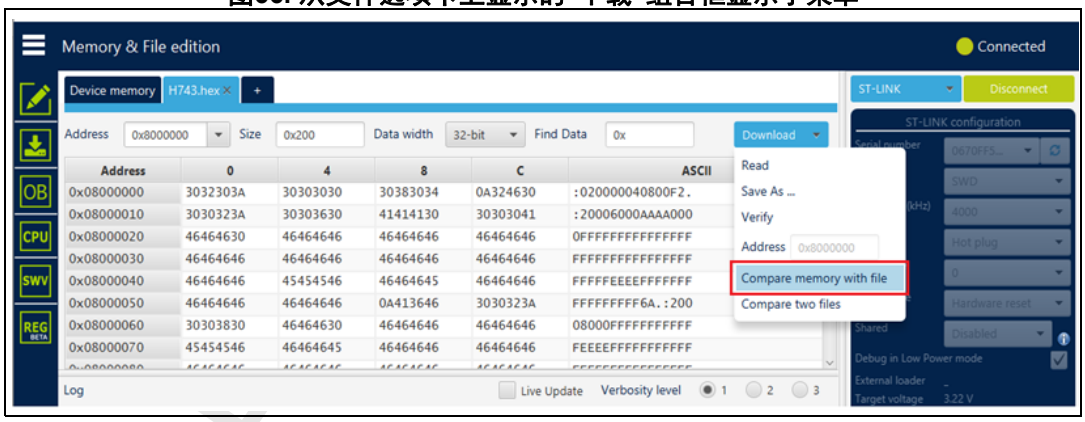


图66. 从文件选项卡上显示的“下载”组合框显示子菜单



注： *除了通过子菜单显示此窗口，用户还可以使用按键组合“Ctrl+T”直接启动操作。*

示例1：内部Flash存储器内容与二进制文件之间的差异

图67. 数据宽度：32位

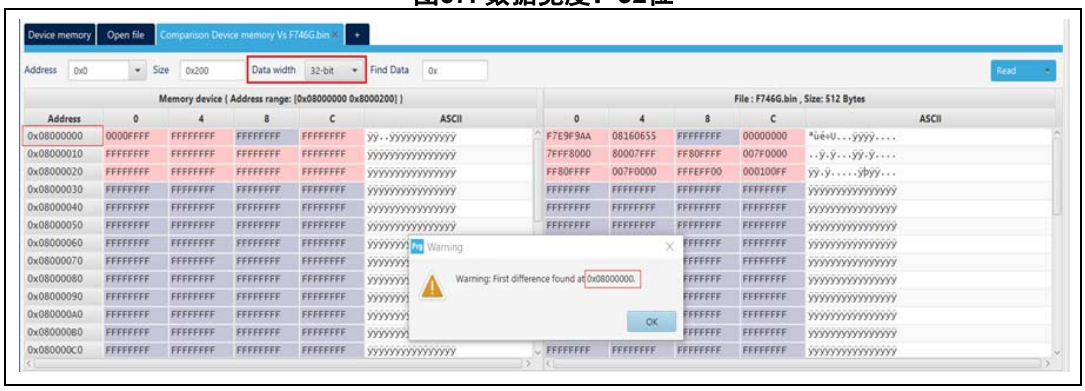


图68. 数据宽度：16位

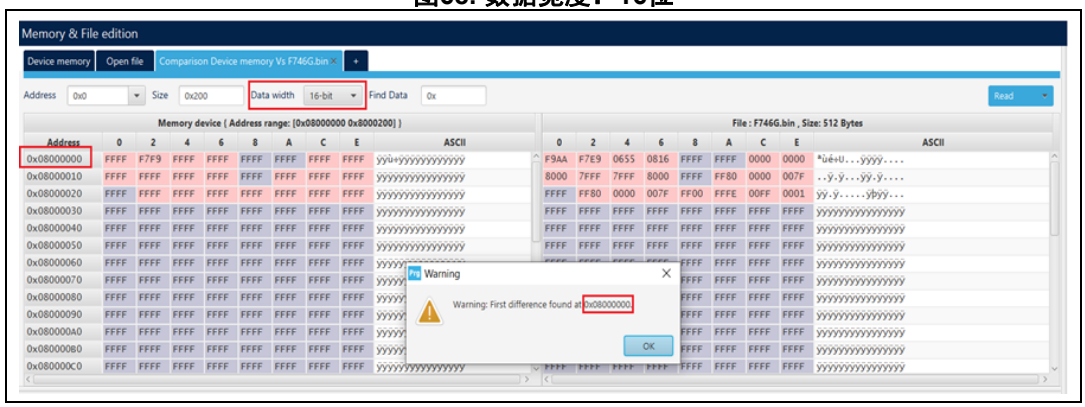
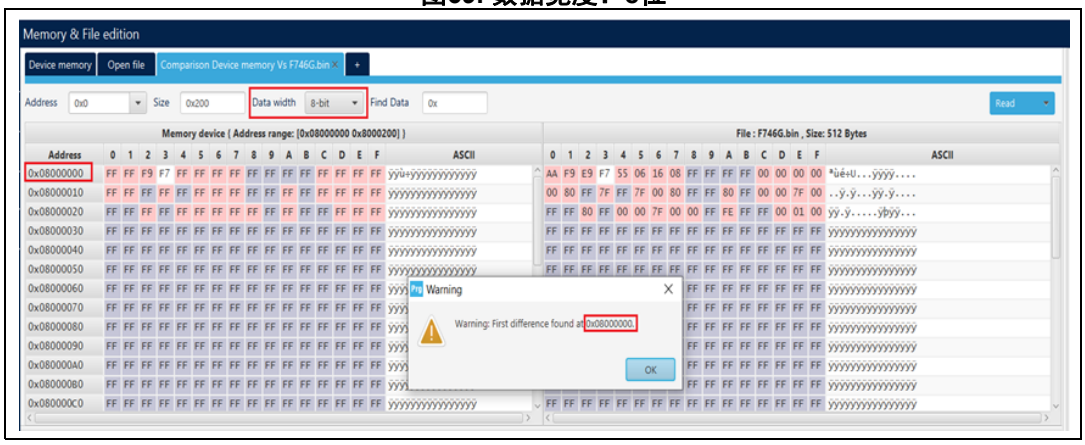


图69. 数据宽度：8位



示例2：外部Flash存储器内容与十六进制文件之间的差异

图70. 数据宽度：32位

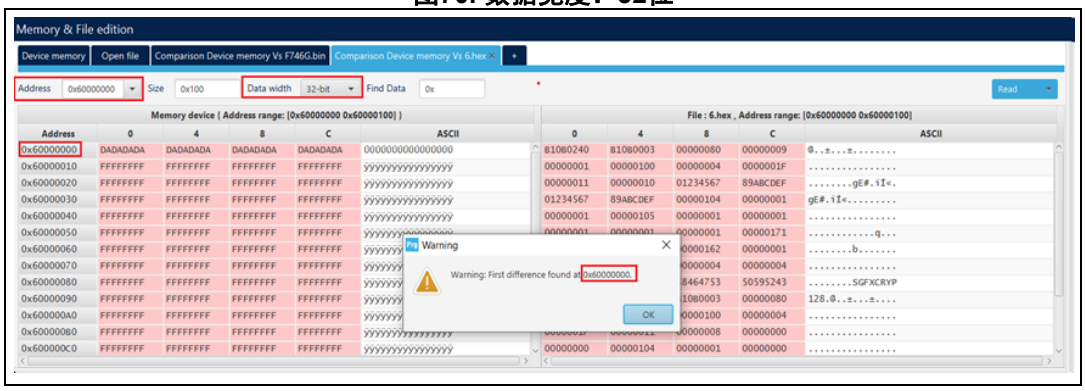


图71. 数据宽度：16位

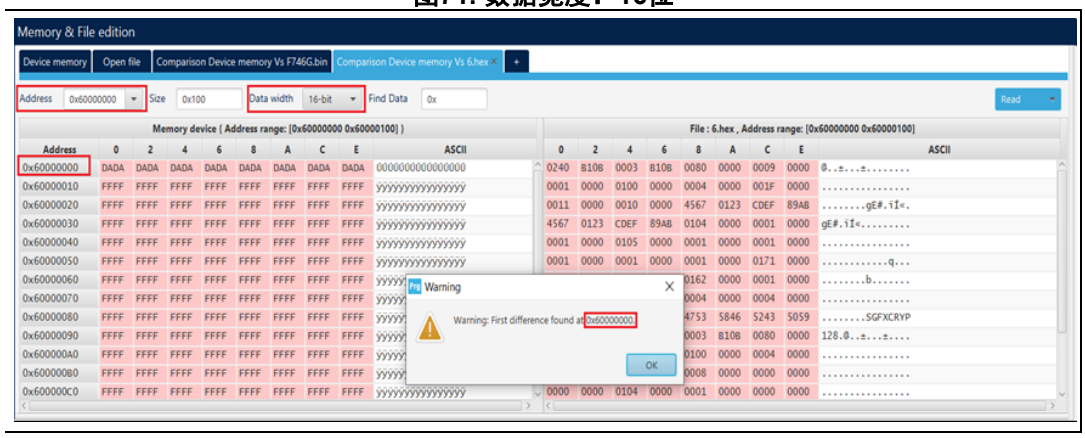
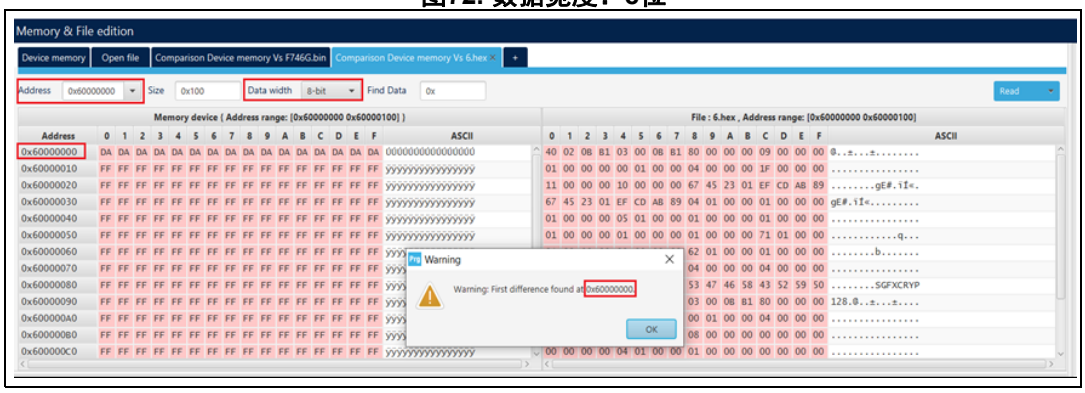


图72. 数据宽度：8位



如果在Flash存储器和文件之间的比较开始后编辑了存储器中的数据，则用户必须在比较选项卡上用读取按钮执行更新。

示例3：编辑后更新Flash存储器和文件之间的比较

图73. 编辑Flash存储器前

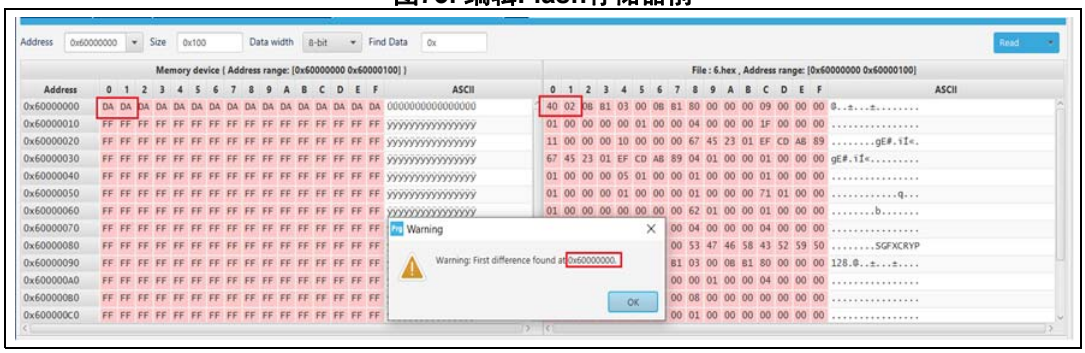
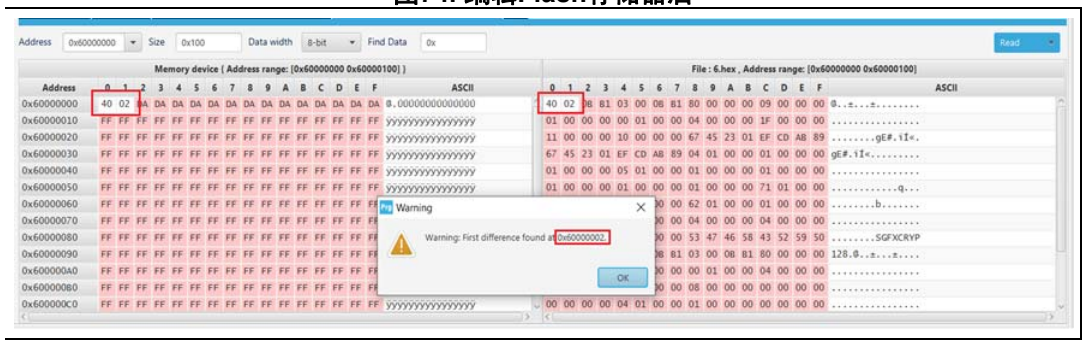
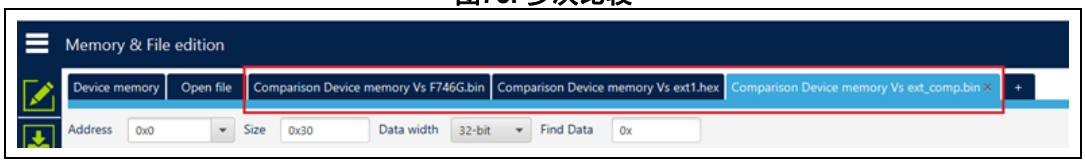


图74. 编辑Flash存储器后



注： *用户可以执行多次Flash存储器和文件之间的比较。*

图75. 多次比较



2.19 比较两个文件

说明：比较两个不同文件（二进制、hex、srec、elf、out和axf）的内容。网格面板上以红色显示每个文件的差异。

此操作无需连接板件。

使用的文件可以是不同大小和类型。

用户可以从不同的子菜单打开比较窗口。

图76. 从设备存储器选项卡上的“读取”组合框显示子菜单

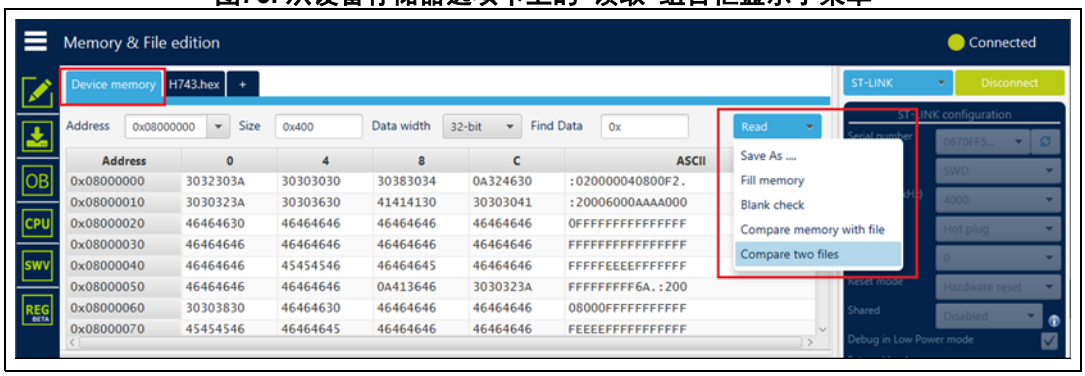


图77. 右键单击“器件存储器”选项卡显示子菜单

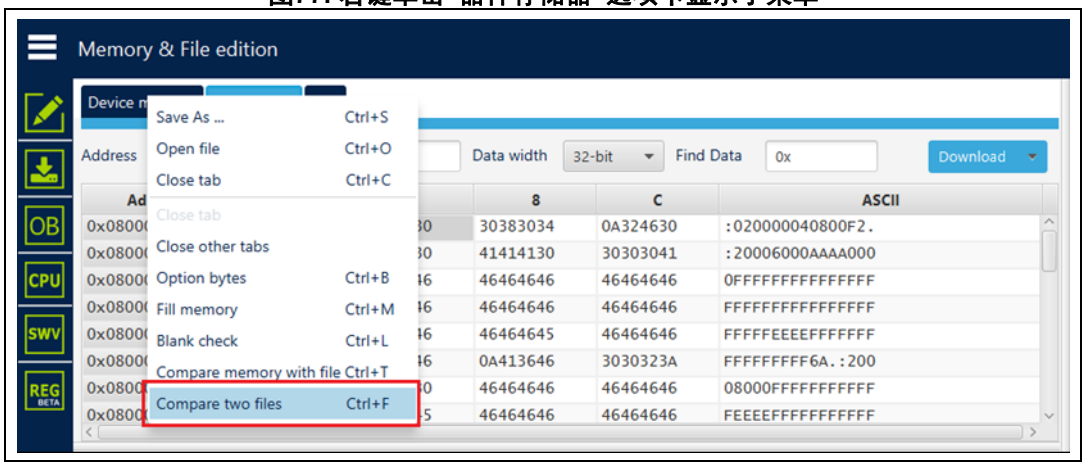


图78. 右键单击网格单元显示子菜单

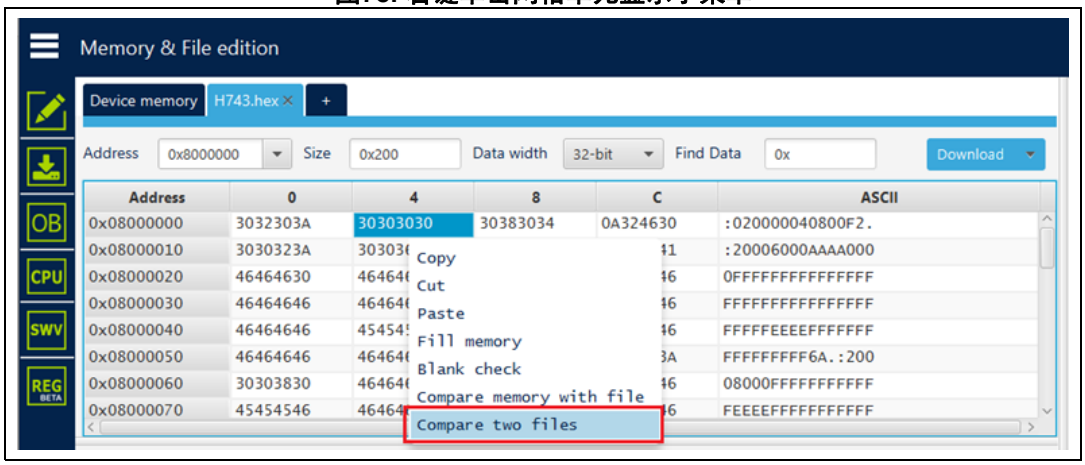


图79. 点击加号选项卡按钮显示子菜单

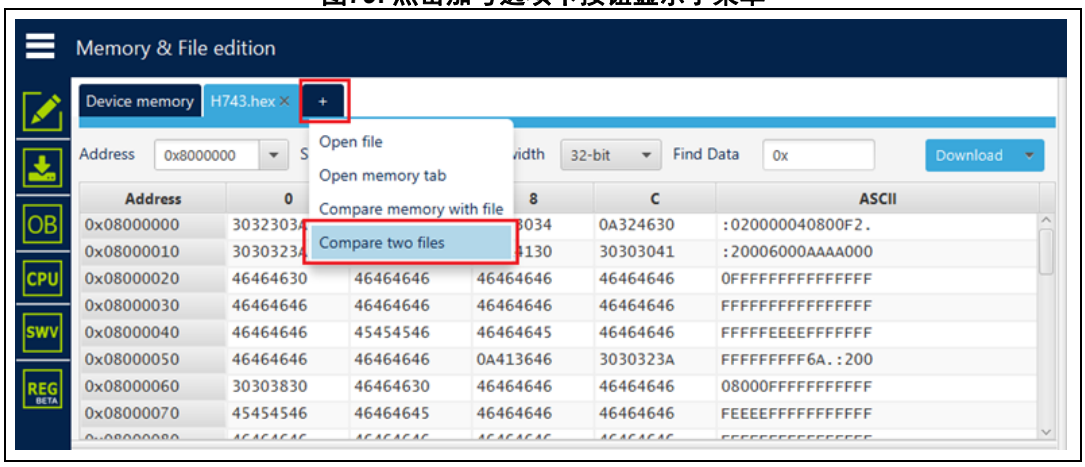


图80. 右键单击打开的文件选项卡显示子菜单

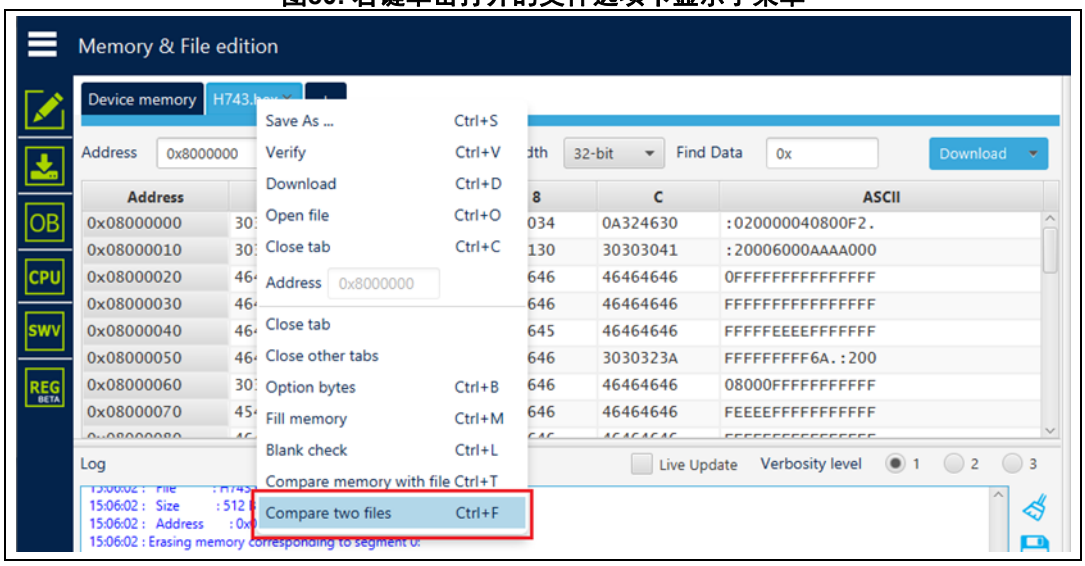
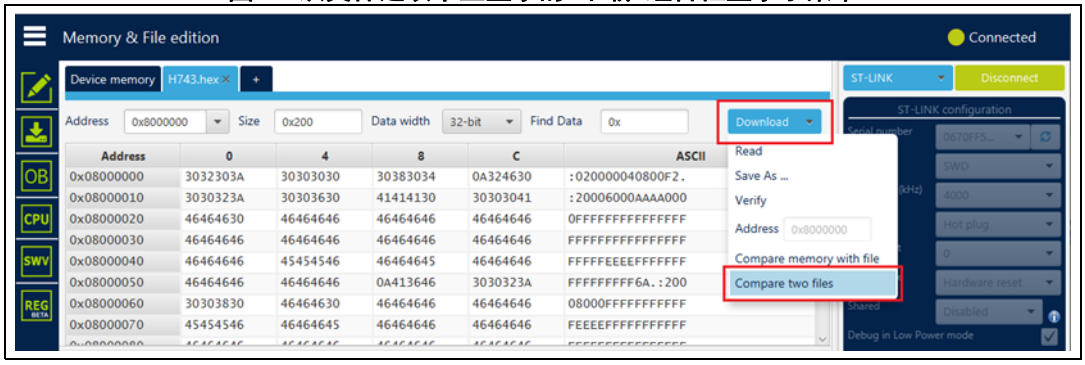


图81. 从文件选项卡上显示的“下载”组合框显示子菜单



注： *除了通过子菜单显示此窗口，用户还可以使用按键组合“Ctrl+F”直接打开它。*

示例：两个具有相同类型和不同大小的文件之间的差异

图82. 数据宽度：32位

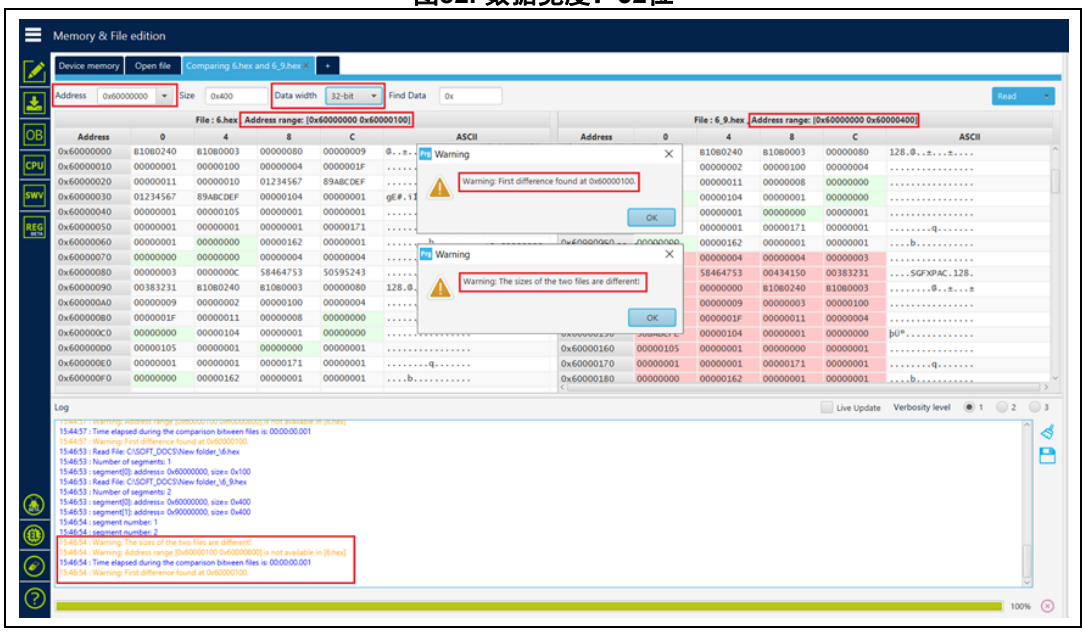


图83. 数据宽度：16位

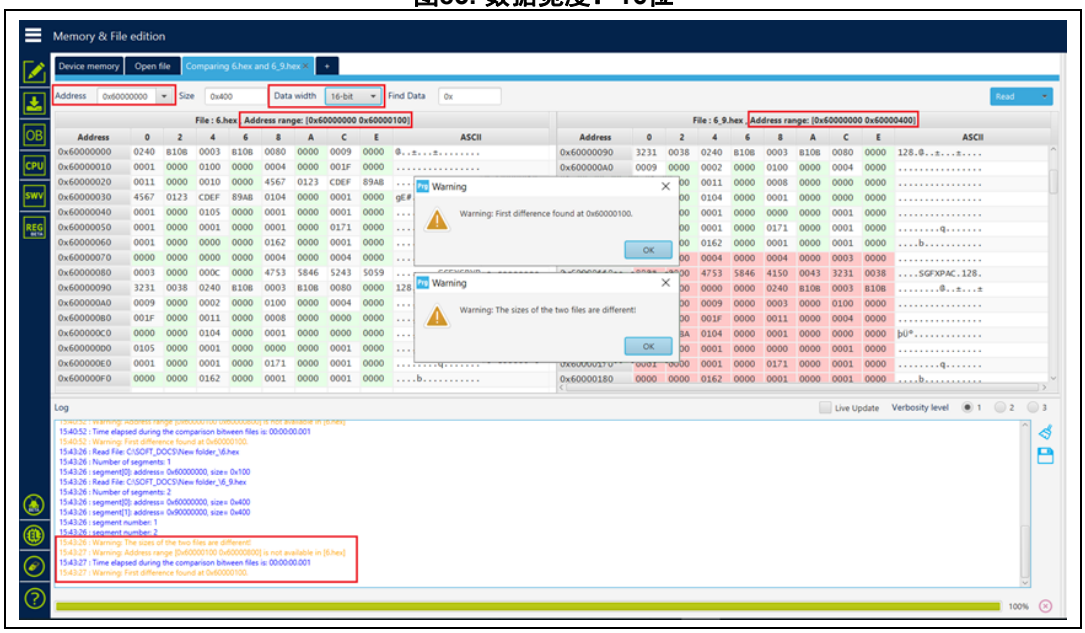
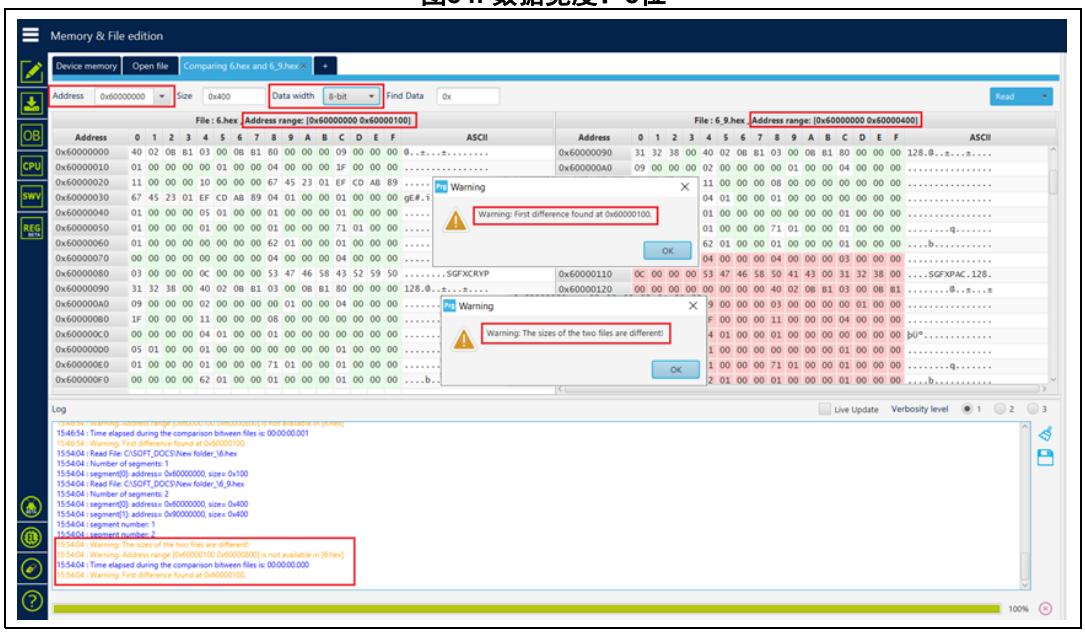
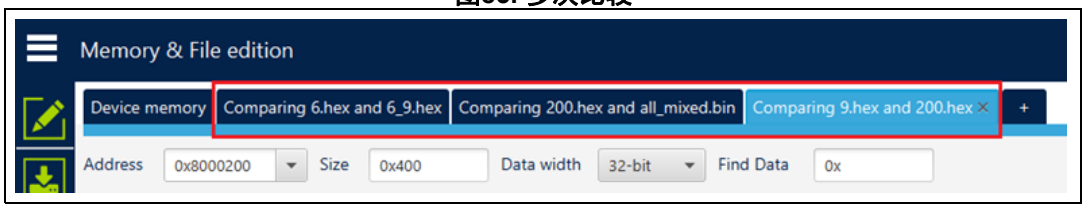


图84. 数据宽度：8位



注： *用户可以执行多次文件之间的比较。*

图85. 多次比较



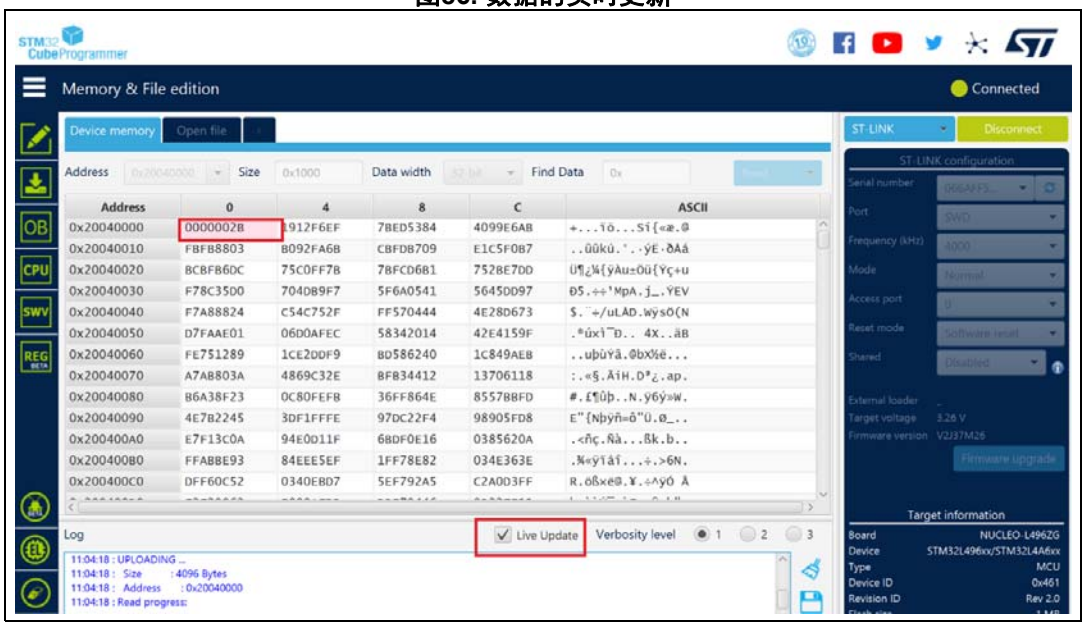
2.20 LiveUpdate特性

-liveUpdate复选框

说明：如果使用此特性，则器件存储器网格将实时更新，修改过的数据显示为粉红色。

在连接器件后，用户可以选中liveUpdate复选框，存储器数据将实时更新。

图86. 数据的实时更新



3 MCU的STM32CubeProgrammer命令行界面（CLI）

3.1 命令行的使用

以下各节介绍如何由命令行来使用STM32CubeProgrammer。可用指令如*图 87*中所示。

注： *为了在macOS上启动命令行界面，需要调用STM32CubeProgrammer.app/Contents/MacOs/bin/STM32_Programmer_CLI。*

图87. STM32CubeProgrammer：可用指令

```
Usage :
STM32_Programmer_CLI.exe [command_1] [Arguments_1][[command_2] [Arguments_2]...

Generic commands:

 -?, -h, --help          : Show this help
 -version, --version     : Displays the tool's version
 -l,     --list          : List all available communication interfaces
     <uart>              : UART interface
     <usb>               : USB interface
 -q,     --quietMode     : Enable quiet mode. No progress bar displayed
 -log,   --log           : Store the detailed output in log file
     [<File_Path.log>]   : Path of the log file,
                           default path = $HOME/.STM32Programmer/trace.log
 -vb,    --verbosity     : Specify verbosity level
     <Level>             : Verbosity level, value in <1, 2, 3>

Available commands for STM32 MCU

 --skipErase             : Skip sector erase before programming
 -sl,    --safelib       : Add a segment into a firmware file <elf,bin
                           hex,srec> containing computed CRC values
                           To use only with the safety lib component
     <file_path>         : File path to be modified
     <start_address>     : Flash memory start address
     <end_address>       : Flash memory end address
     <slice_size>        : Size of data per CRC value
 -ms,    --mergesbsfu    : Add a binary header and a sbsfu segment to an elf fil

     <elf_file_path>     : File path to be modified
     <header_file_path>  : Header file path
     <sbsfu_file_path>   : SBSFU file path
 -c,     --connect       : Establish connection to the device
     <port=<PortName>    : Interface identifier. ex COM1, /dev/ttyS0, usb1,
                           JTAG, SWD...>
    UART port optional parameters:
     [br=<baudrate>]     : Baudrate. ex: 115200, 9600, etc, default 115200
     [P=<parity>]        : Parity bit, value in <NONE,ODD,EVEN>, default EVEN
     [db=<data_bits>]    : Data bit, value in <6, 7, 8> ..., default 8
     [sb=<stop_bits>]    : Stop bit, value in <1, 1.5, 2> ..., default 1
     [fc=<flowControl>]  : Flow control
                           Value in <OFF,Hardware,Software) ..., default OFF
                           Not supported for STM32MP
     [noinit=noinit_bit]: Set No Init bits, value in <0,1) ..., default 0
     [console]           : Enter UART console mode
    JTAG/SWD debug port optional parameters:
     [freq=<frequency>] : Frequency in KHz. Default frequencies:
                           4000 SWD 9000 JTAG with STLINKv2
                           24000 SWD 21333 with STLINKv3
     [index=<index>]     : Index of the debug probe. default index 0
     [sn=<serialNumber>]: Serial Number of the debug probe
     [ap=<accessPort>]   : Access Port index to connect to. default ap 0
     [mode=<mode>]       : Connection mode. Value in <UR/HOTPLUG/NORMAL>
                           default mode: NORMAL
     [reset=<mode>]      : Reset modes: SWrst/HWrst/Crst. Default mode: SWreset
    SPI port optional parameters:
     [br=<baudrate>]     : Baudrate.
     [cpha=<cpha_val>]   : 1Edge or 2Edge. default 1Edge
     [cpol=<cpol_val>]   : low or high
     [crc=<crc_val>]     : enable or disable <0/1>.
     [crcpol=<crc_pol>]  : crc polynom value.
     [datasize=<size>]   : 8bit/16bit
     [direction=<val>]   : Direction: 2LFullDuplex/2LRxOnly/1LRx/1LTx
     [firstbit=<val>]    : First Bit: MSB/LSB
     [frameformat=<val>]: Frame Format: Motorola/TI
     [mode=<val>]        : Mode: master/slave
     [nss=<val>]         : NSS: soft/hard
     [nsspulse=<val>]    : NSS pulse: Pulse/NoPulse
     [delay=<val>]       : Delay: Delay/NoDelay, delay of fiew microseconds
     [noinit=noinit_bit]: Set No Init bits, value in <0,1) ..., default 0
    CAN port optional parameters:
     [br=<rbaudrate>]    : Baudrate : 125, 250, 500, 1000 Kbps, default 125
     [mode=<canmode>]    : CAN Mode : NORMAL, LOOPBACK..., default NORMAL
     [ide=<type>]        : CAN Type : STANDARD or EXTENDED, default STANDARD
     [rt=<format>]       : Frame Format: DATA or REMOTE, default DATA
     [fifo=<afifo>]      : Msg Receive : FIFO0 or FIFO1, default FIFO0
     [fm=<fmode]         : Filter Mode : MASK or LIST, default MASK
     [fs=<fscale>)]      : Filter Scale: 16 or 32, default 32
     [fe=<fenable>)]     : Filter Activation : ENABLE or DISABLE, default ENABLE
     [fb=<fbanknb>)]     : Filter Bank Number : 0 to 13, default 0
     [noinit=noinit_bit]: Set No Init bits, value in <0,1) ..., default 0
    I2C port optional parameters:
     [add=<ownadd>)]     : Slave address : address in hex format
     [br=<sbaudrate>)]   : Baudrate : 100 or 400 Kbps, default 400
     [sm=<smode>)]       : Speed Mode : STANDARD or FAST, default FAST
     [am=<addmode>)]     : Address Mode : 7 or 10 bits, default 7
     [af=<afilter>)]     : Analog filter : ENABLE or DISABLE, default ENABLE
     [df=<dfilter>)]     : Digital filter : ENABLE or DISABLE, default DISABLE
     [dnf=<dnfilter>)]   : Digital noise filter : 0 to 15, default 0
     [rt=<rtime>)]       : Rise time : 0-1000(STANDARD), 0-300(FAST), default 0
     [ft=<ftime>)]       : Fall time : 0-300 (STANDARD), 0-300(FAST), default 0
     [noinit=noinit_bit]: Set No Init bits, value in <0,1) ..., default 0
 -e,     --erase         : Erase memory pages/sectors devices:
                           Not supported for STM32MP
     [all]               : Erase all sectors
     [<sectorsCodes)>]   : Erase the specified sectors identified by sectors
                           codes. ex: 0, 1, 2 to erase sectors 0, 1 and 2
     [<[start end])>]    : Erase the specified sectors starting from
                           start code to end code, ex: -e [5 10]
 -w,     --write
 -d,     --download      : Download the content of a file into device memory
     <file_path>         : File path name to be downloaded: <bin, hex, srec,
                           elf, stm32 or tsv file>
     [<address>)]        : Start address of download
 -w32                    : Write a 32-bits data into device memory
     <address>           : Start address of download
     <32-bit_data>       : 32-bit data to be downloaded
                           values should be separated by space
 -v,     --verify        : Verify if the programming operation is achieved
                           successfully
 -r32                    : Read a 32-bit data from device memory
     <address>           : Read start address
     <size>              : Size of data
 -rst                    : Reset system
 -hardRst                : Hardware reset
                           Available only with JTAG/SWD debug port
 -halt                   : Halt core
 -step                   : Step core
                           Available only with JTAG/SWD debug port
 -score                  : Get core status
                           Available only with JTAG/SWD debug port
 -coreReg                : Read/Write core registers
     [<core_register>)]    R0/../R15/PC/LR/PSP/MSP/XPSR/APSR/IPSR/EPSR/
                           PRIMASK/BASEPRI/FAULTMASK/CONTROL
     [core_reg=<value>)]   value in case of write opration
                           Note: multiple registers can be handled at once
                           Available only with JTAG/SWD debug port

 -r,     --read
 -u,     --upload        : Upload the device memory content to a .bin file
     <address>           : Start address of read and upload
     <size>              : Size of memory content to be read
     <file_path>         : Binary file path

 -el,    --extload       : Select a custom external memory-loader
     <file_path>         : External memory-loader file path
 -s,     --start
 -g,     --go            : Run the code at the specified address.
     [<address>)]        : Start address
 -rdu,   --readunprotect: Remove memory's Read Protection by shifting the RDP
                           level from level 1 to level 0.

 -ob,    --optionbytes   : This command allows the user to manipulate the device
                           's OptionBytes by displaying or modifying them.
     [displ]             : This option allows the user to display the whole set
                           of Option Bytes.
     [OptByte=<value>)] : This option allows the user to program the given
                           Option Byte.
```

3.2 通用指令

本节介绍所有STM32 MCU都支持的一组指令。

3.2.1 连接指令

-c, --connect

说明：建立到设备的连接。该指令允许主机打开所选设备的端口（UART/USB/JTAG/SWD/SPI/CAN/I2C）。

语法：-c port=<Portname> [noinit=<noinit_bit>] [options]

port=<Portname	接口标识符，ex COMx（适用于Windows），/dev/ttySx（适用于Linux），usbx对应于USB接口，SPI、I2C和CAN分别对应于SPI、I2C和CAN接口。
[noinit=<noinit_bit>]	设置No Init位，值为{0, 1} …，默认值为0。如果之前的连接通常为激活状态，则可以使用Noinit = 1。

- ST-LINK选项

[freq=<frequency>]	连接中使用的频率，单位为kHz。SWD端口的默认值为4000 kHz，JTAG端口的默认值为9000 kHz

注： *输入的频率值四舍五入，以便与ST-LINK工具支持的值相匹配。*

[index=<index>]

[sn=<serialNumber>]	调试工具的序列号。如需连接序列号已知的特定ST-LINK工具，使用此选项。不要在同一连接指令中将此选项与索引选项一起使用。
[mode=<mode>]	连接模式。值为{NORMAL/UR/HOTPLUG}。默认值为NORMAL。
Normal	使用“正常”连接模式时，目标复位，随后挂起。使用“复位模式”选项来选择复位的方式。
UR	“复位状态下连接”模式允许在执行任何指令之前使用复位向量捕获连接到目标。这在很多情况下是很有用的，例如当目标包含了禁用JTAG/SWD引脚的代码时。
HOTPLUG	“热插拔”模式下可以在不停机或不复位的情况下连接到目标。这对于在应用运行时更新RAM地址或IP寄存器非常有用。
POWERDOWN	允许将目标置于调试模式，即使自目标上电后应用尚未启动。硬件复位信号必须连接在ST-Link与目标之间。此特性在某些使用STMPS2141电源开关的板件（MB1360、MB1319、MB1361和MB1355）上可能不完全有效。
[ap=<accessPort>]	访问端口索引。默认访问索引值为0。
[speed=]	连接速度。默认为可靠。 仅可用于Cortex-M33。
Reliable	允许用户以慢速模式连接。

Fast	允许用户以快速模式连接。
[shared]	启用共享模式可将STM32CubeProgrammer或其他调试器的两个或更多实例连接到同一ST-LINK工具。
[tcpport=<Port>]	选择TCP端口用于连接ST-Link服务器。必须选择共享选项。默认值为 7184。
[dLPM / LPM]	禁用/启用低功耗模式下的调试（为支持的器件（STM32U5/ WB/L4系列）启用默认配置）。

注： *仅Windows系统支持共享模式。*

- USB选项

DFU接口连接支持两个选项，即产品和供应商ID（默认值：PID=0xDF11，VID=0x0483）。

- SPI选项

[br=<baudrate>] 波特率（如187、375和750），默认值为375

注： *为了使用高速SPI，必须重视基础设施硬件以确保总线上的正确连接。*

[cpha=<cpha_val>]	1Edge或2Edge，默认为1Edge
[cpol=<cpol_val>]	低或高，默认为低
[crc=<crc_val>]	启用或禁用（0/1），默认为0
[crcpol=<crc_pol>]	CRC多项式值
[datasize=<size>]	8或16位，默认为8位
[direction=<val>]	2LFullDuplex/2LRxOnly/1LRx/1LTx
[firstbit=<val>]	MSB/LSB，默认为MSB
[frameformat=<val>]	Motorola/TI，默认为Motorola
[mode=<val>]	主设备/从设备，默认为主设备
[nss=<val>]	软/硬，默认为硬
[nsspulse=<val>]	脉冲/无脉冲，默认为脉冲
[delay=<val>]	延迟/无延迟，默认为延迟

- I2C选项

[add=<ownadd>] 从设备地址：十六进制格式的地址

注： *必须始终插入I2C地址，否则不建立连接。*

[br=<sbaudrate>] 波特率：100或400 Kbps，默认为400 Kbps。

[sm=<smode>] 速度模式：标准或快速，默认为快速。

[am=<addmode>] 地址模式：7或10位，默认为7位。

[af=<afilter>] 模拟滤波器：启用或禁用，默认为启用。

[df=<dfilter>] 数字滤波器：启用或禁用，默认为禁用。

[dnf=<dnfilter>] 数字噪声滤波器：0至15，默认为0。

[rt=<rtime>] 上升时间：0-1000（标准），0-300（快速），默认为0。

[ft=<ftime>] 下降时间：0-300（标准），0-300（快速），默认为0。

- CAN选项

[br=<rbaudrate>] 波特率：125、250…，默认为125。

[mode=<canmode>] 模式：正常、环回…，默认为正常。

注： *软件必须请求硬件进入正常模式，以便在CAN总线上实现同步并开始主机与CAN器件之间的接收和发送。推荐使用正常模式。*

[ide=<type>]

[rtr=<format>]

[fifo=<afifo>] 分配的FIFO：FIFO0或FIFO1，默认为FIFO0

[fm=<fmode] 滤波器模式：屏蔽或倾斜，默认为屏蔽

[fs=<fscale>]

[fe=<fenable>]

[fbn=<fbanknb>]

- 使用UART

./STM32_Programmer.sh -c port=/dev/ttyS0 br=115200

此示例的结果如*图 88*中所示。

图88. 使用RS232的连接操作

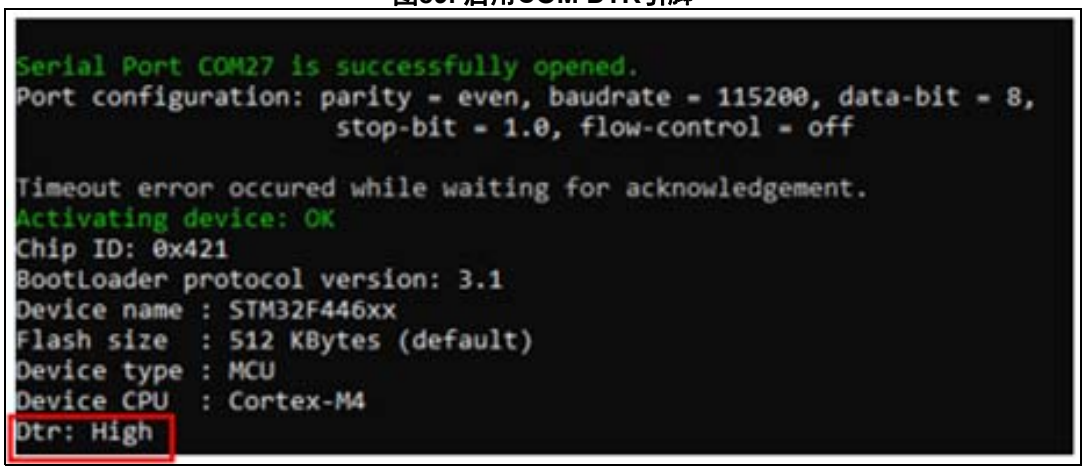
```
$ ./STM32_Programmer.sh -c port=/dev/ttyS0 br=115200
 Serial Port /dev/ttyS0 is successfully opened.
   Port configuration: parity = none, baudrate = 115200, data-bit = 8,
                       stop-bit = 1.0, flow-control = off
 Activating device: OK
 Chip ID: 0x500
 BootLoader version: 3.1
```

STM32CubeProgrammer提供配置RTS和DTR引脚的功能：

- RTS，使用方式如下：rts=low
- DTR，使用方式如下：dtr=high

示例：STM32_Programmer_CLI.exe -c port=COM27 dtr=high（参见*图 89*）。

图89. 启用COM DTR引脚



使用USB的示例

./STM32_Programmer.sh -c port=usb1

此示例的结果如*图 90*中所示。

图90. 使用USB的连接操作

```
establishing connection with the target device

USB speed             : FULL_SPEED(12MBit/s)
Manufacturer ID       : STMicroelectronics
Product ID            : STM32  BOOTLOADER
Serial number         : 326F37603234
Firmware version      : 1.1a
Device ID             : 0x0419

   AREA NAME          SECT.NBR        ADDRESS         SIZE          TYPE

   Internal Flash       0000          0x08000000      0016 KB       REW
                        0001          0x08004000      0016 KB       REW
                        0002          0x08008000      0016 KB       REW
                        0003          0x0800c000      0016 KB       REW
                        0004          0x08010000      0064 KB       REW
                        0005          0x08020000      0128 KB       REW
                        0006          0x08040000      0128 KB       REW
                        0007          0x08060000      0128 KB       REW
                        0008          0x08080000      0128 KB       REW
                        0009          0x080a0000      0128 KB       REW
                        0010          0x080c0000      0128 KB       REW
                        0011          0x080e0000      0128 KB       REW
                        0012          0x08100000      0016 KB       REW
                        0013          0x08104000      0016 KB       REW
                        0014          0x08108000      0016 KB       REW
                        0015          0x0810c000      0016 KB       REW
                        0016          0x08110000      0064 KB       REW
                        0017          0x08120000      0128 KB       REW
                        0018          0x08140000      0128 KB       REW
                        0019          0x08160000      0128 KB       REW
                        0020          0x08180000      0128 KB       REW
                        0021          0x081a0000      0128 KB       REW
                        0022          0x081c0000      0128 KB       REW
                        0023          0x081e0000      0128 KB       REW

   Option Bytes         0000          0x1fffc000      0016 B        RW
                        0001          0x1ffec000      0016 B        RW

   OTP Memory           0000          0x1fff7800      0512 B        RW
                        0001          0x1fff7a00      0016 B        RW

   Device Feature       0000          0xffff0000      0004 B        RW
```

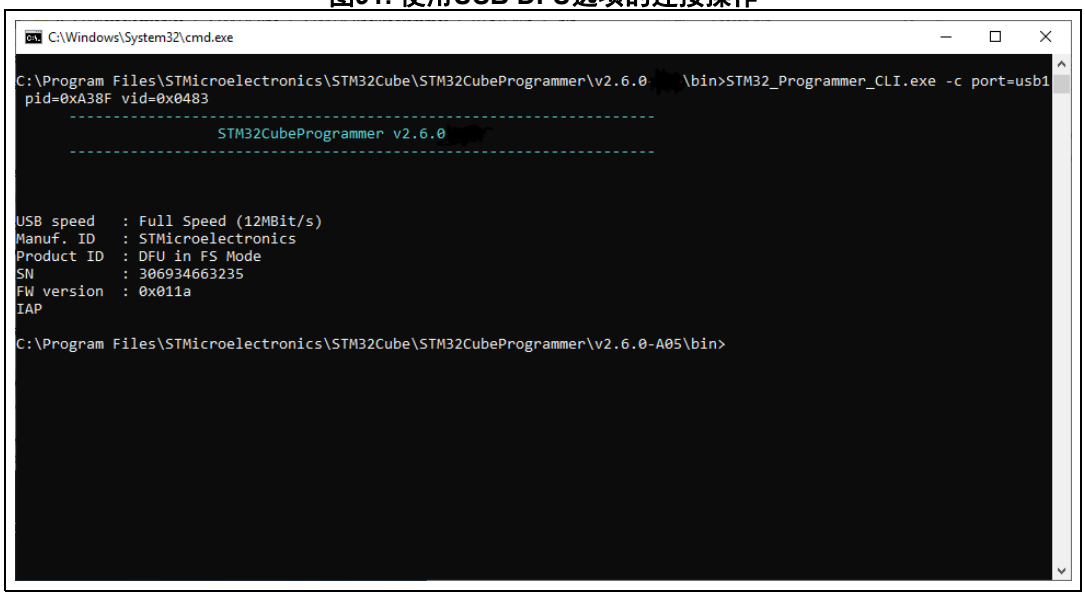
注： *当使用USB接口时，所有配置参数（如波特率、奇偶校验、数据位、频率和索引）都会被忽略。为了使用UART接口进行连接，端口配置（波特率、奇偶校验、数据位、停止位和流控制）必须根据使用的器件进行有效组合。*

使用DFU IAP/USBx选项的示例

/STM32_Programmer.sh -c port=usb1 pid=0xA38F vid=0x0438

此示例的结果如*图 91*中所示。

图91. 使用USB DFU选项的连接操作



注： *产品ID和供应商ID的默认值为ST产品值（PID=0xDF11，VID=0x0483）。*

使用JTAG/SWD调试端口的示例

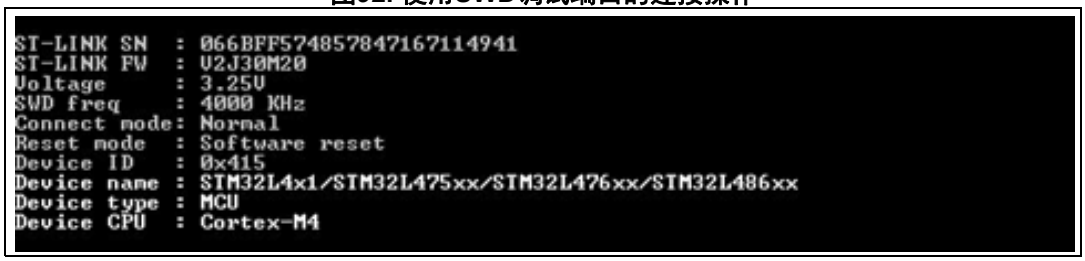
为了使用端口连接模式连接ST-LINK工具，至少需要使用连接指令说明端口名称（例如：-c port=JTAG）。

注： *在尝试通过JTAG进行连接时，请确保所用设备包含有JTAG调试端口。*

还有其他一些与JTAG/SWD调试端口有关的参数，它们都使用默认值（请参阅工具的帮助菜单获取有关默认值的更多信息）。

以下示例显示了STM32连接到设备ID 0x415的示例。

图92. 使用SWD调试端口的连接操作



这个例子的对应命令行是–c port=SWD freq=3900 ap=0

在连接指令中 (-c port=SWD freq=3900 ap=0)

- <port>参数是必需的。
- 该命令行中未提及索引。索引参数取默认值0。
- 输入的频率为3900 kHz，但是利用4000 kHz来建立连接。这是由于使用SWD和JTAG调试端口时，ST-LINK工具使用固定值。
- ST-LINK v2/v2.1
  - SWD（4000、1800、950、480、240、125、100、50、25、15、5）kHz
  - JTAG（9000、4500、2250、1125、562、281、140）kHz
- ST-LINK v3
  - SWD（24000、8000、3300、1000、200、50、5）
  - JTAG（21333、16000、12000、8000、1777、750）

如果输入的值不符合这些值中的任何一个，则会考虑次高值。默认频率值为：

- SWD：STLinkV2：4000 kHz，STLinkV3：24000 kHz
- JTAG：STLinkV2：9000 kHz，STLinkV3：21333 kHz

注： *JTAG频率选择仅支持从V2J23开始的ST-LINK固件版本。*

为了连接到访问端口0，本例中使用了ap参数，因此连接指令之后使用的任何指令都要通过选定的访问端口来建立。

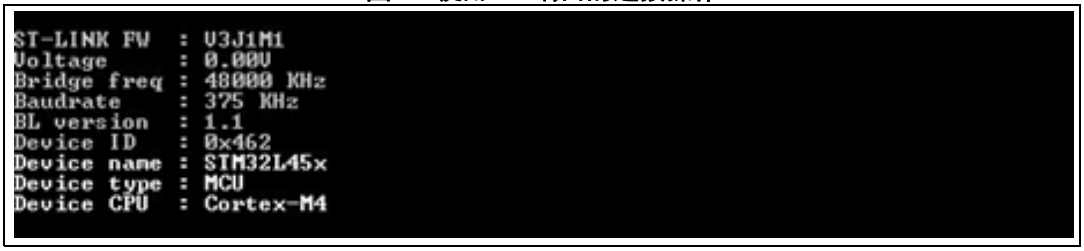
注： *连接到设备时会显示ST-LINK工具固件版本。务必使用ST-LINK固件V2J28M17（STSW-LINK007）的最新版本，可从ST网站www.st.com获取最新版本。*

使用SPI的示例

STM32_Programmer_CLI -c port=SPI br=375 cpha=1edge cpol=low

此示例的结果如*图 93*中所示。

图93. 使用SPI端口的连接操作



注： *在尝试通过SPI进行连接时，确保所用器件支持SPI引导加载程序。*

与SPI端口的连接中使用了其他具有默认值的参数，还有其他一些参数必须具有特定值（参见工具的帮助菜单了解更多信息）。

使用CAN的示例

STM32_Programmer_CLI -c port=CAN br=125 fifo=fifo0 fm=mask fs=32 fe=enable fbn=2

此示例的结果如*图 94*中所示。

图94. 使用CAN端口的连接操作

注： *并非所有器件都实现此特性，确保您使用的器件支持CAN引导加载程序。*

与CAN端口的连接中使用了其他具有默认值的参数，还有其他一些参数必须具有特定值（参见工具的帮助菜单了解更多信息）。

使用I2C的示例

STM32_Programmer_CLI -c port=I2C add=0x38 br=400 sm=fast

在连接指令中：

- 参数<add>从一个器件变为另一个，请参考AN2606选取正确的器件。在本例中，STM32F42xxx的引导加载程序地址为0x38。
- 波特率参数
直接取决于速度模式参数<sm>，例如：如果sm=standard，则波特率不支持值400。

此示例的结果如*图 95*中所示。

图95. 使用I2C端口的连接操作

注： *每个I2C连接操作的地址参数都是强制性的。*

注： *并非所有器件都实现此特性，确保器件支持I2C引导加载程序。*

与I2C端口的连接中使用了其他具有默认值的参数，还有其他一些参数必须具有特定值（参见工具的帮助菜单了解更多信息）。

注： *为了使用多个STM32CubeProgrammer实例进行一个以上STM32器件的并行编程，必须在合适的实例中添加每个器件的序列号，如下面的示例所示：*

- *“–c port=swd/usb sn=SN1”（STM32CubeProgrammer的实例1）*
- *“–c port=swd/usb sn=SN2”（STM32CubeProgrammer的实例2）*
- *“–c port=swd/usb sn=SN3”（STM32CubeProgrammer的实例3）*

3.2.2 擦除指令

-e, --erase

说明：根据给定的参数，该指令可用于擦除特定扇区或整个Flash存储器。完成此操作可能需要1秒或更长时间，具体取决于要擦除的对象的大小。

语法：

[all]

[<sectorsCodes>]	擦除代码标识的扇区（如**0,1,2**表示擦除扇区0、1和2）。对于EEPROM：**ed1 & ed2**。
[<[start end]>]	擦除从起始代码到结束代码之间的指定扇区，如**-e [5 10]**。

示例

./STM32_Programmer.sh --connect port=/dev/ttyS0 -e 2 4

此指令只擦除扇区2和4。

注： *如果有多个外部加载程序，则首选外部存储器擦除过程中考虑的程序。*

3.2.3 下载指令

-w, --write, -d, --download

说明：将指定二进制文件中的内容下载到设备存储器中。下载操作之前先进行擦除操作，然后下载Flash存储器。写入地址仅用于下载二进制文件。

语法： -w <file_path> [start_address]

[file_path]	要下载的文件路径
[start_address]	下载的起始地址

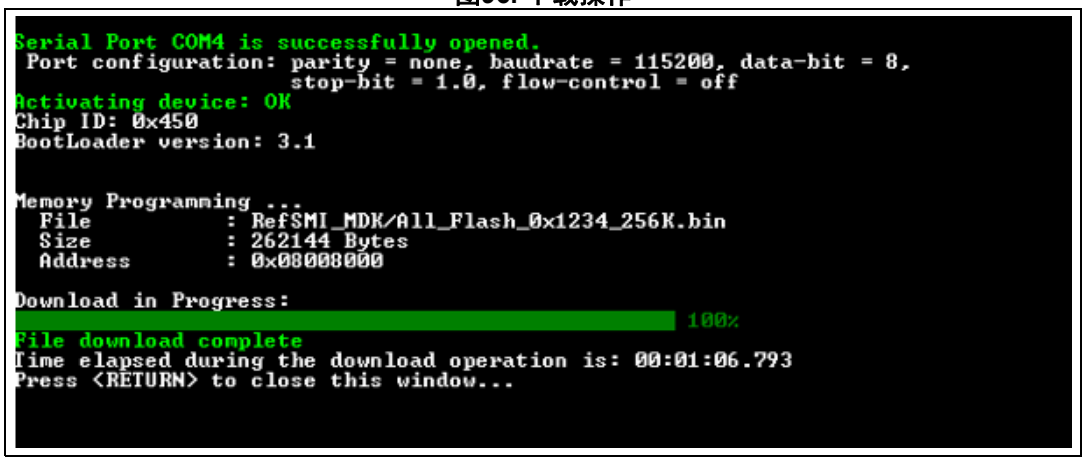
示例

-c port=COM4 -w RefSMI_MDK/All_Flash_0x1234_256K.bin 0x08008000

该指令将二进制文件“All_Flash_0x1234_256K.bin”编程到地址0x08008000处。

此示例的结果如*图 96*中所示。

图96. 下载操作



注： *为了确认下载成功，在写入指令后立即调用确认选项（-v或-verify），否则将忽略确认选项。*

3.2.4 下载32位数据指令

-w32

说明： 将指定的32位数据从指定地址开始下载到Flash存储器中。

语法： -w32 <start_address> <32_data_bits>

<start_address> 下载的起始地址。

<32_data_Bits> 要下载的32位数据。数据必须用换码字符分隔。

示例

./STM32_Programmer.sh -c port=/dev/ttyS0 br=9600 -w32 0x08000000 0x12345678 0xAABBCCFF 0x12AB34CD –verify

注： *该指令能够将32位数据（0x12345678、0xAABBCCFF和0x12AB34CD）从地址0x08000000开始写入Flash存储器。*

3.2.5 下载64位数据指令

-w64

说明：将指定的64位数据下载到目标地址。

语法： -w64 <start_address> <64-bit_data>

<start_address> 下载的起始地址。

<64_data_Bits> 要下载的64位数据。数据必须用换码字符分隔。

示例：

/STM32_Programmer_CLI.exe -c port=swd –w64 0x08000000 0x12345678AABBCCFF

3.2.6 Read指令

-r, --read, -u, --upload

说明：从指定地址开始读取设备存储器内容并将其上传到指定的二进制文件中。

语法： --upload <start_address> <size> <file_path>

<start_address>	读取的起始地址。
<size>	要读取的存储器内容大小。
<file_path>	上载存储器内容的二进制文件路径。

示例

./STM32_Programmer.sh -c port=/dev/ttyS0 br=9600 --upload 0x20007000 2000 “/local/ benayedh/Binaries/read2000.bin”

此指令可以从地址0x20007000开始读取2000个字节，并将内容上传到二进制文件 *“/local/benayedh/Binaries/read2000.bin”*

-r32

说明：读取32位数据存储器。

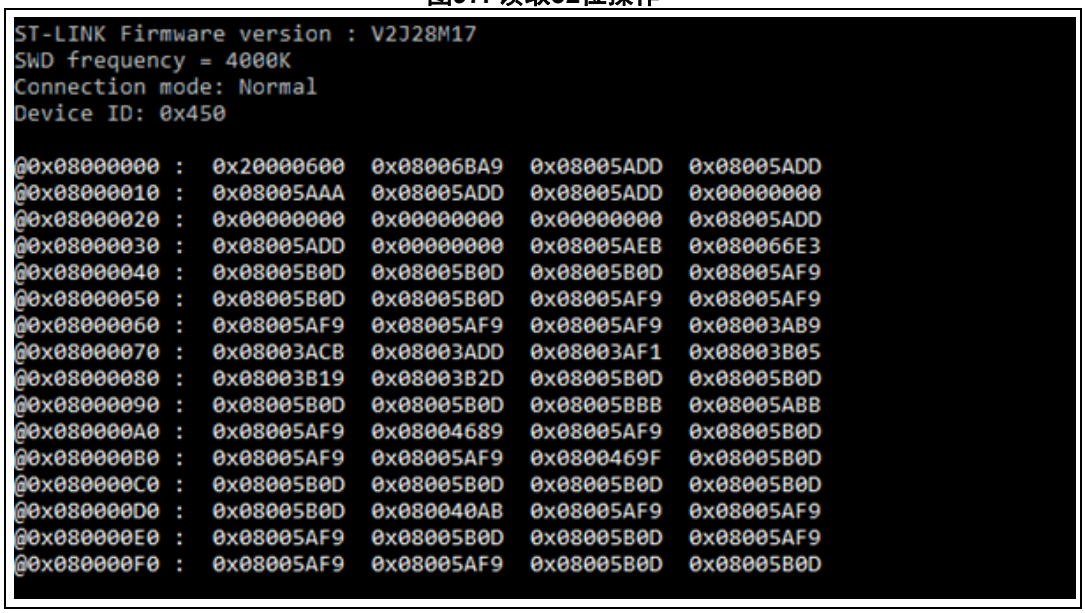
语法： -r32 <start_address> <size>

<start_address>	读取的起始地址。
<size>	要读取的存储器内容大小。

示例

./STM32_Programmer.sh -c port=SWD –r32 0x08000000 0x100

图97. 读取32位操作



注： *-r32指令最大支持32KB。*

3.2.7 start指令

-g, --go, -s, --start

说明：该指令允许从指定地址开始执行设备存储器中的程序。

语法： --start [start_address]

[Start_address] 要执行的应用程序起始地址。

示例

./STM32_Programmer.sh --connect port=/dev/ttyS0 br=9600 --start 0x08000000

该指令运行在0x08000000处指定的代码。

3.2.8 调试指令

下列指令仅在使用JTAG/SWD调试端口时可用。

-rst

说明：执行软件系统复位；

语法：**-rst**

-hardRst

说明：通过调试连接器中的RESET引脚生成硬件复位。

JTAG连接器（引脚15）的RESET引脚须连接到器件复位引脚。

语法：**-hardRs**t

-halt

说明：停止内核。

语法：**-halt**

-step

说明：执行一个指令。

语法：**-step**

-score

说明：显示Cortex-M内核状态。

内核状态可以是以下状态中的一种：“运行”、“暂停”、“锁定”、“复位”和“锁定或保持复位状态”

语法：**-score**

-coreReg

说明：读取/写入Cortex-M内核寄存器。在执行读取/写入操作之前，内核暂停。

语法：**-coreReg [<core_register>]**
R0/../R15/PC/LR/PSP/MSP/XPSR/APSR/IPSR/EPSR/PRIMASK/BASEPRI/FAULTMASK/CONTROL

[core_reg=<value>]：写入操作要在内核寄存器中写入的值。可同时处理多个寄存器。

示例

-coreReg	此指令显示内核寄存器的当前值。
-coreReg R0 R8	此指令显示R0和R8的当前值。
-coreReg R0=5 R8=10	此指令修改R0和R8的值。

3.2.9 列表指令

-l, -list

说明：此指令对所有UART、DFU和STLink接口可用。

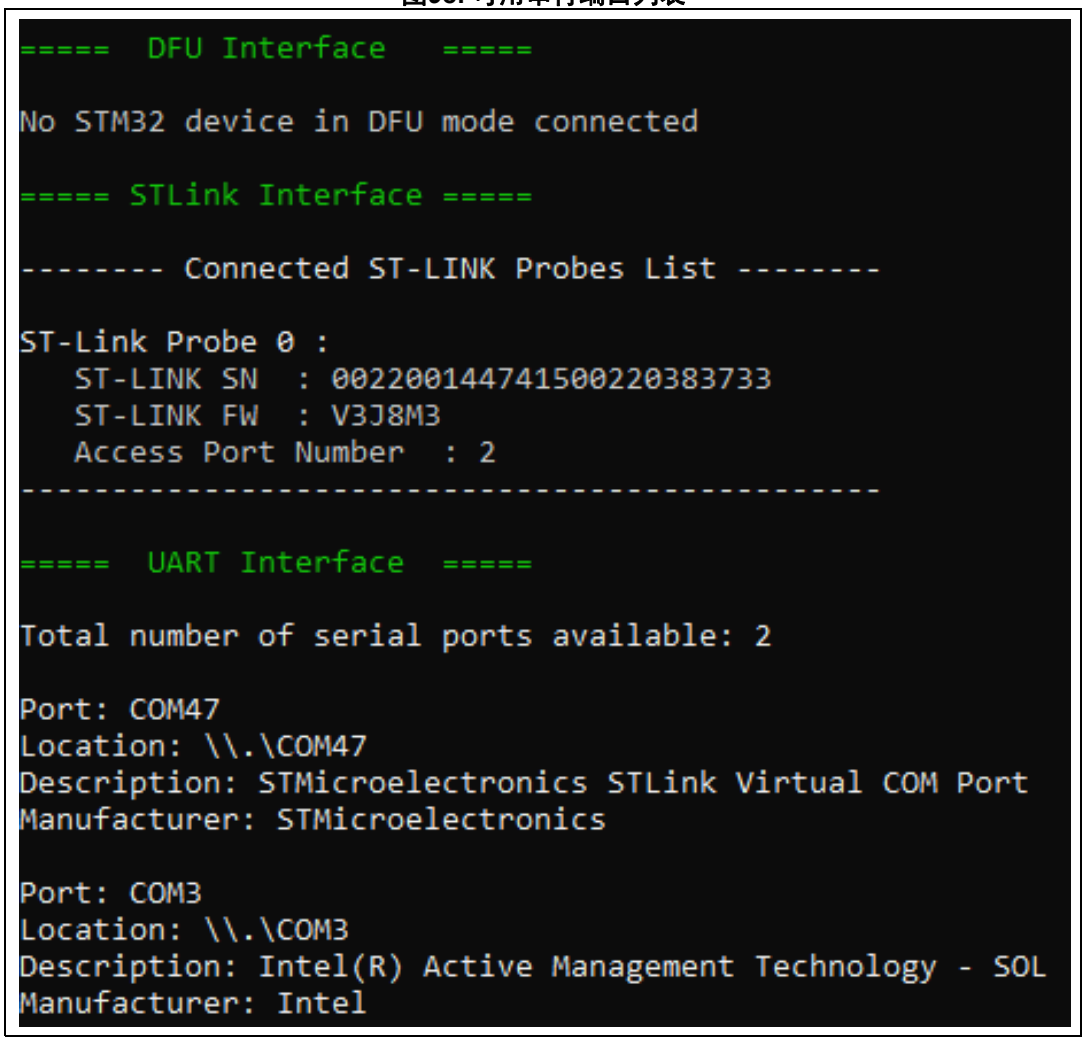
语法： **-l, --list**

示例

./STM32_Programmer.sh --list

此示例的结果如*图 98*中所示。

图98. 可用串行端口列表



3.2.10 QuietMode指令

-q, --quietMode

说明：该指令在下载和读取指令期间禁用进度条显示。

语法： -q, --quietMode

示例

./STM32_Programmer.sh –c port=/dev/ttyS0 br=115200 –quietMode –w binaryPath.bin 0x08000000

3.2.11 Verbosity指令

-vb, --verbosity

说明：此指令可以显示更多消息，内容更详细。

语法： -vb <level>

<Level> ：详细级别，值为{1, 2, 3} 默认值 vb=1

示例

./STM32_Programmer.sh –c port=/dev/ttyS0 br=115200 –vb 3

此示例的结果如*图 99*中所示。

图99. 详细程度指令

```
$ ./STM32_Programmer.sh -c port=/dev/ttyS0 br=115200 -vb 3
 Serial Port /dev/ttyS0 is successfully opened.
   Port configuration: parity = none, baudrate = 115200, data-bit = 8,
                       stop-bit = 1.0, flow-control = off
  Sending init command:
  byte 0x7F sent successfully to target
  Received response from target: 0x79
  Activating device: OK
  Sending GetID command and its XOR:
  byte 0x02 sent successfully to target
  byte 0xFD sent successfully to target
  Received response from target: 0x79
  Received response from target: 0x01050079
  Chip ID: 0x500
  Sending Get command and its XOR:
  byte 0x00 sent successfully to target
  byte 0xFF sent successfully to target
  Received response from target: 0x79
  Received response from target: 0x07
  Received response from target: 0x07310001020311213179
  BootLoader version: 3.1
```

3.2.12 Log指令

-log, --log

说明：此可追溯性指令能够将全部流量（详细级别最高）存储到日志文件中。

语法： -log [filePath.log]

[filePath.log]	日志文件的路径，默认路径为 $HOME/.STM32CubeProgrammer/trace.log。

示例

./STM32_Programmer.sh –c port=/dev/ttyS0 br=115200 –log trace.log

此示例的结果如*图 100*中所示。

图100. Log指令

```
$ ./STM32_Programmer.sh -c port=/dev/ttyS0 br=115200 -log trace.log

Log output file:    trace.log
  Serial Port /dev/ttyS0 is successfully opened.
   Port configuration: parity = none, baudrate = 115200, data-bit = 8,
                       stop-bit = 1.0, flow-control = off
  Activating device: OK
  Chip ID: 0x500
  BootLoader version: 3.1
```

日志文件trace.log包含详细消息，如*图 101*所示。

图101. 日志文件内容

```
16:41:19:345
Log output file:    trace.log
16:41:19:368 Serial Port /dev/ttyS0 is successfully opened.
16:41:19:368  Port configuration: parity = none, baudrate = 115200, data-bit = 8,
                    stop-bit = 1.0, flow-control = off
16:41:19:368 Sending init command:
16:41:19:368 byte 0x7F sent successfully to target
16:41:19:369 Received response from target: 0x79
16:41:19:369 Activating device: OK
16:41:19:369 Sending GetID command and its XOR:
16:41:19:369 byte 0x02 sent successfully to target
16:41:19:369 byte 0xFD sent successfully to target
16:41:19:370 Received response from target: 0x79
16:41:19:370 Received response from target: 0x01050079
16:41:19:370 Chip ID: 0x500
16:41:19:370 Sending Get command and its XOR:
16:41:19:370 byte 0x00 sent successfully to target
16:41:19:370 byte 0xFF sent successfully to target
16:41:19:371 Received response from target: 0x79
16:41:19:371 Received response from target: 0x07
16:41:19:371 Received response from target: 0x07310001020311213179
16:41:19:371 BootLoader version: 3.1
```

3.2.13 外部加载指令

-el

说明：此指令允许输入一个或多个外部存储器加载程序的路径，以便使用外部存储器执行编程、写入、擦除和读取操作。

语法：**-el [externalLoaderFilePath1.stldr]** 外部加载程序文件的绝对路径。

-el [externalLoaderFilePath1.stldr]… -el [externalLoaderFilePath10.stldr] 外部加载程序文件的绝对路径。

示例 1：

./STM32_Programmer.sh -c port=swd -w “file.bin” 0x90000000 –v –el “/local/user/externalLoaderPath.stldr”

示例2：

./STM32_Programmer.sh -c port=swd –e all –el “/local/user/externalLoaderPath.stldr”

示例 3：

./STM32_Programmer.sh -c port=swd -w “file.bin” 0x90000000 –v –el “/local/user/externalLoaderPath1.stldr” “/local/user/externalLoaderPath2.stldr”

注： *仅有SWD/JTAG端口支持该指令。*

注： *最多可使用十个外部加载程序。*

3.2.14 使用引导加载程序接口的外部加载程序指令

-elbl

说明：此指令可以为用于执行编程、写入、擦除和读取操作的外部存储器加载程序的路径提供使用引导加载程序接口的外部存储器（仅在RSS/RSSe背景下）。此指令只在执行SFIx过程时使用。

语法：**-elbl [externalLoaderFilePath.stldr]** 外部加载程序文件的绝对路径。

示例 1：

>STM32_Programmer_CLI.exe -c port=usb1 -elbl MX25LM51245G_STM32L552E-EVAL-SFIX-BL.stldr -sfi out.sfix hsm=0 license.bin -rsse RSSe\L5\enc_signed_RSSe_sfi_jtag.bin

注： *仅引导加载程序接口（UART/I2C/SPI/USB）支持该指令。*

SFIx的外部加载程序

SFIx操作的外部加载程序与RSSe_SFI_CallNsFunction一致，因此，外部加载程序中使用的所有函数的签名都必须与该函数相同。

rsse_sfi_ns_call_t

rsse_sfi_ns_call_t description in C coding language :

```
typedef uint32_t (*rsse_sfi_ns_call_t)(void * input_param);
```

这样一来，外部加载程序中这些函数的实现必须稍作修改才能与输入参数同步。

修改后扇区擦除函数的示例：

```
KeepInCompilation int SectorErase (uint32_t *params)
{
  int result = 0;
  uint32_t BlockAddr;
  uint32_t EraseStartAddress = params[0];
  uint32_t EraseEndAddress = params[1];
```

3.2.15 解除读保护指令

-rdu, --readunprotect

说明：此指令通过将RDP级别从级别1更改为级别0来删除存储器读保护。

语法： --readunprotect

示例

./STM32_Programmer.sh –c port=swd –rdu

3.2.16 TZ降级指令

-tzenreg, --tzenregression

说明：此指令通过禁用TZEN（从1变为0）来解除TrustZone保护。

语法：--tzenregression

示例

./STM32_Programmer.sh –c port=usb1 –tzenreg

注： *仅引导加载程序接口和具有可信区的MCU支持此指令。*

3.2.17 选项字节指令

-ob, --optionbytes

说明：该指令允许用户通过显示或修改设备的选项字节对其加以控制。

语法： -ob [displ] / -ob [OptByte=<value>]

[displ]： 用于显示选项字节的完整集合。

[OptByte=<value>]： 用于给定选项字节的编程。

示例

./STM32_Programmer.sh –c port=swd –ob rdp=0x0 –ob displ

注： 关于器件选项字节的更多信息，请参考ST网站www.st.com上提供的编程手册和参考手册中的专门章节。

3.2.18 Safety lib指令

-sl, --safelib

说明：该指令允许通过添加加载区（段）来修改固件文件，加载区（段）中包含有用户程序计算出的CRC值。

支持格式：bin、elf、hex和Srec。

语法：-sl <file_path> <start_address> <end_address> <slice_size>

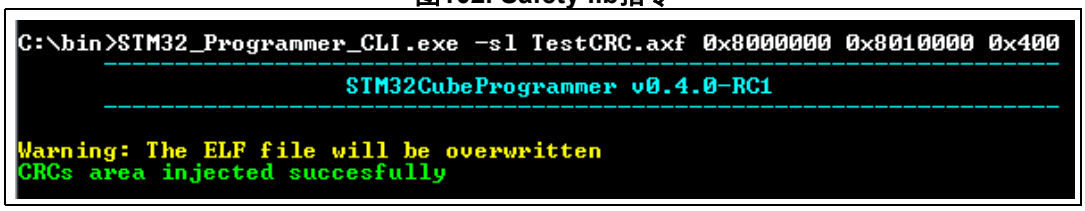
<file_path>	文件路径（bin、elf、hex或Srec）
<start_address>	Flash存储器起始地址
<end_address>	Flash存储器结束地址
<slice_size>	每个CRC值的大小

示例

STM32_Programmer_CLI.exe –sl TestCRC.axf 0x8000000 0x8010000 0x400

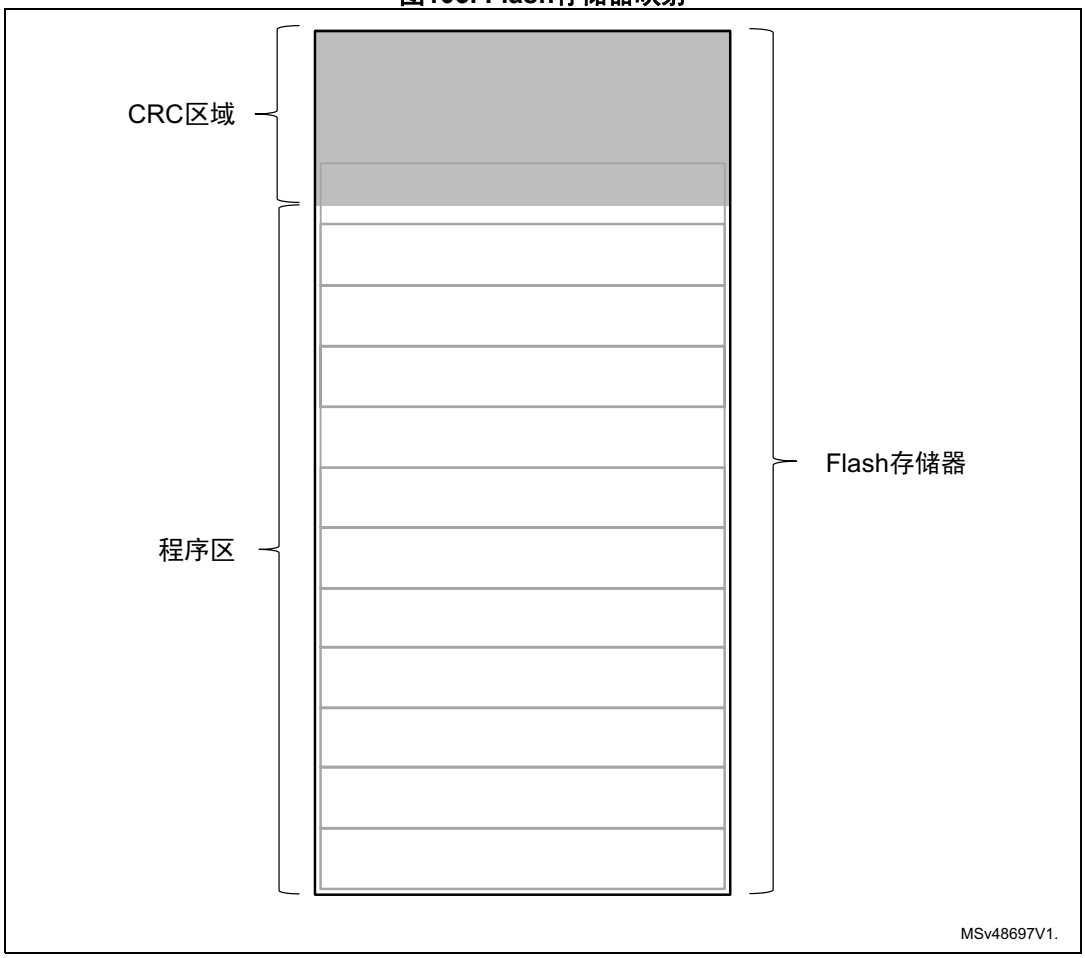
结果示于*图 102*中。

图102. Safety lib指令



程序Flash存储器分为多个片段，其大小作为safety lib指令的参数给出，如上面的示例所示。对于每个片段，分别计算CRC值并将其置于CRC区域中。CRC区位于存储器末尾，如*图 103*所示。

图103. Flash存储器映射



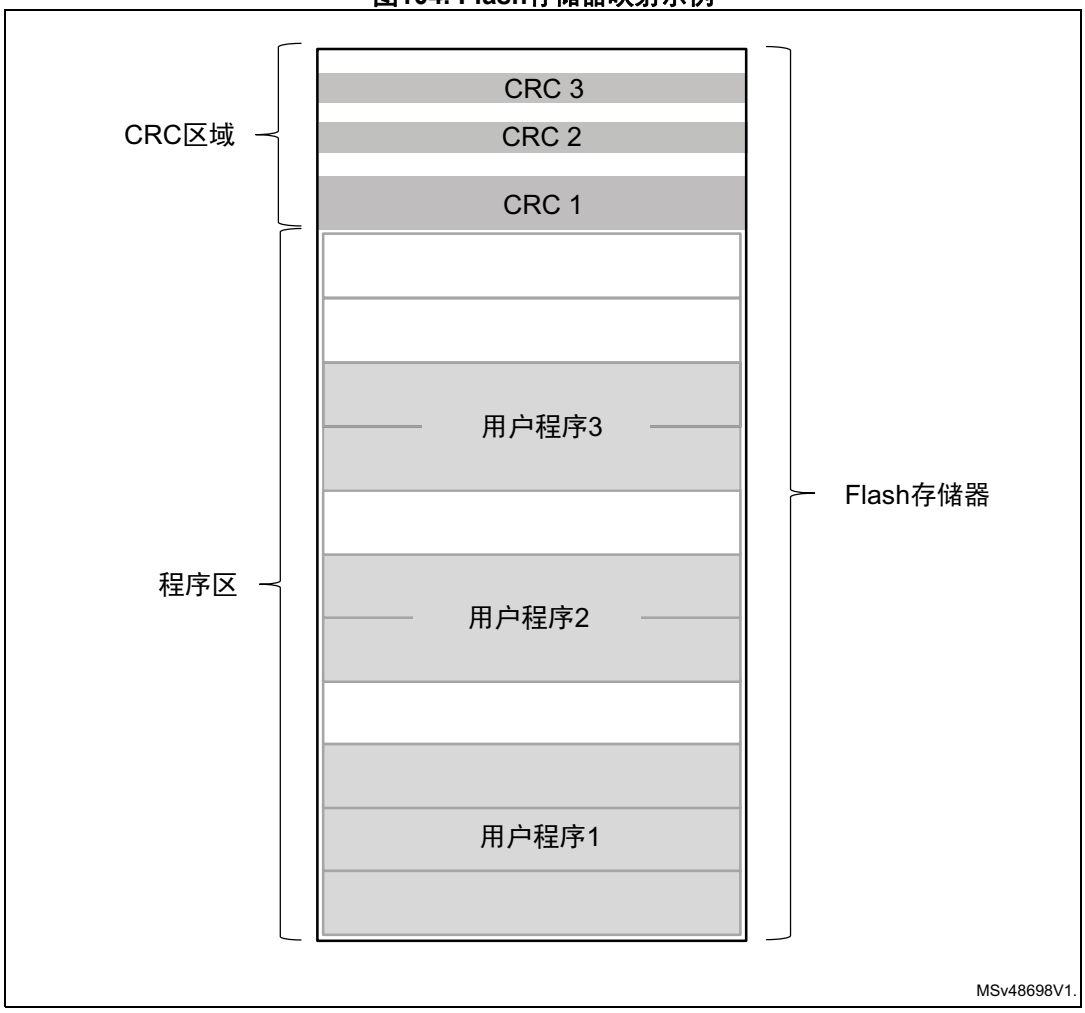
CRC区域的地址和大小确定如下：

CRCs_Area_Size = Flash_Size / Slice_Size * 4 字节

CRCs_Start_Address = Flash_End_Address - CRCs_Area_Size

CRC区域中的CRC值根据Flash存储器中用户程序的位置放置，参见*图 104*。

图104. Flash存储器映射示例



CRC区域内的CRC范围地址计算如下：

$$@ = \text{CRCs\_Start\_Address} + \left(\frac{\text{UserProg\_Start\_Address} - \text{Flash\_Start\_Address}}{\text{Slice\_Size}} \cdot 4 \text{ bytes} \right)$$

3.2.19 SFI特定的指令的安全编程

安全固件安装（SFI）特性支持安全固件刷写（在一些STM32器件上可用）。固件提供方能够保护其内部固件免受任何非法访问，并控制可编程的器件数量。

可使用不同通信信道（如JTAG/SWD或引导加载程序接口（UART、SPI和USB））执行受保护的固件安装。

请参考 *使用STM32CubeProgrammer进行安全编程*（AN5054）了解详细信息，该应用笔记可从ST网站*www.st.com*获取。

-sfi, --sfi

说明：sfi文件编程

语法：-sfi [<protocol=Ptype>] <.sfi file_path> [hsm=0|1] <lic_path|slot=slotID> [<licMod_path>|slot=slotID]

[<protocol=Ptype>]	要使用的协议类型：静态/实时（目前仅支持静态协议），默认为静态。
<file_path>	要编程的sfi文件路径。
[hsm=0|1]	为HSM设置用户选项，使用值 {0（不使用HSM），1（使用HSM）}，默认值：hsm=0。
<lic_path|slot=slotID>	SFI许可文件路径（若hsm=0）或 读写器插槽ID（若使用HSM（hsm=1））。
[<licMod_path>|slot=slotID]	若不使用HSM（hsm=0），为集成SMI许可文件路径的列表；若使用HSM（hsm=1），则为读写器插槽ID。仅在组合情况下使用，SFI文件中的列表顺序必须与模块集成顺序相对应。

-rsse, --rsse

说明：该指令用于选择根安全服务扩展库（RSSe）。对于使用RSSe进行安全固件安装（SFI）的器件，为强制命令。RSSe二进制文件可在STM32CubeProgrammer/bin/RSSe文件夹中找到。

语法：-rsse <file_path>

<file_path> RSSe文件的路径

-a, --abort

说明：该指令用于清理未正常完成的过程。目前正在进行的操作停止且系统回到空闲状态。

语法：-a

3.2.20 SFIx特定的指令的安全编程

安全固件安装（SFIx）特性支持安全的外部固件刷写（在一些具有OTFDEC功能的STM32器件上可用）。固件提供方能够保护其外部固件/数据免受任何非法访问，并控制可编程的器件数量。

SFIx安全编程只能通过JTAG/SWD接口来执行。

请参考 *使用STM32CubeProgrammer进行安全编程*（AN5054）了解详细信息。

-sfi, --sfi

说明：sfix文件编程

语法：-sfi [<protocol=Ptype>] <.sfix file_path> [hsm=0|1] <lic_path|slot=slotID> [<licMod_path>|slot=slotID]

[<protocol=Ptype>]	要使用的协议类型：静态/实时（目前仅支持静态协议），默认为静态。
<file_path>	要编程的sfi文件路径。
[hsm=0|1]	为HSM设置用户选项，使用值 {0（不使用HSM），1（使用HSM）}，默认值：hsm=0。
<lic_path|slot=slotID>	SFI许可文件路径（若hsm=0）或 读写器插槽ID（若使用HSM（hsm=1））。
[<licMod_path>|slot=slotID]	若不使用HSM（hsm=0），为集成SMI许可文件路径的列表；若使用HSM（hsm=1），则为读写器插槽ID。仅在组合情况下使用，SFI文件中的列表顺序必须与模块集成顺序相对应。

-elbl --extload	选择自定义外部存储器加载程序，仅适用于JTAG/SWD接口
<file_path>	外部存储器加载程序文件路径
-elbl --extloadbl	为引导加载程序接口选择自定义外部存储器加载程序
<file_path>	外部存储器加载程序文件路径

-rsse, --rsse

说明：该指令用于选择根安全服务扩展库（RSSe）。对于使用RSSe进行安全固件安装（SFI）的器件，为强制命令。RSSe二进制文件可在STM32CubeProgrammer bin/RSSe文件夹中找到。

语法：-rsse <file_path>

<file_path> RSSe文件的路径

-a, --abort

说明：该指令用于清理未正常完成的过程。正在进行的操作停止且系统回到空闲状态。

语法：-a

注： *由于RSS已经执行了一些初始化，因此对于SFIx用例而言ExternalLoader是不同的，其External FlashLoader名称的末尾标记有–SFIX。*

3.2.21 HSM相关的指令

为了控制可编程的器件数量，ST提供了基于HSM（硬件安全模块）的安全固件刷写服务作为要在编程室部署的许可生成工具。

有两个HSM版本可用：

- HSMv1：静态HSM，用于生成固件许可，以便进行事先选定器件的STM32安全编程。
- HSMv2：动态HSM，是上一个版本的更新版本，可面向器件的STM32安全编程生成固件许可，这些器件是在OEM现场通过个性化数据选定的。

在使用HSM之前，必须使用Trusted Package Creator进行编程，此工具可通过一些特定的输入配置对两个版本进行编程，详情见UM2238。

请参考 *使用STM32CubeProgrammer进行安全编程*（AN5054）了解详细信息。

-hsmgetinfo

说明：读取HSM可用信息

语法：-hsmgetinfo [slot=<SlotID>]

[slot=<SlotID>] 智能卡读卡器的插槽ID
默认值：slot=1（PC集成式智能卡读卡器）

-hsmgetcounter

说明：读取许可计数器的当前值

：-hsmgetcounter [slot=<SlotID>]

[slot=<SlotID>] 智能卡读卡器的插槽ID
默认值：slot=1（PC集成式智能卡读卡器）

-hsmgetfwid

说明：读取固件/模块标识符

语法：-hsmgetfwid [slot=<SlotID>]

[slot=<SlotID>] 智能卡读卡器的插槽ID
默认值：slot=1（PC集成式智能卡读卡器）

-hsmgetstatus

说明：读取卡的当前生命周期状态

语法：-hsmgetstatus [slot=<SlotID>]

[slot=<SlotID>]	智能卡读卡器的插槽ID 默认值：slot=1（PC集成式智能卡读卡器）

-hsmgetlicense

说明：如果计数器不为null，则获取当前芯片的许可

语法：-hsmgetlicense <file_path> [slot=<SlotID>] [protocol=<Ptype>]

<file_path>	存储所接收许可的文件路径
[slot=<SlotID>]	智能卡读卡器的插槽ID 默认值：slot=1（PC集成式智能卡读卡器）
[<protocol=Ptype>]	要使用的协议类型：静态/实时 目前只支持静态协议 默认值：静态

-hsmgetlicensefromcertifbin, -hsmglfcb

说明：如果计数器不为null，则获取当前证书二进制文件的许可。

语法：-hsmglfcb <certif_file_path.bin> <license_file_path.bin> [slot=<SlotID>] [protocol=<Ptype>]

<certif_file_path.bin>	从中读取了输入证书的文件路径。
<license_file_path.bin>	存储所接收许可的文件路径
[slot=<SlotID>]	智能卡读卡器的插槽ID。 默认值：slot=1（PC集成式智能卡读卡器）

3.2.22 STM32WB特定的指令

-antirollback

说明：执行防回滚操作

语法：-antirollback

-startfus

说明：启动FUS

语法：-startfus

-getuid64

说明：读取器件唯一标识符（UID）

语法：-getuid64

-fusgetstate

说明：读取FUS状态

语法：-fusgetstate

-fusopgetversion

说明：读取FUS Operator版本

语法：-fusgetversion

注： *FUS Operator版本不能通过引导加载程序接口获得。*

-fwdelete

说明：删除BLE栈固件

语法：-fwdelete

-fwupgrade

说明：升级BLE栈固件或FUS固件。

语法：-fwupgrade <file_path> <address> [firstinstall=0|1] [startstack=0|1] [-v]

<file_path>	新固件映像文件路径
<address>	下载的起始地址
[firstinstall=0|1]	如果是首次安装，则为1，否则为0 可选，默认值为**firstinstall=0**
[-v]	在开始升级前确认下载操作是否成功完成

-startwirelessstack

说明：启动无线栈

语法：-startwirelessstack

-authkeyupdate

说明：验证密钥更新

语法：-authkeyupdate <file_path>

<file_path>	验证密钥文件路径。 这是在使用**-sign**指令进行固件签名时STM32TrustedPackageCreator生成的公钥。

-authkeylock

说明：验证密钥锁定

锁定后，不能再用**-authkeyupdate**指令对其进行修改

语法：-authkeylock

-wusrkey

关于客户密钥存储的更多信息，请参考已经提到的AN5185。

语法：-wusrkey <file_path> <keytype=1|2|3>

<file.path>：二进制格式的客户密钥

<keytype=1|2|3>：用户密钥类型值：1（简单）、2（主密钥）或3（加密）

-startwirelessstack

说明：启动无线栈

语法：-startwirelessstack

注： *只有在使用SWD、USB DFU和UART接口时这些指令才可用。*

注： *必须是正在复位模式。*

SWD接口的用法示例

- FUS升级：
  STM32_Programmer_CLI.exe -c port=swd mode=UR -ob nSWboot0=0 nboot1=1 nboot0=1 -fwupgrade stm32wb5x_FUS_fw.bin 0x080EC000 firstinstall=1
- 栈安装：
  STM32_Programmer_CLI.exe -c port=swd mode=UR -ob nSWboot0=0 nboot1=1 nboot0=1 -fwupgrade stm32wb5x_BLE_Stack_fw.bin 0x080EC000
- 用户应用安装：
  STM32_Programmer_CLI.exe -c port=swd mode=UR -d UserApplication.bin 0x08000000 -v

注： *自FUS v1.2.0起提供-antirollback指令。*

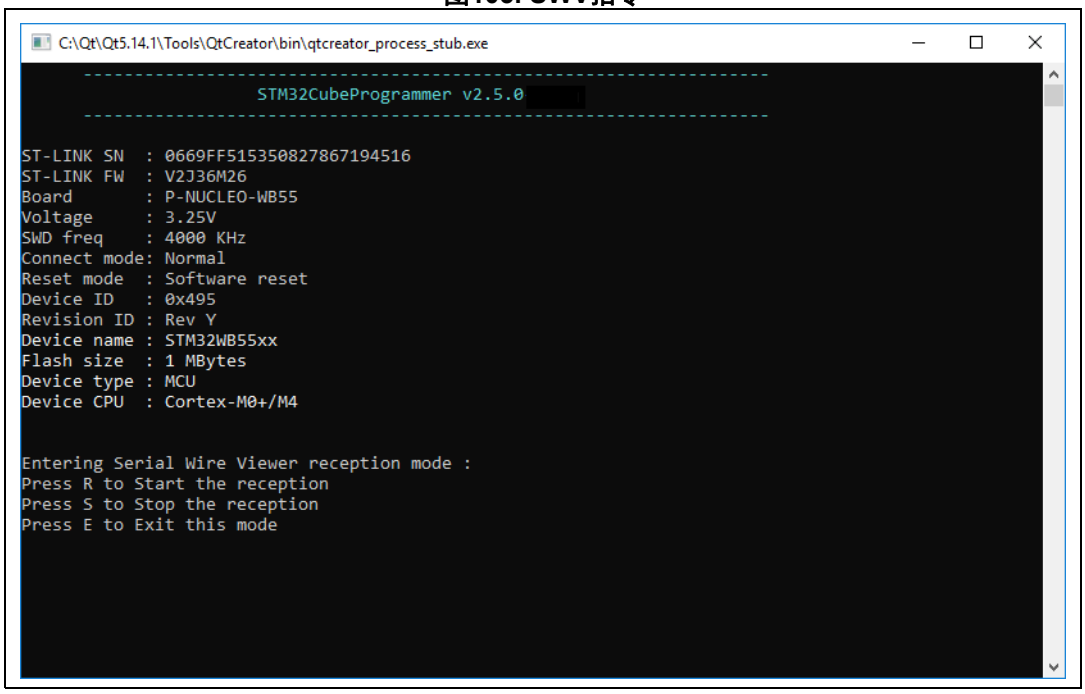
3.2.23 串行线查看器（SWV）指令

-SWV

说明：此指令用于访问串行线查看器控制台模式，以便显示通过SWO从目标发送的printf数据。

在该模式下（参见*图 105*），用户分别可以通过按下键盘上的“R”和“S”按钮开始和停止SWO数据接收。接收的SWO数据显示在控制台上。用户可通过按下“E”按钮退出串行线查看器控制台模式并终止接收会话。

图105. SWV指令



语法： swv <freq=<frequency>> <portnumber=0-32> [<file_Path.log>]

<freq=<frequency>>	以MHz为单位的系统时钟频率。
<portnumber=0-31|all>	ITM端口编号，值：0-31，或“全部”（所有端口）。
[<file_Path.log>]	SWV日志文件路径（可选）。如果未指定，则默认为“$USER_HOME/STMicroelectronics/STM32Programmer/SWV_Log/swv.log”

示例：

STM32_Programmer_CLI.exe -c port=swd -swv freq=32 portnumber=0 C:\Users\ST\swvLog\example.log

注： *只有使用SWD接口时串行线查看器才可用。*

注： *由于ST-LINK硬件缓冲区的容量有限，传输期间可能会丢失一些SWV字节。*

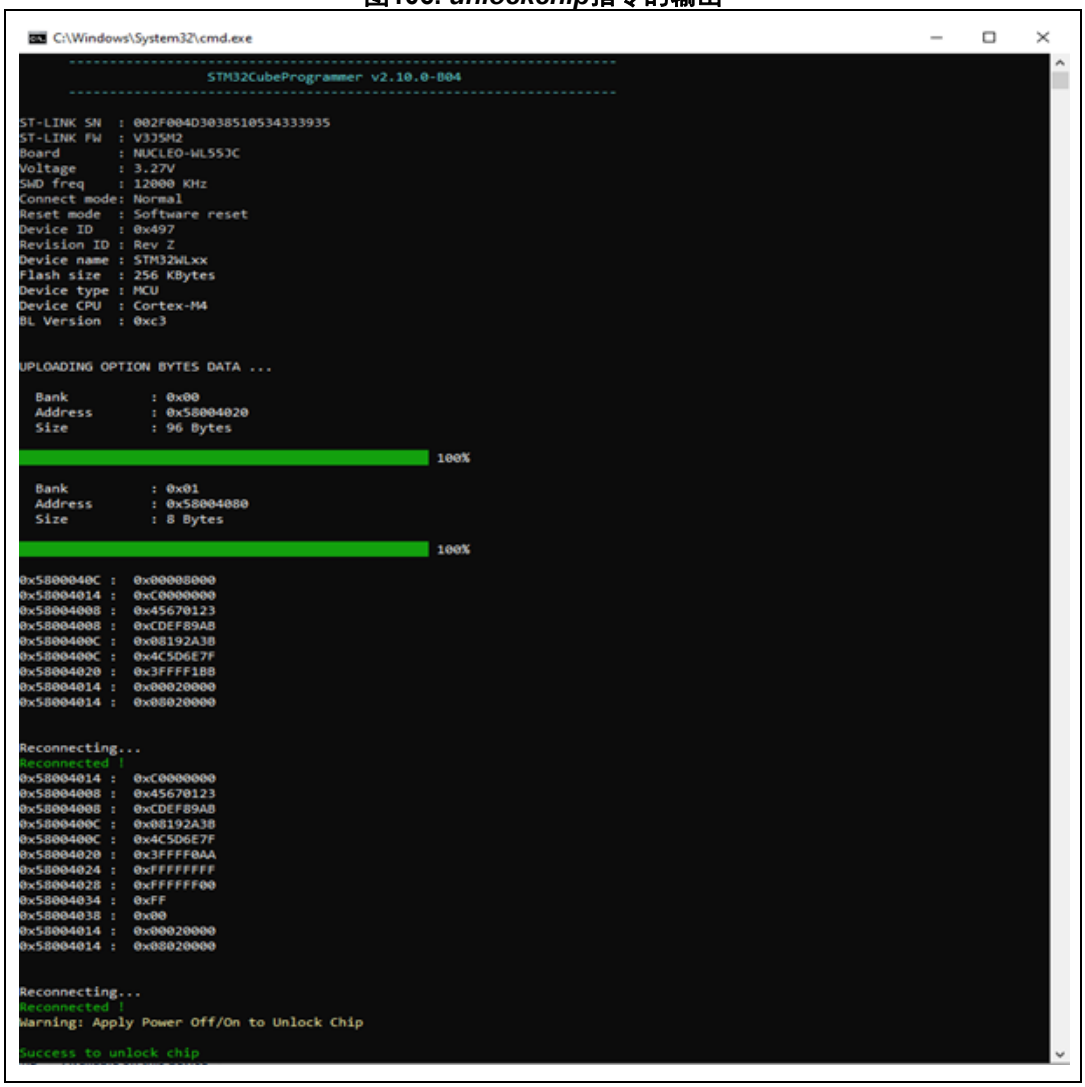
3.2.24 STM32WL的特定指令

在执行加密固件安装之前，将器件设置为默认状态，即安全性禁用（ESE＝0x0）且所有选项字节均为默认值。

为此，STM32CubeProgrammer允许用户使用两个命令行执行这些步骤：

1. **desurity**：用于禁用安全特性。
   示例：STM32_Programmer_CLI.exe -c port=swd mode=hotplug -dsecurity
2. **说明**：该指令用于将选项字节配置为其默认值。
   示例：STM32_Programmer_CLI.exe -c port=swd mode=hotplug -setdefaultob
3. **-ob unlockchip**：如果设定了不良选项字节，则用户可使用该指令解锁器件。
   示例：STM32_Programmer_CLI.exe -c port=swd –ob unlockchip

图106. *unlockchip*指令的输出



注： *解锁芯片指令只对STLink连接可用。*

在执行这些指令后执行重启。当不能使用常用方法修改选项字节时，用户可使用这两个指令解锁板件。

*图 107*和*图 108*显示了这些命令行的结果。

图107. 禁用安全特性

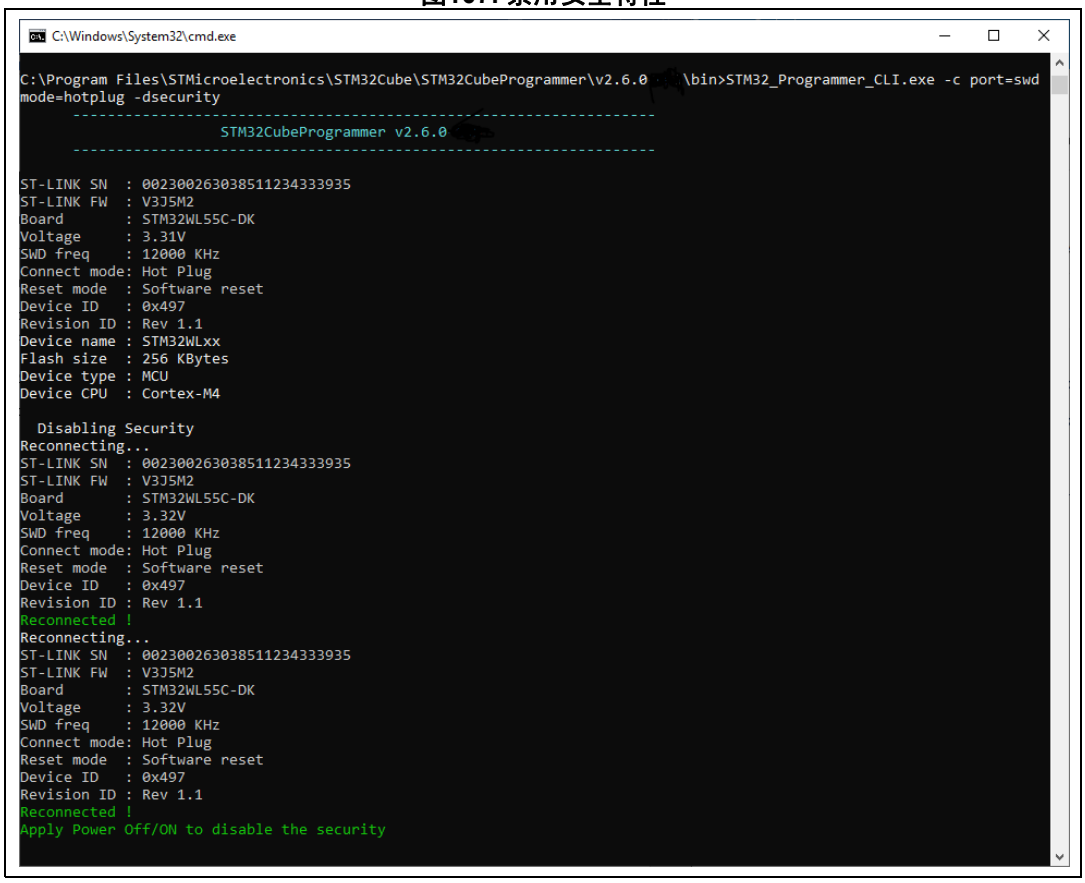
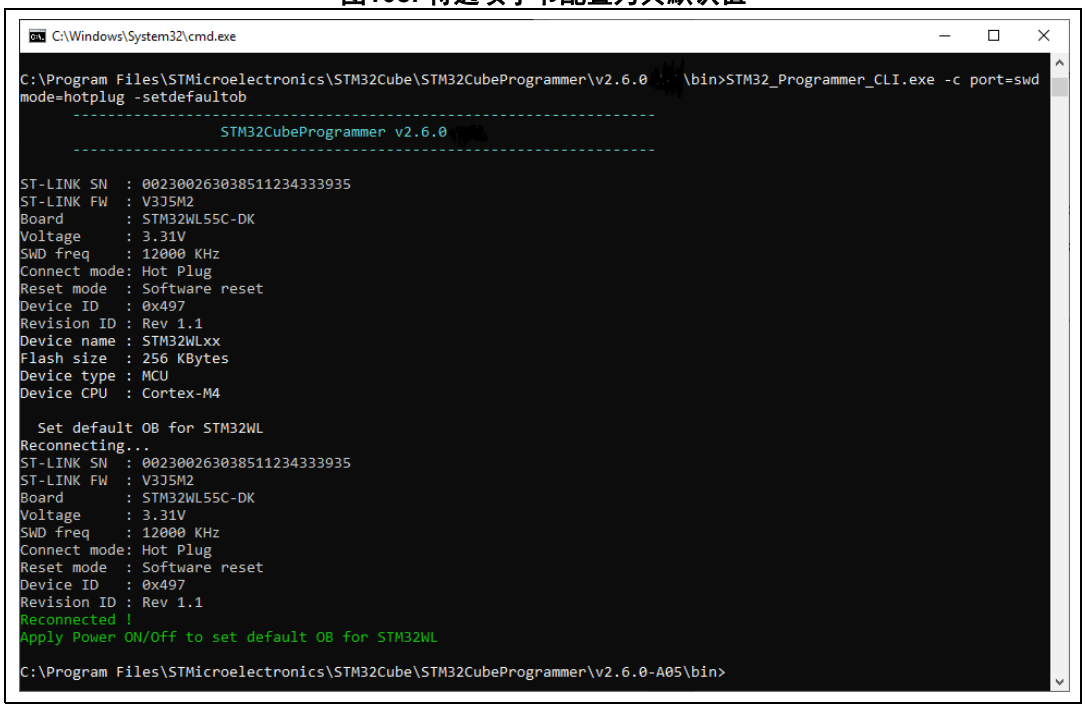


图108. 将选项字节配置为其默认值



如果用户锁定板件且无法用这两个指令解锁板件，则有专门的脚本可以用来解锁。这些脚本位于“../bin/STM32WLScripts”目录下，其中包含一个命令行，该命令行使用–wdbg选项在OPTR寄存器中直接写入脚本。

STM32Scripts文件夹包含两个文件和Readme.txt：

1. “SetRDPLevelCM0.bat”用于通过Cortex M0+解锁板件
2. “SetRDPLevelCM4.bat”用于通过Cortex M4解锁板件

注： *如果SFI指令执行失败，则必须使用禁用安全特性命令行（-dsecurity）将STM32WL芯片设置为默认状态，然后设置默认选项字节命令行（-setdefaultob）。*

3.2.25 SigFox凭证指令

仅STM32WL器件支持这些指令。

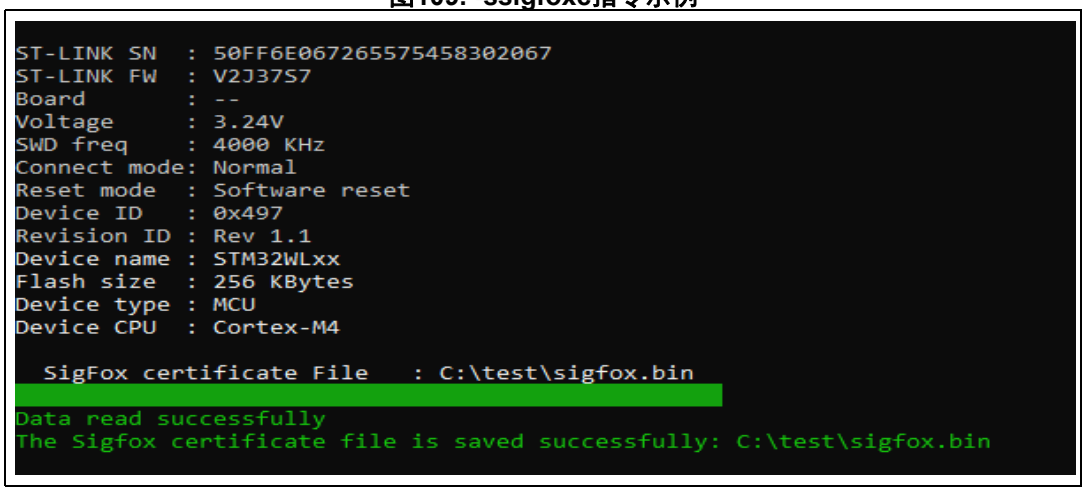
-ssigfoxc

说明：该指令用于将芯片证书保存为二进制文件。

语法： -ssigfoxc <binary_file_path>

示例：STM32_Programmer_CLI.exe -c port=swd -ssigfoxc “/local/user/chip_certif.bin”

图109. -ssigfoxc指令示例



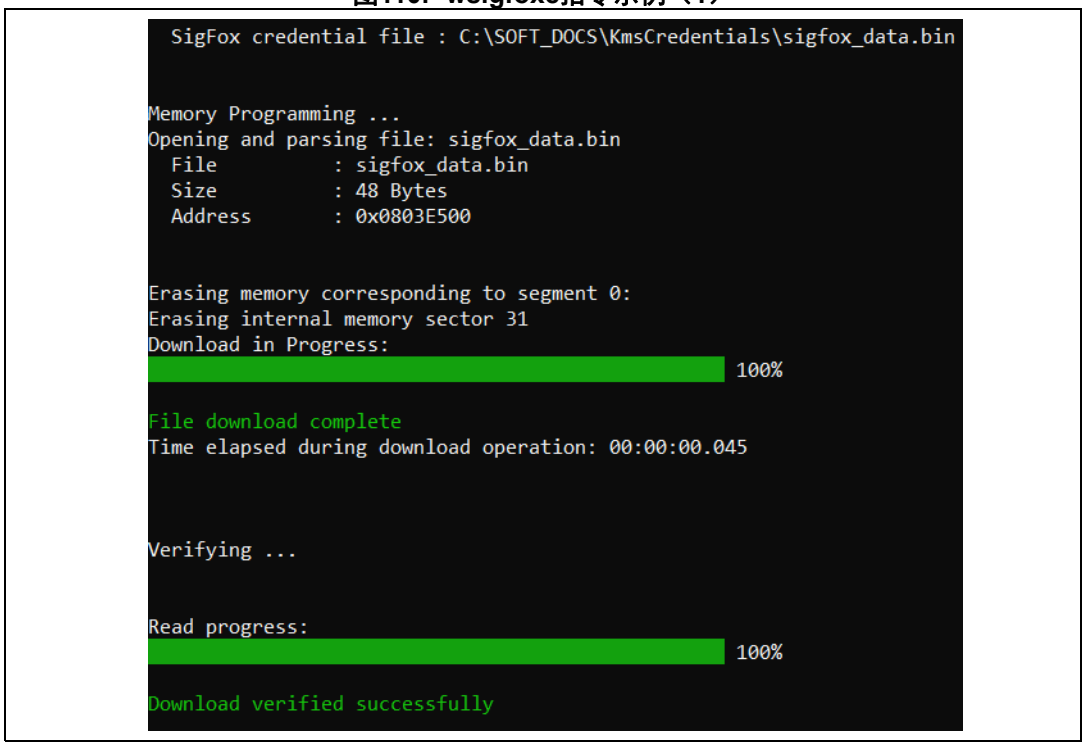
-wsigfoxc

说明：该指令用于在地址0x0803E500写入芯片证书

语法：-wsigfoxc <binary_file_path> <address> [地址为可选项，默认为0x0803E500]

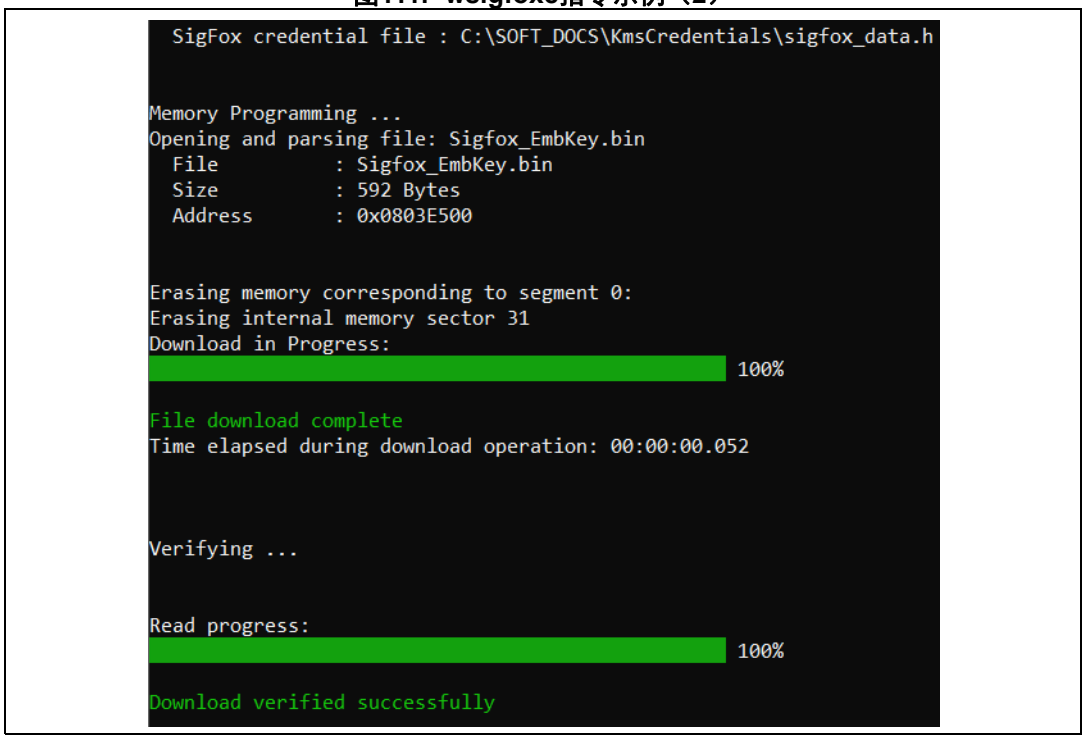
示例1：STM32_Programmer_CLI.exe -c port=swd -wsigfoxc “/local/user/sigfox_data.bin”0x0803E500

图110. -wsigfoxc指令示例（1）



示例2：STM32_Programmer_CLI.exe -c port=swd -wsigfoxc “/local/user/sigfox_data.h”

图111. -wsigfoxc指令示例（2）



3.2.26 寄存器查看器

-regdump

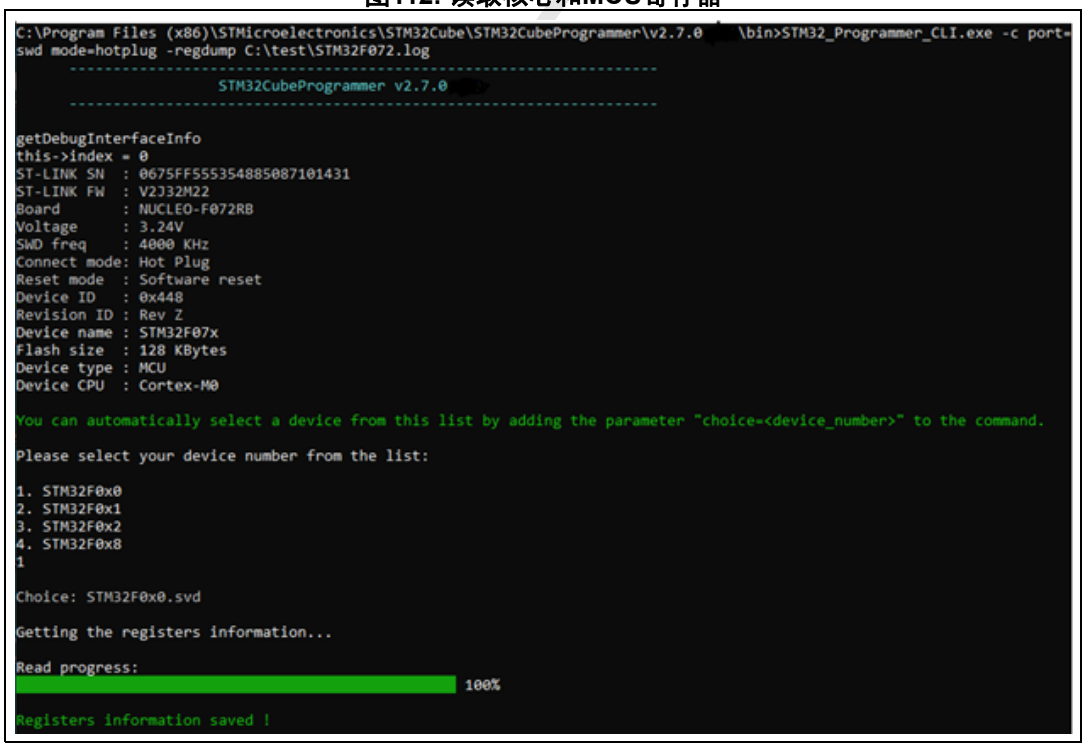
说明：读取并转存内核和MCU寄存器

语法：-regdump <file_path.log> [choice=<number>]

<file_path.log>	日志文件路径
[choice=<number>]	兼容器件列表中的器件编号（可选）。 如果在未带可选参数的情况下执行该指令，则将显示该列表。

示例：STM32_Programmer_CLI.exe -regdump C:\test\STM32F072.log

图112. 读取核心和MCU寄存器



3.2.27 硬性故障分析器

使用特定命令行开始分析（参见*第 2.13 节*）。

语法：**-hf**

输出跟踪数据包含不同种类的重要信息，有助于更好地理解导致特定故障的原因。

显示描述性消息“STM32CubeProgrammer故障分析器”，表示检测流程已开始。

注： *在执行故障分析器指令之前，必须建立与目标之间的连接。*

示例

使用与GUI模式相同的示例（除零）。

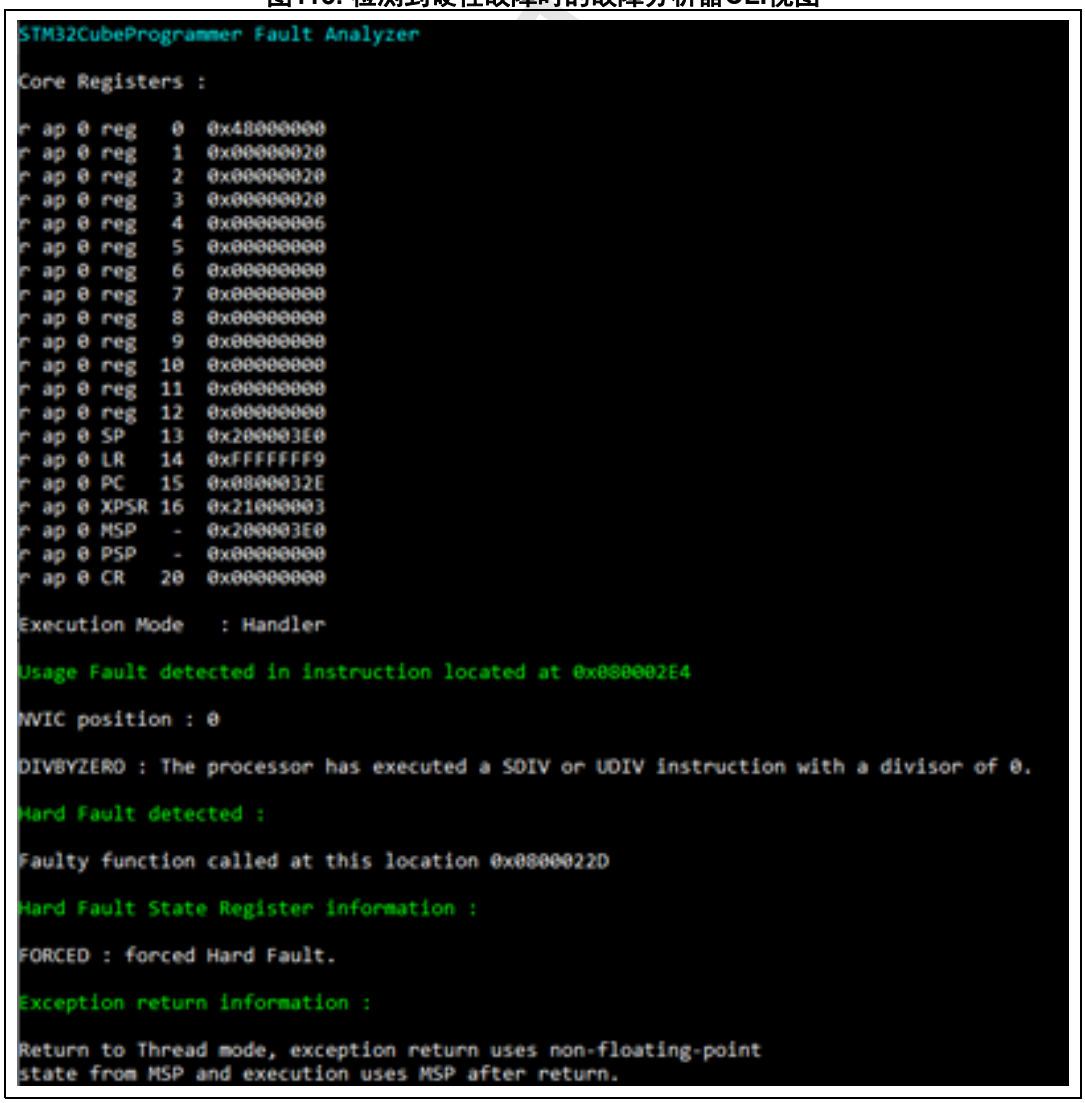
指令：**-c port=swd mode=hotplug -hf**

在命令行输出中，绿色消息显示“检测到硬性故障” 和“处理器执行了除零的SDIV或UDIV指令”。

可提取的有用信息：

- 故障指令地址：0x80002E4
- 位于此地址的函数调用了故障指令：0x800022D
- NVIC位置：0，窗口看门狗中断
- 执行模式：Handler
- 内核寄存器捕获。

图113. 检测到硬性故障时的故障分析器CLI视图



3.2.28 具有密码的RDP降级

一些STM32产品（如STM32U5系列）能够使用基于密码的可选RDP降级，包括RDP 2级。

参考手册中提供了关于此硬件机制的详细信息。

密码锁定和解锁的CLI指令为：

- lockRDP1

说明：允许用户用密码锁定自1级起的RDP降级。

语法：- lockRDP1 <Password first 32 bits> <Password second 32 bits>

示例：

STM32_Programmer_CLI -c port=swd mode=hotplug -lockRDP1 0x12345678 0xDEADBEEF

- lockRDP2

说明：此指令允许用户用密码锁定自2级起的RDP降级。

语法：- lockRDP2 <Password first 32 bits> <Password second 32 bits>

示例：

STM32_Programmer_CLI -c port=swd mode=hotplug – lockRDP2 0x12345678 0xDEADBEEF

- unlockRDP1

说明：此指令可用密码锁定自1级起的RDP降级。

语法：- unlockRDP1 <Password first 32 bits> <Password second 32 bits>

示例：

STM32_Programmer_CLI -c port=swd mode=hotplug -unlockRDP1 0x12345678 0xDEADBEEF

- unlockRDP2

说明：此指令允许用户用密码解锁自2级起的RDP降级。

语法：- unlockRDP2 <Password first 32 bits> <Password second 32 bits>

示例：

STM32_Programmer_CLI -c port=swd mode=hotplug – unlockRDP2 0x12345678 0xDEADBEEF

注： *由于列出的指令不包含RDP降级操作，在解锁RDP后，用户必须执行RDP降级。*

注： *为了用密码解除RDP降级，用户必须使用Lock指令和密码（值为0x00000000 0x00000000），如* ***STM32_Programmer_CLI -c port=swd mode=hotplug -lockRDP1 0x00000000 0x00000000***

3.2.29 GetCertif指令

-gc

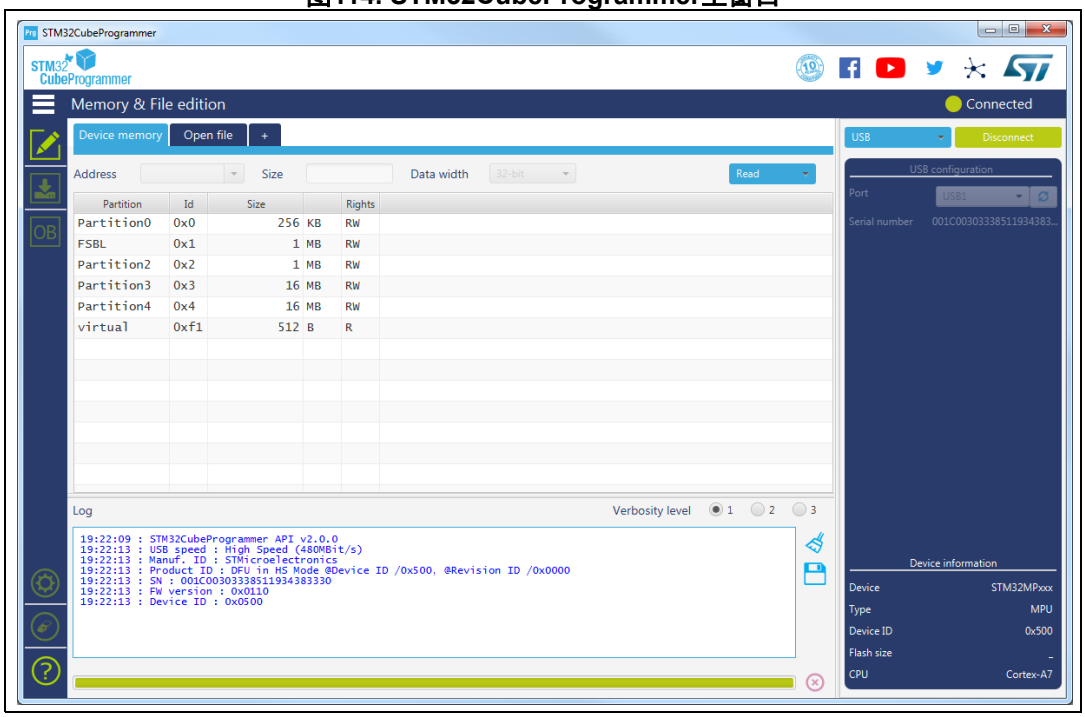
说明：该指令用于读取芯片证书。

语法：-gc certification.bin

4 MPU的STM32CubeProgrammer用户界面

4.1 主窗口

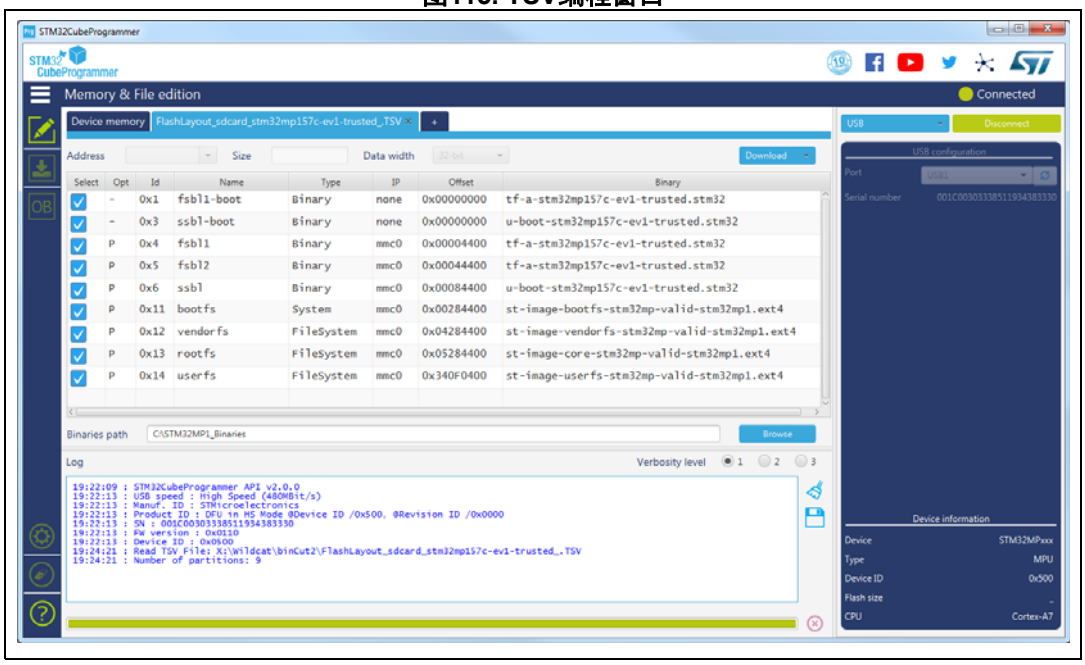
图114. STM32CubeProgrammer主窗口



在主窗口中，用户可以选择用于连接STM32MP1 BootROM的接口，可使用的接口为USB-DFU和UART（对于STM32MP1系列，不能通过STLink接口进行编程）。连接（使用连接按钮）后，窗口会显示可用的分区，用户可以打开TSV文件进行编程。

4.2 编程窗口

图115. TSV编程窗口



为了执行TSV文件编程，用户必须执行以下操作：

- 使用“打开文件”选项卡打开TSV文件，如果TSV文件格式正确，则在主窗口中显示TSV内容。STM32MP1Linux分发中提供了TSV文件，请参考STM32MP1wiki了解详细信息。
- 在“二进制文件路径”文本框中指定二进制文件路径。
- 在“选择”列中选中要编程的分区列表，默认选中所有分区。
- 使用“下载”按钮启动下载。

关于刷写操作的详细信息，请参考AN5275（可从ST网站*www.st.com*获取）。

5 MPU的STM32CubeProgrammer CLI

5.1 STM32MP1的可用指令

本节介绍STM32MP1器件支持的指令。

5.1.1 连接指令

-c, --connect

说明：建立到设备的连接。该指令允许主机打开选定的设备端口（UART/USB）

语法：-c port=<Portname> [noinit=<noinit_bit>] [br=<baudrate>] [P=<Parity>] [db=<data_bits>] [sb=<stop_bits>] [fc=<flowControl>]

port=<Portname>	接口标识符： – ex COMx（适用于Windows） – /dev/ttySx（适用于Linux） – usbx对应于USB接口
[noinit=<noinit_bit>]	设置No Init位，值为{0,1}，默认值为0。 如果先前的连接为激活状态（无需发送0X7F），则可以使用Noinit=1。
[br=<baudrate>]	波特率，如9600和115200，默认值为115200。
[P=<Parity>]	奇偶校验位，值为(EVEN, NONE, ODD)，默认为EVEN。
[db=<data_bits>]	数据位，值为(6, 7, 8)，默认值为8。
[sb=<stop_bits>]	停止位，值为(1, 1.5, 2)，默认值为1。
[fc=<flowControl>]	流控制，值为（关闭、软件、硬件）。STM32MP1系列尚不支持软件和硬件流控制，默认为关闭。

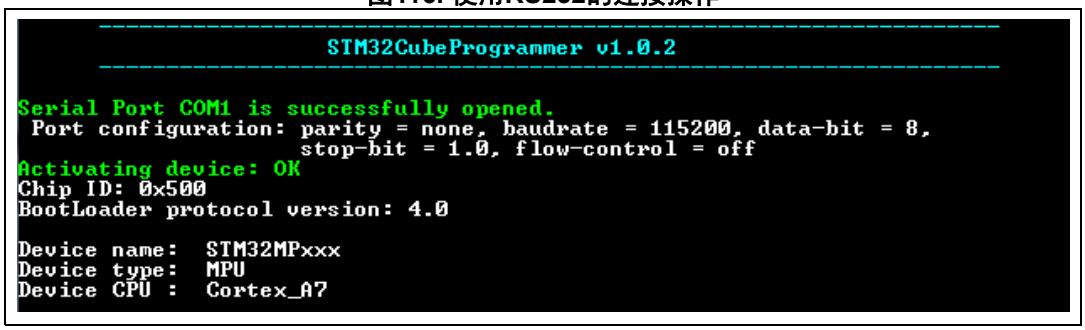
示例

使用 UART：

./STM32_Programmer.sh -c port=/dev/ttyS0 p=none

此示例的结果如*图 116*中所示。

图116. 使用RS232的连接操作



注： *当使用USB接口时，所有配置参数（如波特率、奇偶校验、数据位、频率和索引）都会被忽略。*

注： *为了使用UART接口进行连接，必须对端口配置（波特率、奇偶校验、数据位、停止位和流控制）进行有效组合。*

5.1.2 GetPhase指令

-p, --phaseID

说明：该指令用于获悉要执行的下一个分区ID。

语法：--phaseID

示例

./STM32_Programmer.sh –c port=/dev/ttyS0 p=none br=115200 --phaseID

5.1.3 下载指令

-w, --write, -d, --download

说明：将指定二进制文件中的内容下载到Flash或系统存储器中的特定分区。

语法：-w <file_path> [partitionID]

[file_path] 要下载的文件路径（bin、stm32、vfat、jffs2、ubi、ext2/3/4和img文件扩展名）。

[partition_ID] 要下载的分区ID。

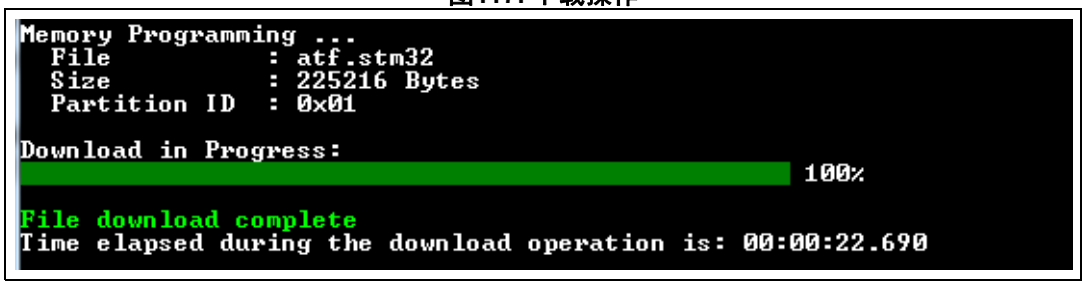
示例

./STM32_Programmer.sh -c port=/dev/ttyS0 p=none -d atf.stm32 0x01

此指令用于下载Atf分区（分区ID：0x01）的atf二进制文件。

此示例的结果如*图 117*中所示。

图117. 下载操作



注： *对于使用USB接口的U-boot，为了对非易失性存储器（简称NVM，具有用下载指令加载的分区）进行编程，用户必须用分区ID执行启动指令。此外，为了执行NVM中加载的应用，需要指定起始地址。*

示例：在候补地址0x1下载和显示

./STM32_Programmer.sh -c port=usb0 -w atf.stm32 0x1 –s 0x01

5.1.4 刷写服务

说明：嵌入式刷写服务旨在按顺序加载引导加载程序请求的分区。为此，STM32CubeProgrammer需要TSV文件，该文件中包含要加载的关于所请求分区的信息。

STM32CubeProgrammer执行下载并从请求的分区ID开始，直至操作结束（phaseID = 0xFE）。

语法：-w < tsv file_path >

<tsv file_path> 要下载的tsv文件的路径。

图118. TSV文件格式

#Opt	Id	Name	Type	IP	Offset	Binary
-	0x01	fsbl1-boot	Binary	none	0x0	tf-a-stm32mp157c-dk2-trusted.stm32
-	0x03	ssbl-boot	Binary	none	0x0	u-boot-stm32mp157c-dk2-trusted.stm32
P	0x04	fsbl1	Binary	mmc0	0x00004400	tf-a-stm32mp157c-dk2-trusted.stm32
P	0x05	fsbl2	Binary	mmc0	0x00044400	tf-a-stm32mp157c-dk2-trusted.stm32
P	0x06	ssbl	Binary	mmc0	0x00084400	u-boot-stm32mp157c-dk2-trusted.stm32
P	0x21	bootfs	System	mmc0	0x00284400	st-image-bootfs-openstlinux-weston-extra-stm32mp1.ext4
P	0x22	vendorfs	FileSystem	mmc0	0x04284400	st-image-vendorfs-openstlinux-weston-extra-stm32mp1.ext4
P	0x23	rootfs	FileSystem	mmc0	0x05284400	st-image-weston-openstlinux-weston-extra-stm32mp1.ext4
P	0x24	userfs	FileSystem	mmc0	0x340F0400	st-image-userfs-openstlinux-weston-extra-stm32mp1.ext4

示例

./STM32_Programmer.sh -c port=/dev/ttyS0 p=none br=115200 -d Flashlayout.tsv

注： *在对Flashlayout.tsv文件进行编程时，U-boot可能花费较长时间才能正确启动，为此，使用超时指令配置超时值（-tm <timeout>）。*

5.1.5 start指令

-g, --go, -s, --start

说明：该指令允许从指定地址开始执行设备存储器中的程序。

语法：--start [start_address/Partition_ID]

[start_address] 要执行的应用程序起始地址。如果未通过STM32MP和UART接口指定，则从上一次加载的分区开始。

[Partition_ID] 只有使用STM32MP器件时才需要此参数。它指定要开始的分区ID。

示例

./STM32_Programmer.sh --connect port=/dev/ttyS0 p=none br=115200 --start 0x03

该指令用于运行在分区0x03指定的代码。

注： *对于使用USB接口的U-boot，为了对NVM（具有用下载指令加载的分区）进行编程，用户需要用分区ID执行启动指令。为了执行NVM中加载的应用，需要指定起始地址。*

示例1：在候补地址0x1下载和显示

./STM32_Programmer.sh -c port=usb0 -w atf.stm32 0x01 –s 0x01

示例2：执行特定地址的代码

./STM32_Programmer.sh -c port=usb0 –s 0xC0000000

5.1.6 读取分区指令

-rp, --readPart

说明：读取指定的分区内容，并从偏移量地址开始将其上传到指定的二进制文件中。仅U-boot支持此指令。

语法： --readPart <partition_ID> [offset_address] <size> <file_path>

[partition_ID] 分区ID

[offset_address] 读取的偏移量地址

<size> 要读取的存储器内容大小

<file_path> 上载存储器内容的二进制文件路径

示例：

./STM32_Programmer.sh -c port=/dev/ttyS0 p=none br=115200 -rp 0x01 0x200 0x1000 readPart1.bin

用户可使用此指令从sebl1分区（位于偏移量地址0x200）读取0x1000字节，并将其内容上传到二进制文件“readPart1.bin”

5.1.7 List指令

-l, -list

说明：此指令可列出所有可用的UART和USB通信接口。

语法：-l, --list <interface_name>

<uart/usb> ：UART或USB接口

示例：
./STM32_Programmer.sh –list uart

5.1.8 QuietMode指令

-q, --quietMode

说明：该指令在下载指令和读取分区指令执行期间禁用进度条显示。

语法：-q, --quietMode

示例：
./STM32_Programmer.sh –c port=/dev/ttyS0 p=none br=115200 --quietMode –w binaryPath.bin 0x01

5.1.9 Verbosity指令

-vb, --verbosity

说明：此指令可以显示更多消息，内容更详细。

语法：-vb <level>

<level>：详细级别，值为{1, 2, 3} 默认值 vb=1

示例：
./STM32_Programmer.sh –c port=/dev/ttyS0 p=none br=115200 –vb 3

5.1.10 Log指令

-log, --log

说明：此可追溯性指令用于将全部流量（详细级别最高）存储到日志文件中。

语法：-log [filePath.log]

[filePath.log]：日志文件的路径（默认路径为$HOME/.STM32CubeProgrammer/trace.log）

示例：
./STM32_Programmer.sh –c port=/dev/ttyS0 p=none br=115200 –log trace.log

此指令生成日志文件“trace.log”，其中包含详细消息（参见*图 119*中的示例）。

图119. 日志文件内容

```
16:41:19:345
Log output file:   trace.log
16:41:19:368 Serial Port /dev/ttyS0 is successfully opened.
16:41:19:368  Port configuration: parity = none, baudrate = 115200, data-bit = 8,
              stop-bit = 1.0, flow-control = off
16:41:19:368 Sending init command:
16:41:19:368 byte 0x7F sent successfully to target
16:41:19:369 Received response from target: 0x79
16:41:19:369 Activating device: OK
16:41:19:369 Sending GetID command and its XOR:
16:41:19:369 byte 0x02 sent successfully to target
16:41:19:369 byte 0xFD sent successfully to target
16:41:19:370 Received response from target: 0x79
16:41:19:370 Received response from target: 0x01050079
16:41:19:370 Chip ID: 0x500
16:41:19:370 Sending Get command and its XOR:
16:41:19:370 byte 0x00 sent successfully to target
16:41:19:370 byte 0xFF sent successfully to target
16:41:19:371 Received response from target: 0x79
16:41:19:371 Received response from target: 0x07
16:41:19:371 Received response from target: 0x07310001020311213179
16:41:19:371 BootLoader version: 3.1
```

5.1.11 OTP编程

说明：这些指令用于从主机进行OTP编程。下文介绍了OTP编程指令功能，如下载或上传完整OTP映像和修改OTP值或属性。

注： *JTAG/SWD 调试端口连接模式不支持下列指令。*

- 将影子寄存器值加载到工具：
  为了执行加载操作，主机请求OTP分区数据，平台用以下网址描述的结构回复：https://wiki.st.com/stm32mpu/index.php/STM32CubeProgrammer_OTP_management。
- 将修改过的影子寄存器写入目标：
  通过执行以下流程执行此操作：
  a) 用户输入每个选定OTP影子寄存器的值和状态。
  b) 工具用新给出的OTP影子寄存器值和状态更新OTP结构。
  c) 工具继续发送更新过的结构，该结构中的“写入/读取配置”字段的bit0置为0（“写入/读取配置”是OTP结构中的#7字）。
  d) 在发送结构后，将重载影子寄存器值以便更新工具中的OTP结构。
- 用修改过的影子寄存器值进行OTP编程：
  在用户更新OTP值且OTP结构被刷新后，主机发送OTP结构，该结构中的“写入/读取配置”字段（OTP结构中的#7字）的bit0置为1。
- 将OTP值重载到影子寄存器：
  在OTP字成功编程后，主机上传OTP结构，以便更新OTP影子寄存器。此操作使主机能够通过“状态”字段中的bit4确认上一次SAFMEM编程的状态。
- BSEC控制寄存器编程：
  在用户更新给定BSEC控制寄存器的值后（配置、调试配置、特性配置和通用锁定配置），主机更新OTP结构并将其发送给器件（“写入/读取配置”字段的bit0置为0）。
- OTP编程CLI：
  为用户提供一组指令，用于在OTP分区执行选定操作流程。下面将对这些指令中的每一个进行说明。

5.1.12 SAFMEM指令编程

说明：此指令允许用户通过修改OTP字进行SAFMEM存储器编程。

语法：-otp program [wordID=(value)] [value=(value)] [sha_rsl=(value)] [sha_wsl=(value)] [sl=(value)] [pl=(value)]

[wordID=(value)]	该字段包含影子寄存器编号（0至95）。必须以十六进制的形式写入值。
[value=(value)]	将值加载到选定OTP影子寄存器中。 必须以十六进制的形式写入值。
[sha_rsl=(value)]	将值加载到相应的影子读取粘性锁定位。值可以是0或1。
[sha_wsl=(value)]	将值加载到相应的影子写入粘性锁定位。值可以是0或1。
[sl=(value)]	将值加载到相应的编程粘性锁定位。值可以是0或1。
[pl=(value)]	将值加载到相应的编程永久锁定位。值可以是0或1。

示例

./STM32_Programmer.sh --connect port=usb1 –otp program wordID=0x00 value=0x3f sl=1 wordID=0x08 value=0x18

5.1.13 分离指令

说明：此指令用于向USB DFU发送分离指令。

语法：-detach

5.1.14 GetCertif指令

说明：该指令用于读取芯片证书。

语法：**-gc certification.bin**

5.1.15 写入blob指令

说明：此指令用于发送blob（秘密信息和许可）。

语法：**-wb blob.bin**

5.1.16 显示指令

说明：此指令用于显示全部或部分OTP结构。

语法：**-otp displ [upper] [lower] [ctrl]**

[upper] 用于显示加载的OTP影子寄存器上限值和状态的选项。

[lower] 用于显示加载的OTP影子寄存器下限值和状态的选项。

[ctrl] 用于显示加载的BSEC控制寄存器的选项。

示例

./STM32_Programmer.sh --connect port=usb1 –otp displ

5.2 安全编程SSP特定的指令

安全机密供应（SSP）特性支持安全秘密刷写流程（在STM32 MPU器件上可用）。
STM32MP1系列支持保护机制，可保护关键操作（如加密算法）和关键数据（如密钥）免受意外访问。

本节提供了STM32SSP指令及其相关工具生态系统的概述，并解释了如何在CM产品制造阶段以此来保护OEM机密。

请参考 *使用STM32CubeProgrammer进行安全编程*（AN5054）了解详细信息。

STM32CubeProgrammer用一些选项导出简单的SSP指令，以便执行SSP编程流程。

-ssp, --ssp

说明：SSP 文件编程

语法：**-ssp <ssp_file_path> <ssp-fw-path> <hsm=0|1> <license_path|slot=slotID>**

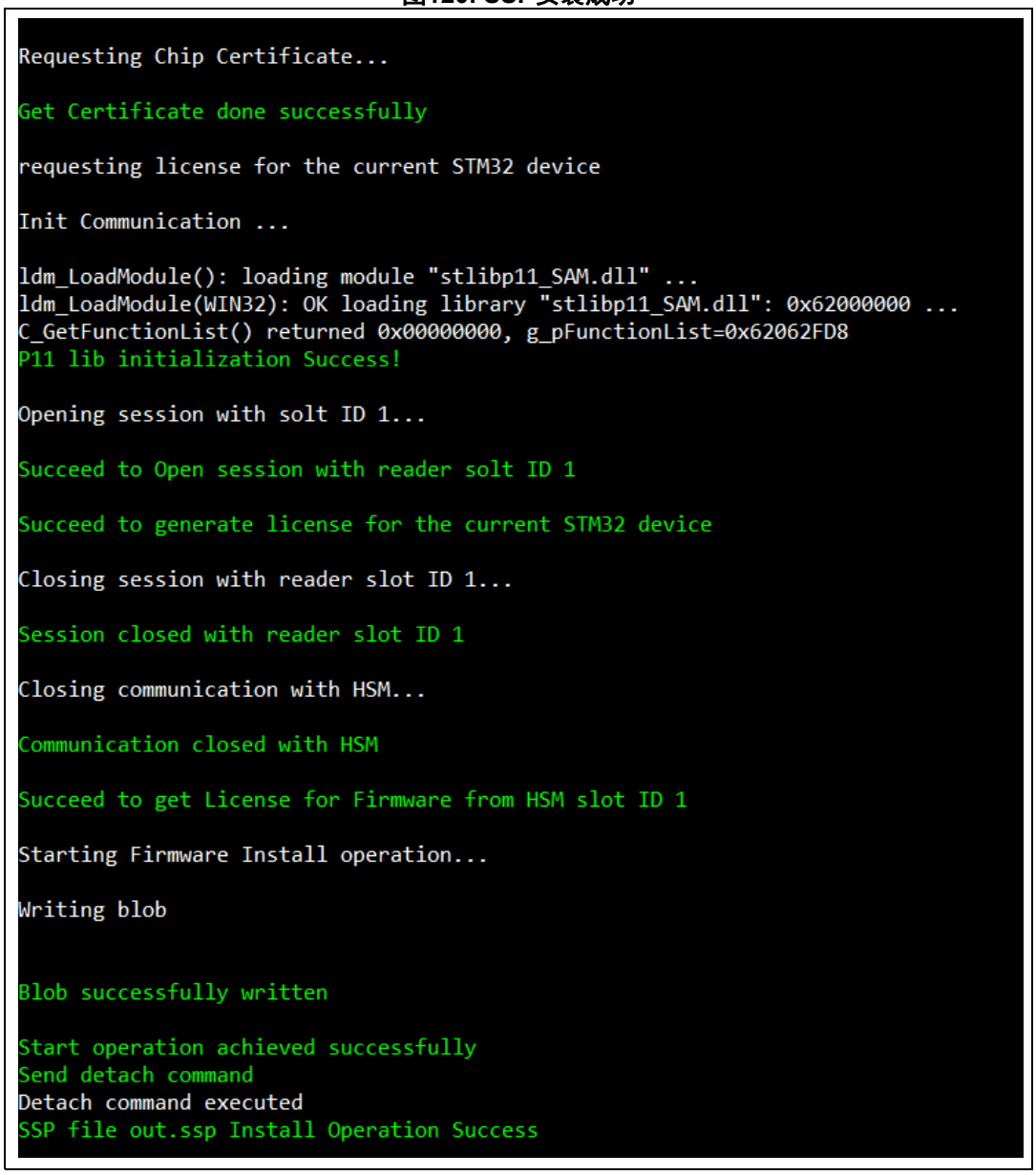
<ssp_file_path>	要编程的 SSP 文件路径，扩展名为 bin 或 ssp。
<ssp-fw-path>	SSP签名固件路径。
<hsm=0|1>	为 HSM 的使用设置用户选项（不使用 HSM 或使用 HSM）。默认值：hsm=0。
<license_path|slot=slotID>	• 许可文件路径（若 hsm=0） • 若使用 HSM（若 hsm=1），为读写器插槽 ID

使用 USB DFU 引导程序接口的示例：

STM32_Programmer_CLI.exe -c port=usb1 –ssp “out.ssp” “tf-a-ssp-stm32mp157f-dk2-trusted.stm32” hsm=1 slot=1

注： *所有 SSP 跟踪数据均显示在输出控制台上。*

图120. SSP安装成功



若有任何错误输入，SSP 过程将中止，然后显示错误信息，指示问题的根本原因。

6 STM32CubeProgrammer C++ API

除了图形用户界面和命令行界面，STM32CubeProgrammer还提供一个C++ API，此API可用于开发应用并利用丰富的特性对STM32微控制器中嵌入的存储器进行编程（通过调试接口或引导加载程序接口（USB DFU、UART、I^2C、SPI和CAN））。

关于C++ API的更多信息，请阅读STM32CubeProgrammer软件包的API\doc文件夹中提供的帮助文件。

7 版本历史

表2. 文档版本历史

日期	版本	变更
2017年12月15日	1	初始版本。
2018年8月2日	2	更新了： – *第 1.1 节：系统要求* – *第 1.2.3 节：macOS 安装* – *第 1.2.4 节：DFU 驱动程序* 增加了： – *第 3.2.8 节：调试指令* – *图1：macOS“允许从以下位置下载应用程序：”选项卡* – *图2：删除旧版驱动程序软件*
2018年9月12日	3	在封面上和*第 2.1.4 节：目标配置面板*中增加了SPI、CAN和I2C设置。 更新了： – *图 11：ST-LINK 配置面板* – *图 87：STM32CubeProgrammer：可用指令*. – *图 92：使用SWD调试端口的连接操作* 替换了*第 3.2.1 节：连接指令*。
2018年11月16日	4	更新了*第 2.1.4 节：目标配置面板*、*第 2.2.1 节：读取和显示目标存储器*、*第 2.2.2 节：读取并显示文件*和*第 2.3.2 节：外部Flash存储器编程*。 更新了*图 9：STM32CubeProgrammer主窗口*、*图 10：展开主菜单*、*图 11：ST-LINK配置面板*、*图 12：UART配置面板*、*图 13：USB配置面板*、*图 14：目标信息面板*、*图 15：SPI配置面板*、*图 16：CAN配置面板*、*图 17：I2C配置面板*、*图 18：存储器和文件编辑：设备存储器选项卡*、*图 20：存储器和文件编辑：文件显示*、*图 21：Flash存储器编程和擦除（内部存储器）*和*图 22：Flash存储器编程（外部存储器）*。 对整个文档进行少量文字修订。
2019年1月3日	5	更新了*第 1.2.4 节：DFU 驱动程序*。 增加了*第 3.2.19 节：SFI特定的指令的安全编程*、*第 3.2.21 节：HSM相关的指令*和*第 6 节：STM32CubeProgrammer C++ API*。 对整个文档进行少量文字修订。
2019年3月4日	6	更新了*Introduction*和*第 1 节：入门指南*。 更新*第 2 节：MCU的STM32CubeProgrammer用户界面*和*第 3 节：MCU的STM32CubeProgrammer命令行界面（CLI）*的标题。 增加了*第 2.5 节：自动模式*、*2.6 节：STM32WB OTA编程*、*第 4 节：MPU的STM32CubeProgrammer用户界面*、*第 5 节：MPU的STM32CubeProgrammer CLI*及其子章节。
2019年4月19日	7	更新了*第 1.1 节：系统要求*、*第 2.2.2 节：读取并显示文件*、*2.6.2 节：OTA更新流程*、*第 3.2.19 节：SFI特定的指令的安全编程*、*第 3.2.21 节：HSM相关的指令*和*第 3.2.22 节：STM32WB特定的指令*。 更新了*图 21：Flash存储器编程和擦除（内部存储器）*。

表2. 文档版本历史（续）

日期	版本	变更
2019年10月11日	8	更新了*图形指南*、*第 3.2.19 节：SFI特定的指令的安全编程*、*第 3.2.21 节：HSM相关的指令*和*第 3.2.22 节：STM32WB特定的指令*。 增加了*第 2.6 节：应用内编程（IAP/USBx）*。 对整个文档进行少量文字修订。
2019年11月8日	9	更新了*第 1.2.1 节：Linux安装*、*第 3.2.22 节：STM32WB特定的指令*和*第 5.1.6 节：读取分区指令*。 对整个文档进行少量文字修订。
2020年1月7日	10	更新了*第 1.1 节：系统要求*、*第 1.2.3 节：macOS安装*和*第 3.2.19 节：SFI特定的指令的安全编程*。 增加了*第 3.2.16 节：TZ降级指令*和*第 3.2.20 节：SFIx特定的指令的安全编程*。 删除了原来的*5.2.12 节：写入BSEC指令*。 对整个文档进行少量文字修订。
2020年2月24日	11	增加了*第 2.7 节：使用图形界面刷写协处理器二进制文件*及其包含的小节。
2020年7月23日	12	增加了*第 2.8 节：串行线查看器（SWV）*、*第 3.2.23 节：串行线查看器（SWV）指令*和*第 5.2 节：安全编程SSP特定的指令*。 更新了*第 3.2.1 节：连接指令*和*第 3.2.2 节：擦除指令*。 对整个文档进行少量文字修订。
2020年11月17日	13	更新了*第 1.1 节：系统要求*、*第 1.2.1 节：Linux安装*、*第 1.2.2 节：Windows安装*、*第 1.2.3 节：macOS安装*、*第 2.3.2 节：外部Flash存储器编程*、*第 2.8 节：串行线查看器（SWV）*、*第 3.2.1 节：连接指令*、*第 3.2.2 节：擦除指令*、*第 3.2.13 节：外部加载指令*、*第 3.2.21 节：HSM相关的指令*、*第 3.2.20 节：SFIx特定的指令的安全编程*、*第 3.2.22 节：STM32WB特定的指令*和*第 5.1.1 节：连接指令*。 增加了*第 2.10 节：具有自定义PID和VID的DFU IAP/USBx*、*第 2.11 节：SigFox™ 凭证*、*使用DFU IAP/USBx选项的示例*、*第 3.2.5 节：下载64位数据指令*、*第 3.2.14 节：使用引导加载程序接口的外部加载程序指令*、*第 3.2.24 节：STM32WL的特定指令*和*5.2.5 节：通过USB串口小工具提供刷写服务*。 更新了*图 22：Flash存储器编程（外部存储器）*、*图 37：SWV窗口*和*图 66：MPU的可用指令*。
2020年11月19日	14	更新了*第 5.1.1 节：连接指令*。 删除了原来的*5.1 节：命令行的使用*和*5.2.5 节：通过USB串口小工具提供刷写服务*。
2021年3月11日	15	更新了*第 1.1 节：系统要求*、*第 1.2.1 节：Linux安装*、*第 1.2.3 节：macOS安装*、*第 2.11 节：SigFox™ 凭证*和*第 3.2.22 节：STM32WB特定的指令*。 增加了*第 2.12 节：寄存器查看器*、*第 2.13 节：硬性故障分析器*及其小节，*第 3.2.26 节：寄存器查看器*以及*第 3.2.27 节：硬性故障分析器*。 对整个文档进行少量文字修订。

表2. 文档版本历史（续）

日期	版本	变更
2021年7月22日	16	更新了*第 2.1.4 节：目标配置面板*、*第 3.2.1 节：连接指令*、*第 3.2.2 节：擦除指令*和*第 3.2.22 节：STM32WB特定的指令*。 增加了 *第 2.14 节：填充存储器指令*、*第 2.15 节：填充存储器操作*、*第 2.16 节：空白检查指令*、*第 2.17 节：空白检查操作*、*第 2.18 节：比较Flash存储器和文件*、*第 2.19 节：比较两个文件*、*第 2.20 节：LiveUpdate特性*和*第 3.2.28 节：具有密码的RDP降级*。 更新了*图 12：UART配置面板*和*图 89：启用COM DTR引脚*。 增加了*图 90：使用USB的连接操作*。 对整个文档进行少量文字修订。
2021年11月17日	17	增加了*第 2.9 节：MCU的STM32CubeProgrammer Script Manager平台*及其包含的小节。 更新了*第 2.1.1 节：主菜单*、*第 2.1.4 节：目标配置面板*、*第 2.6 节：应用内编程（IAP/USBx）*、*第 2.7 节：使用图形界面刷写协处理器二进制文件*及其子章节、*第 2.9 节：MCU的STM32CubeProgrammer Script Manager平台*、*第 3.2.1 节：连接指令*和*第 3.2.22 节：STM32WB特定的指令*。 删除了原来的*2.6 节：STM32WB OTA编程*。 更新了*图 9：STM32CubeProgrammer主窗口*、*图 10：展开主菜单*、*图 11：ST-LINK配置面板*和*图 14：目标信息面板*。 对整个文档进行少量文字修订。
2022年2月28日	18	增加了*第 1.3 节：更新*及其小节，*第 2.4.1 节：MCU解锁（特定于STM32WL系列）*以及*第 3.2.29 节：GetCertif指令*。 更新了*第 1.1 节：系统要求*、*第 2.1.4 节：目标配置面板*、*第 2.7.2 节：密钥配置*、*第 3.2.1 节：连接指令*、*第 3.2.9 节：列表指令*、*第 3.2.22 节：STM32WB特定的指令*和*第 3.2.24 节：STM32WL的特定指令*。 更新了*图 98：可用串行端口列表*。

表3. 中文文档版本历史

日期	版本	变更
2022年8月31日	1	中文初始版本。

重要通知 - 请仔细阅读

意法半导体公司及其子公司（“ST”）保留随时对ST产品和/或本文档进行变更、更正、增强、修改和改进的权利，恕不另行通知。买方在订货之前应获取关于ST产品的最新信息。ST产品的销售依照订单确认时的相关ST销售条款。

买方自行负责对ST产品的选择和使用，ST概不承担与应用协助或买方产品设计相关的任何责任。

ST不对任何知识产权进行任何明示或默示的授权或许可。

转售的ST产品如有不同于此处提供的信息的规定，将导致ST针对该产品授予的任何保证失效。

ST和ST徽标是ST的商标。若需ST商标的更多信息，请参考 *www.st.com/trademarks*。所有其他产品或服务名称均为其各自所有者的财产。

本文档中的信息取代本文档所有早期版本中提供的信息。