UM2237
用户手册

STM32CubeProgrammer软件说明

引言

STM32CubeProgrammer（STM32CUBEPROG）为任意环境下的STM32微控制器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），采用手动操作或通过脚本自动操作。

本用户手册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

图片索引

表格索引

1 入门指南

本节介绍安装STM32CubeProgrammer软件工具的要求和步骤。STM32CubeProgrammer可支持基于Arm® Cortex®-M处理器的STM32 32位器件。

1.1 系统要求

支持的操作系统和架构为：

- Linux® 32位和64位（已在Ubuntu 14.04上测试）
- Windows® 10-7-8 32位和64位
- macOS®（最小版本OS X® Yosemite）

必须安装来自Oracle®的Java™SE运行环境1.8（版本1.8.121或更新版本）。（可从*www.oracle.com*上下载）

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。更多信息请查看用户手册UM2238。

1.2.1 Linux安装

如果使用USB端口连接STM32器件，则需要在机器终端中输入以下命令来安装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，您需要将位于Driver/Dev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。*

要安装STM32CubeProgrammer工具，您需要下载并解压zip包，并执行*SetupSTM32CubeProgrammer-vx.y.z.linux*，它将指导您完成安装过程。

1.2.2 Windows安装

要安装STM32CubeProgrammer工具，您需要下载并解压zip包，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

要安装STM32CubeProgrammer工具，您需要下载并解压zip包，并执行*SetupSTM32CubeProgrammer-vx.y.z.app*，它将指导您完成安装过程。

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DFU Driver文件夹中找到。

注： *如果您的计算机上安装了DFUSE驱动程序，首先需要卸载它，然后运行前面提到的.bat文件。*

图1. 具有DfuSe驱动程序的STM32 DFU器件

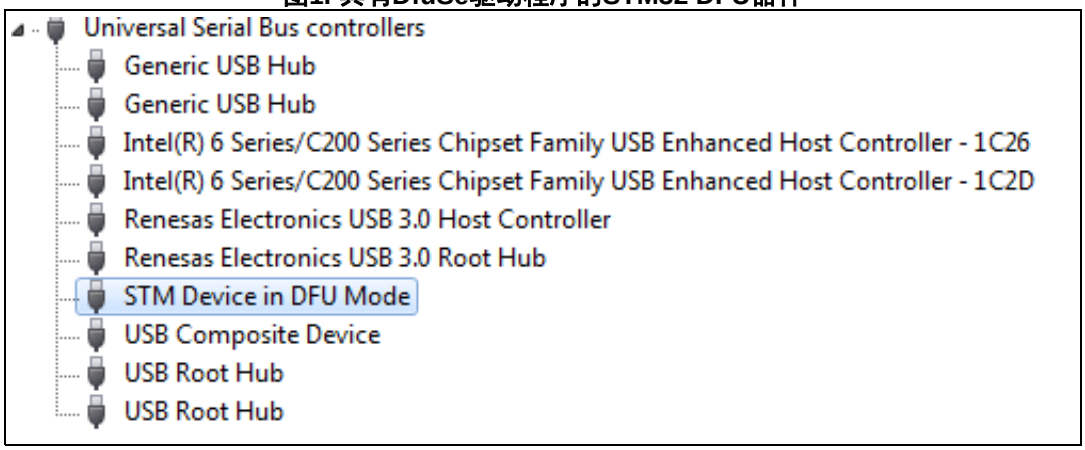
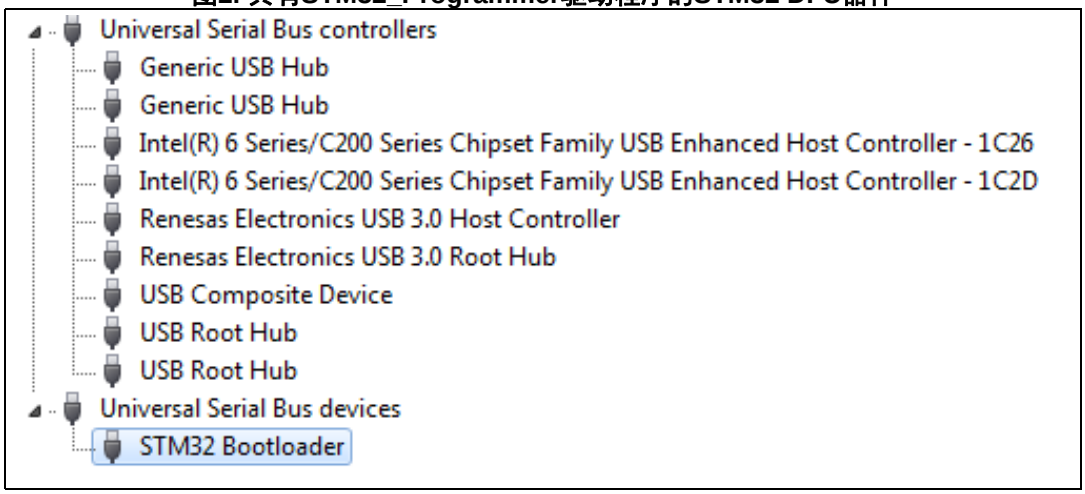


图2. 具有STM32_Programmer驱动程序的STM32 DFU器件



注： *在Windows 7 PC上使用USB DFU接口或STLink接口时，请确保所有USB 3.0控制器的驱动程序都是最新的。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。*

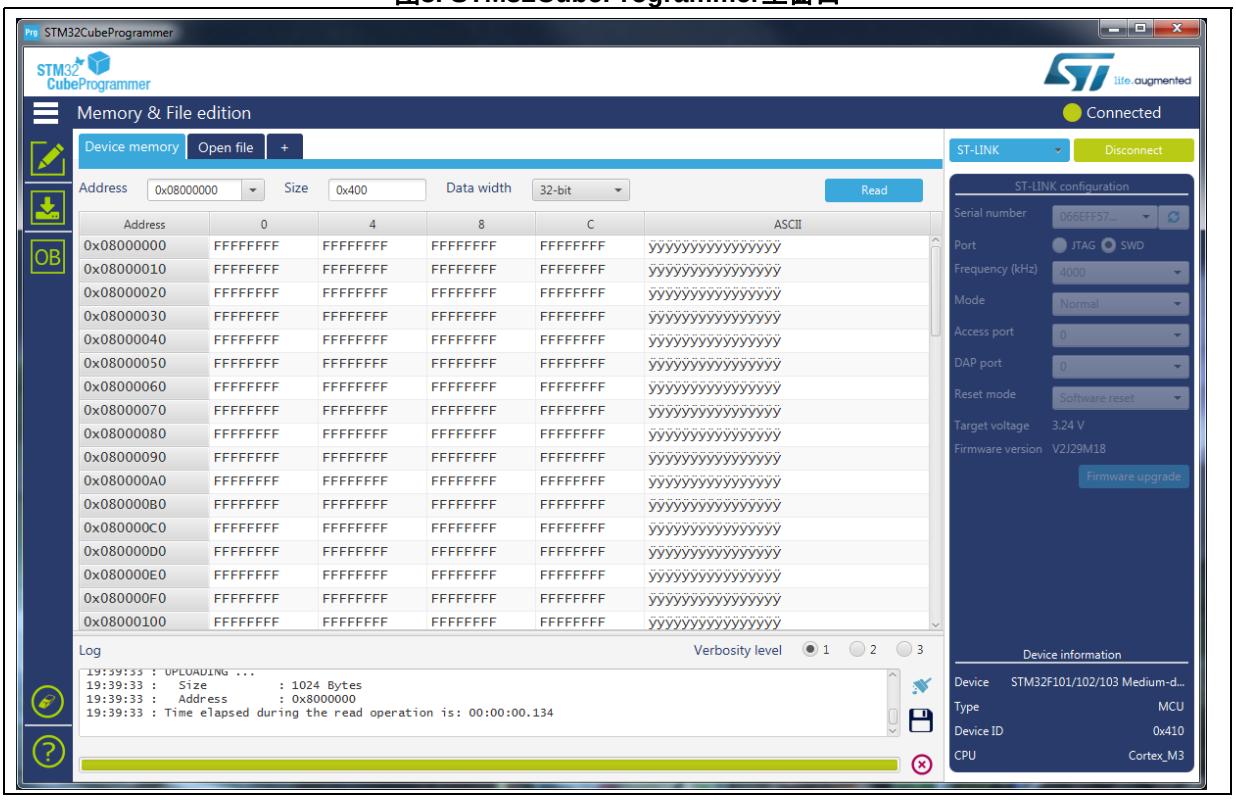
1.2.5 ST-LINK驱动程序

为了能够使用ST-LINK/V2、
ST-LINKV2-1或ST-LINK/V3，通过调试接口连接到STM32器件，您需要运行
*“stlink_winusb_install.bat”*文件来安装ST-LINK驱动程序。该驱动程序随发布包提供，可在*“Driver/stsw-link009_v3”*文件夹下找到。

2 STM32CubeProgrammer用户接口

2.1 主窗口

图3. STM32CubeProgrammer主窗口



主窗口由以下部分组成：

2.1.1 主菜单

主菜单支持在存储器和文件编辑（Memory & file edition）、存储器编程和擦除（Programming & erasing）以及选项字节工具（Option bytes）这三个主面板之间进行切换。

通过点击左上角的Hamburger菜单（三线按钮），主菜单展开并显示文字说明：

图4. 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）。

在日志面板的右侧有两个按钮，一个按钮可以清理日志，第二个按钮可以将日志保存到日志文件中。

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（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

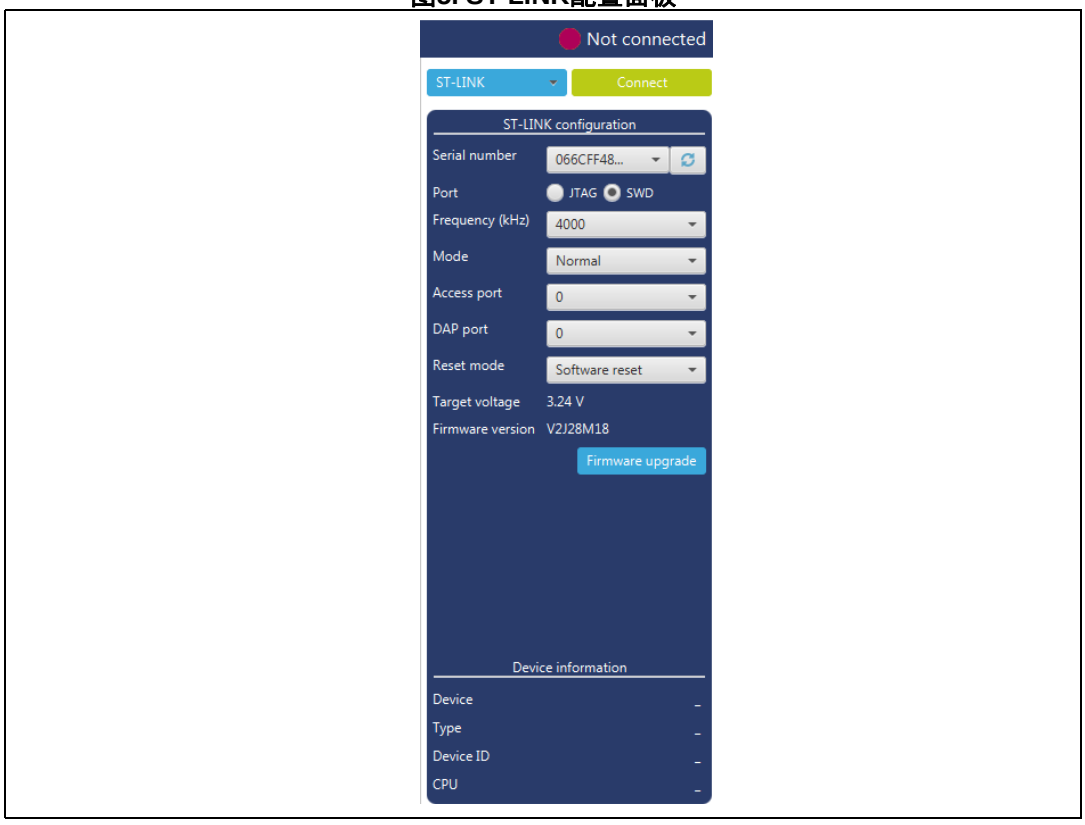
这是连接目标之前要查看的第一个面板。它允许你选择目标接口；可以使用ST-LINK调试工具的调试接口，或者使用经过UART或USB的引导装载程序接口。

刷新按钮能够检查连接到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。

STLINK 设置

图5. ST-LINK配置面板



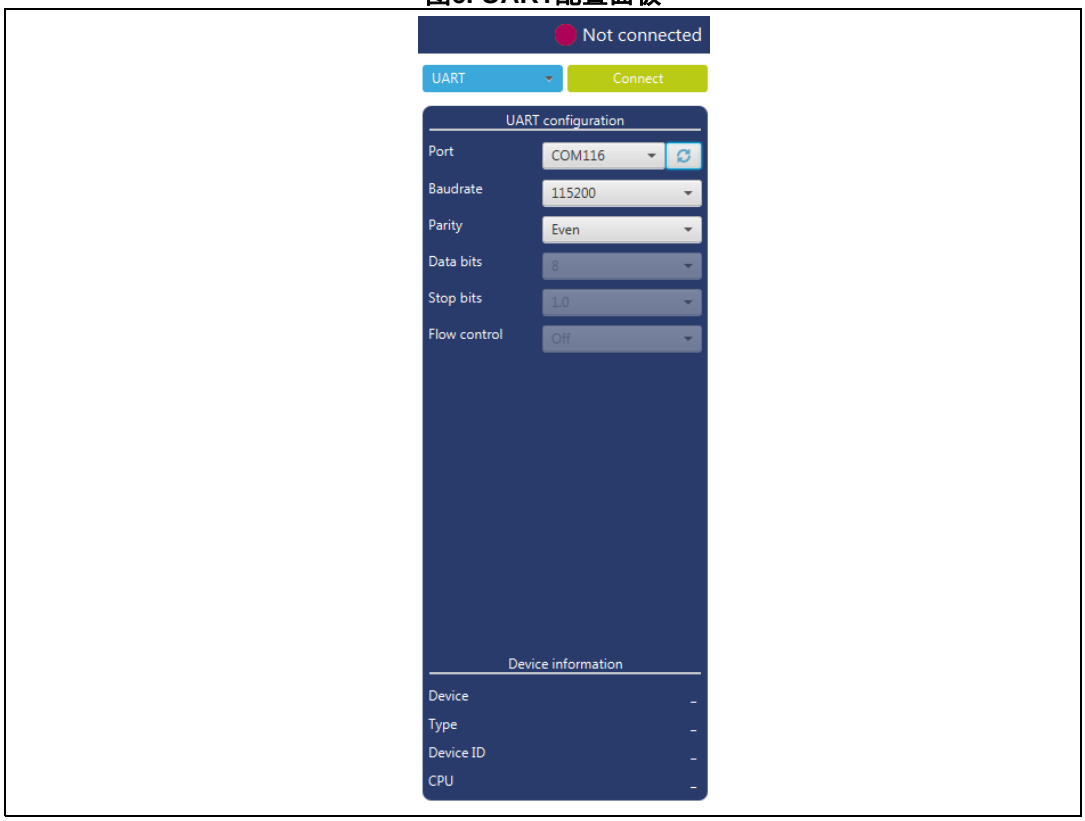
- **序列号**：该字段包含所有连接的ST-LINK工具的序列号。用户可以根据其序列号选择其中一个。
- **端口**：ST-LINK工具支持两种调试协议：JTAG和SWD。

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。
- **DAP端口**： 当前版本不支持。
- **模式**：
  - **正常**：使用“Normal”连接模式时，目标是休眠的，随后挂起。使用“Reset Mode”选项来选择复位的方式
  - **复位下连接**：“Connect Under Reset”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**：“Hot Plug”模式下可以在不停机或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。
- 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引脚应连接到器件复位引脚。
  - **内核复位**： 通过应用中断和复位控制寄存器（AIRCR）仅将内核Cortex-M复位。
- **目标电压**： 使用ST-LINK/V2或ST-LINK/V2-ISOL时，测量目标电压并在此显示。
- **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6. UART配置面板



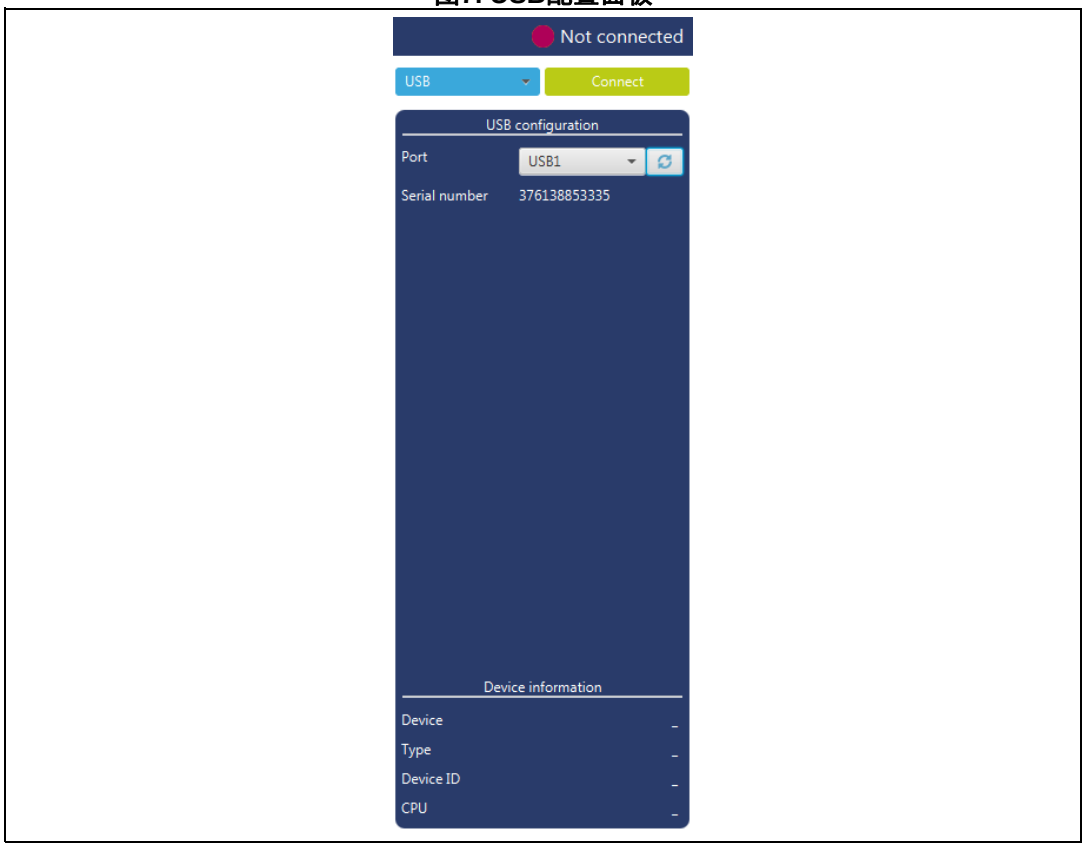
- **端口**：选择目标STM32所连接的COM端口。您可以使用刷新按钮重新检查PC上可用的COM端口。

注： *应该使用启动引脚和/或选项位，使得STM32在引导加载程序模式下启动。关于STM32引导加载程序的更多详细信息，请参考 AN2606。*

- **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，奇偶校验应为“偶”。
- **数据位**： 应始终为8。STM32仅支持8位数据。
- **停止位**： 应始终为1。STM32仅支持1位停止位。
- **流控制**： 应始终关闭。

USB 设置

图7. 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： *应该使用启动引脚和/或选项位，使得STM32在引导加载程序模式下启动。关于STM32引导加载程序的更多详细信息，请参考AN2606。*

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 8*所示：

图8. 目标信息面板

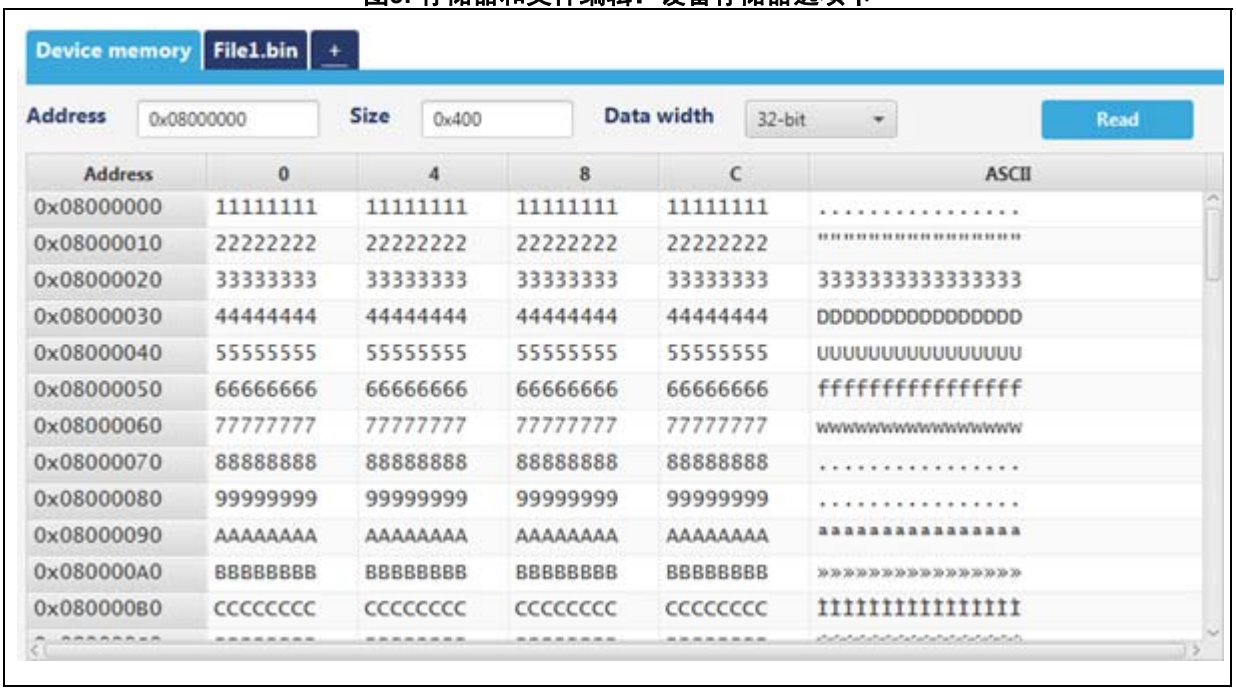

Connected
ST-LINK
Disconnect
ST-LINK configuration
Serial number
066EFF57...
Port
JTAG
SWD
Frequency (kHz)
4000
Mode
Normal
Access port
0
DAP port
0
Reset mode
Software reset
Target voltage
3.25 V
Firmware version
V2J29M18
Firmware upgrade
Device information
Device
STM32F101/102/103 Medium-d...
Type
MCU
Device ID
0x410
CPU
Cortex_M3

2.2 存储器和文件编辑

存储器和文件编辑面板允许你做两件事：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

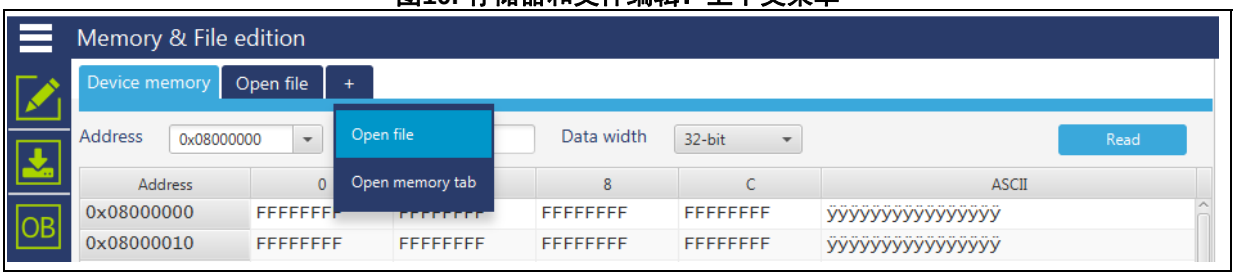
图9. 存储器和文件编辑：设备存储器选项卡



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，您需要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。您可以使用“Data width”组合框以不同格式（8位、16位和32位）显示数据。

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“Device memory”选项卡，或打开文件并将其显示在“文件”选项卡中：

图10. 存储器和文件编辑：上下文菜单

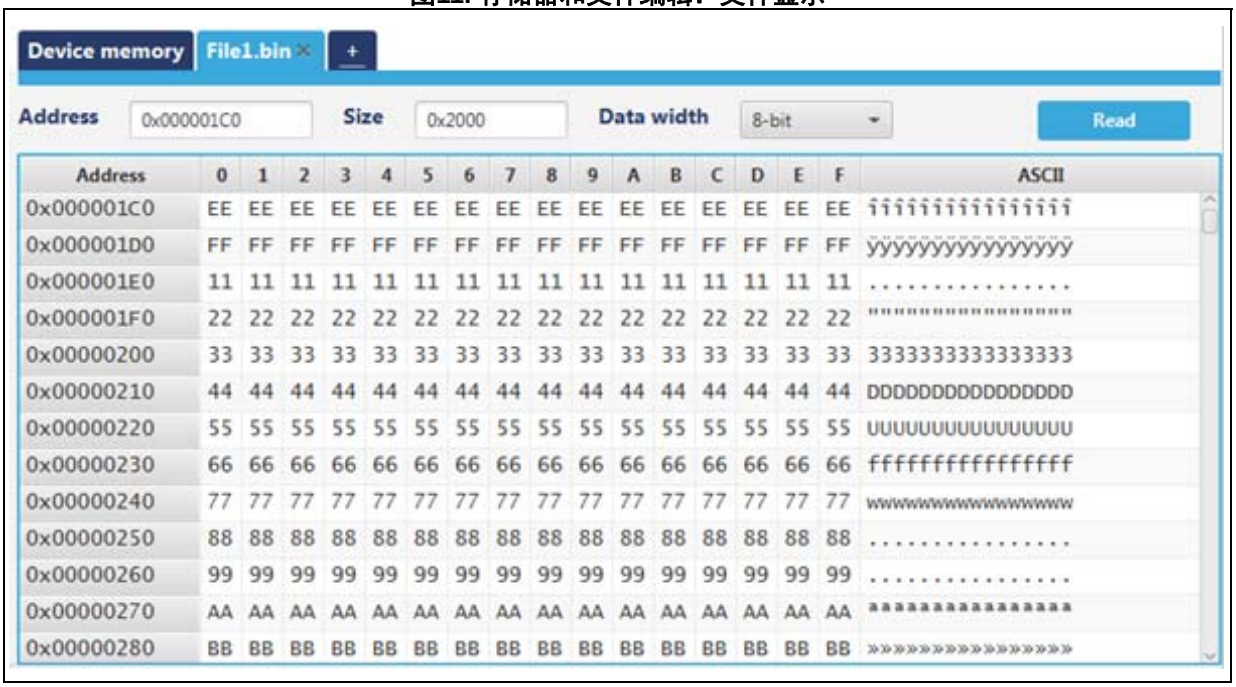


2.2.2 读取并显示文件

要打开并显示文件，只需点击“+”并选择“Open File”菜单，如*图 11*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）。

图11. 存储器和文件编辑：文件显示



文件打开并解析后，会在专用选项卡中显示其文件名，如*图 11*所示。文件大小显示在“Size”字段中，十六进制、srec或ELF文件的起始地址显示在“Address”字段中。对于二进制文件，它为0。您可以修改地址栏，使其从某个偏移量开始显示文件内容。

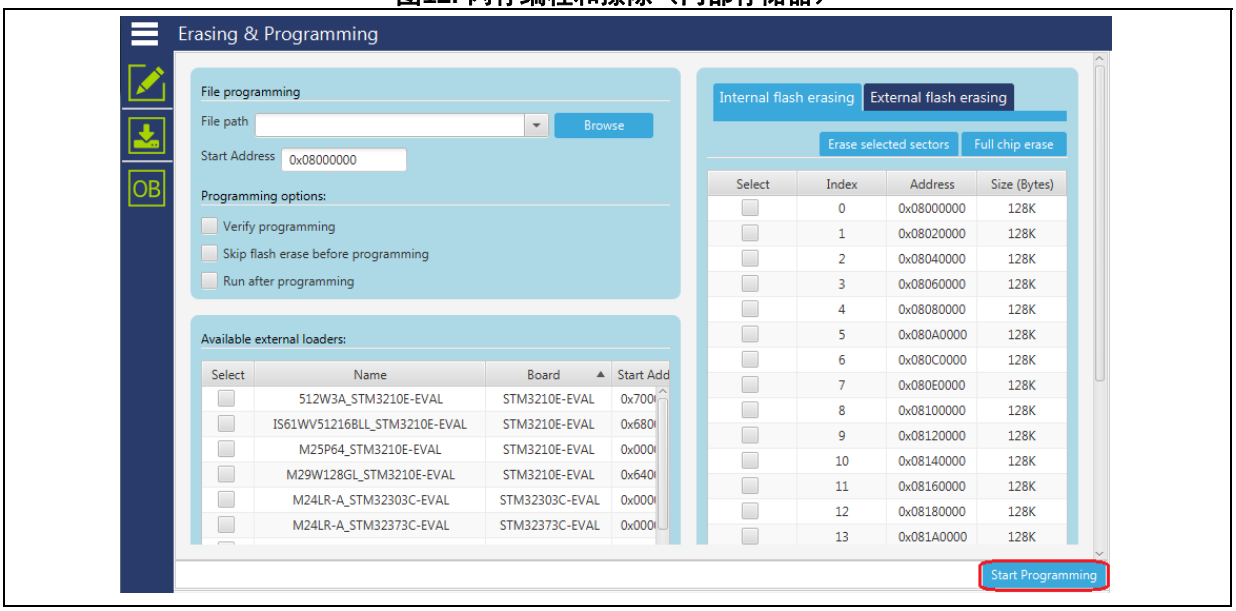
对于“设备内存Device memory”选项卡，您可以使用“Data width”组合框以不同格式（8位、16位和32位）显示文件存储器内容。

2.3 存储器编程和擦除

该面板专用于闪存编程和擦除操作。

2.3.1 内部闪存编程

图12. 闪存编程和擦除（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 sectors”按钮。

“Full chip erase”按钮将擦除所有的闪存。

存储器编程

要编程存储器，您需要执行以下步骤：

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，应该设置编程的地址。
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。
   - 编程前跳过闪存擦除：如果选中该选项，则工具在编程前不会擦除内存。只有当您确定目标内存已被擦除时才应选中该选项。
   - 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。

窗口底部的进度条显示擦除和编程操作的进度。

2.3.2 外部闪存编程

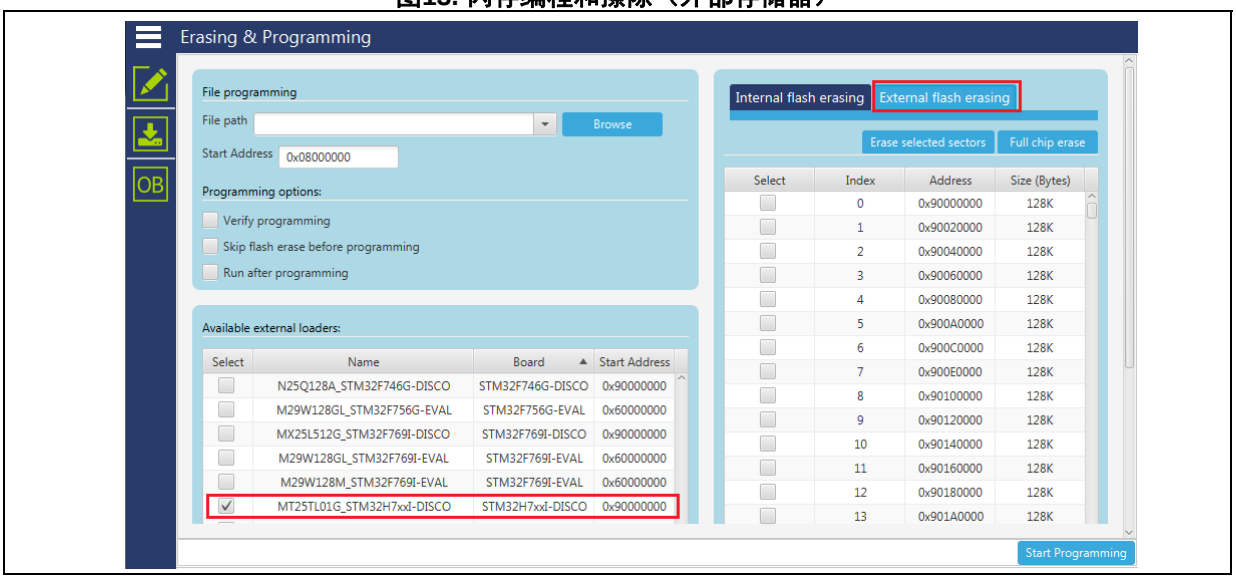
如果您需要通过任意可用接口（SPI，FMC，FSMC，QSPI，OCTOSPI……）对连接到STM32的外部存储器进行编程，那么您需要一个外部加载程序。

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。如果您需要创建新的外部加载程序，请参阅*第 2.3.3节：为外部存储器开发自定义加载程序*获取有关如何创建它的更多详细信息。

要编程外部存储器，您需要选择工具要用的外部加载程序来读取、编程或擦除外部存储器，如图12b所示。选定之后，该外部加载程序就可用于其存储器范围内的任何存储器编程。

右侧的“External flash erasing”选项卡显示存储器扇区，并允许扇区擦除或全芯片擦除。

图13. 闪存编程和擦除（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。这些示例可用于三种工具链：MDK-ARM™，EWARM和TrueSTUDIO®。自定义加载程序的开发可以使用上述三个工具链之一来执行，能够保持相同的编译器/链接器配置，如示例中所示。

外部闪存编程机制与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闪存加载程序都能与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）。*

不应修改链接文件（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” = 要擦除的第一个扇区的地址，“EndAddress” = 要擦除的扇区末尾地址。

注： *注意：该函数在外部SRAM存储器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……）和闪存
- 调用SectorErase()来擦除所需的闪存扇区
- 调用Write()函数来编程存储器。

除了这些函数，我们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,
};
```

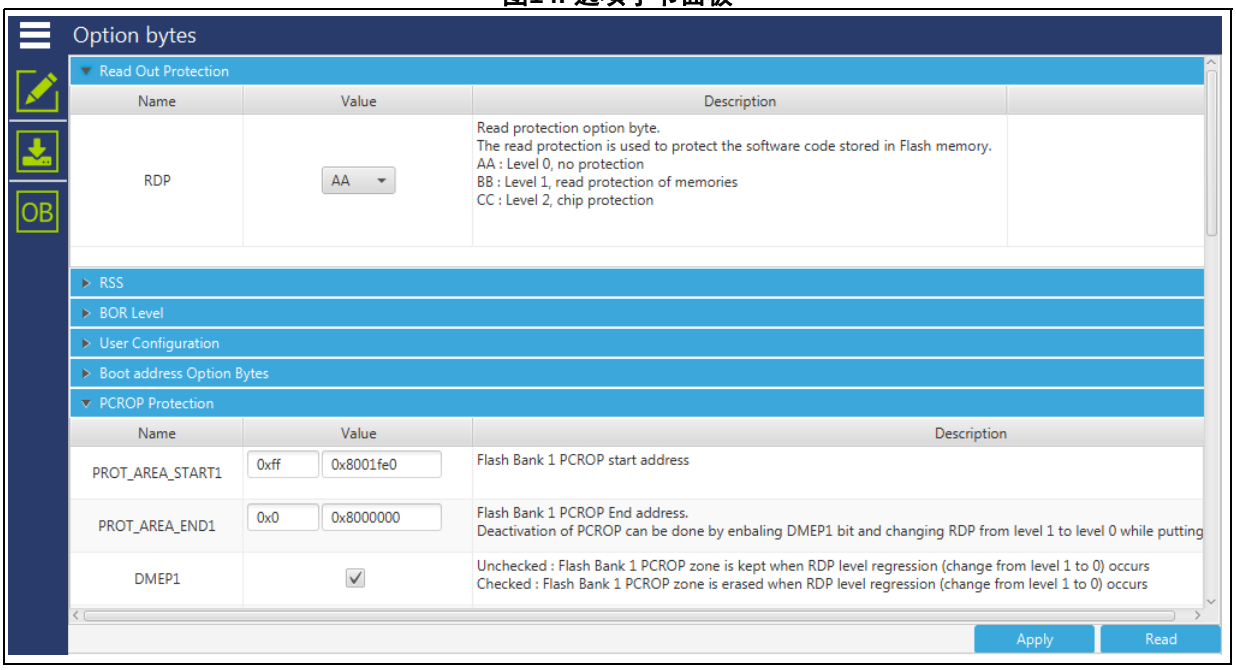
2.4 选项字节

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、其值以及其对设备的影响的描述。

您可以通过更新值字段来修改这些选项字节的值，然后单击应用按钮，进行编程以及验证修改的选项字节是否编程良好。

您可以随时点击读取按钮，读取并刷新显示的选项字节。

图14. 选项字节面板



更多详细信息，请参阅 *www.st.com* 上闪存编程手册和参考手册中的选项字节部分。

3 STM32CubeProgrammer命令行接口（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。可用命令如*图 15*中所示。

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

图15. STM32CubeProgrammer的可用命令

```
Usage :
STM32_Programmer_CLI.exe [command_1] [Agruments_1][[command_2] [Agruments_2]...]

Generic commands

 -?, -h, --help          : Show this help
 -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

 -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
 -c,     --connect       : Establish connection to the device
     <port=<PortName>    : Interface identifer. 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
     [noinit=noinit_bit]: Set No Init bits, value in <0,1> ..., default 0
    JTAG/SWD debug port optional parameters:
     [freq=<frequency>]  : Frequency in KHz. default frequency 4000/STLinkV2
                           1000/STLinkV3 in SWD and 9000 in JTAG with ST-LINK
                           Probes
     [index=<index>]     : Index of the debug probe. default index 0
     [ap=<accessPort>]   : Access Port index to connect to. default ap 0
     [mode=<mode>]       : Connection mode. Value in <UR/HOTPLUG/NORMAL>
                           default mode: NORMAL

 -e,     --erase         : Erase memory pages:
                           devices
     [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
 -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 custon 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系列都支持的一组命令。

3.2.1 连接命令

-c, --connect

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

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，例如COMx（对于windows），/dev/ttySx（对于linux），对于USB接口为usbx

[noinit=<noinit_bit>] ：设置No Init位，值为{0,1} ...，默认为0。

如果先前的连接始终处于活动状态（无需发送0X7F），则使用Noinit = 1

[br=<baudrate>] ：波特率，例如9600, 115200, …，默认值为115200。

[P=<Parity>] ：奇偶校验位，值为(EVEN, NONE, ODD)，默认为EVEN。

[db=<data_bit>] ：数据位，值为(6, 7, 8)，默认值为8。

[sb=<stop_bit>] ：停止位，值为(1, 1.5, 2)，默认值为1。

[fc=<flowControl>] ：流控制，值为(OFF, Software, Hardware)，默认为OFF。

[freq=<frequency>] ：频率，单位为kHz，用于连接。SWD端口默认值为4000 kHz，JTAG端口默认值为9000 kHz。

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=<probe_index>] ：调试工具的索引。默认索引值为0。

[mode=<connect_mode>] ：连接模式。值为<UR/HOTPLUG/NORMAL>。默认值为NORMAL

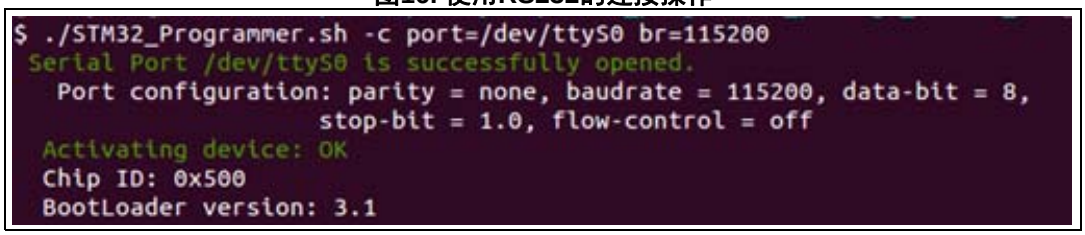
[ap=<access_port>] ：访问端口索引。默认访问索引值为0

示例：

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

此示例的结果如*图 16：使用RS232的连接操作*中所示：

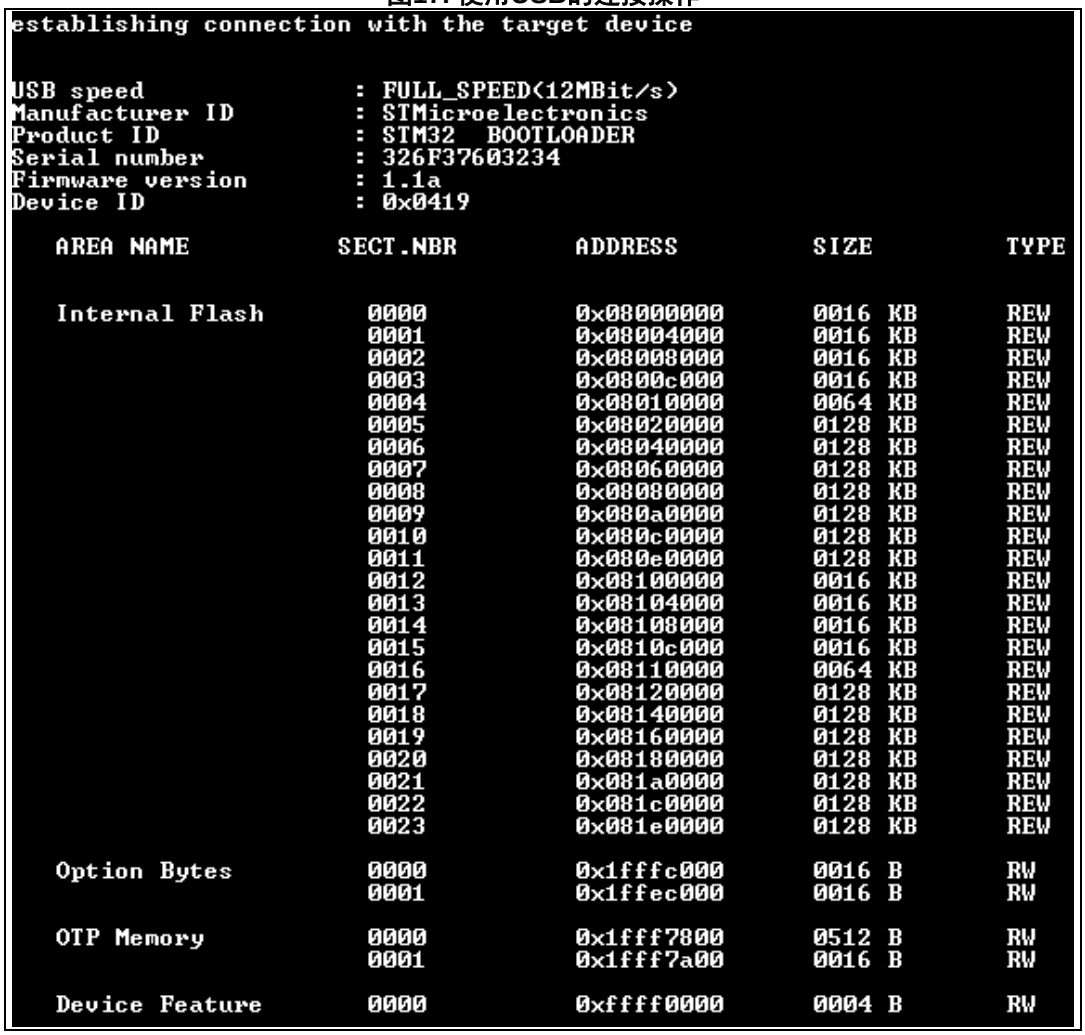
图16. 使用RS232的连接操作



- 使用 USB：
  ./STM32_Programmer.sh -c port=usb1

此示例的结果如*图 17：使用USB的连接操作*中所示：

图17. 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          0x1ffff7a00     0016 B        RW

   Device Feature       0000          0xffff0000      0004 B        RW
```

注： *使用USB接口时，所有配置参数都将被忽略（波特率、奇偶校验、数据位、频率、索引等）要使用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）应根据所用设备进行有效组合。*

使用JTAG/SWD调试端口

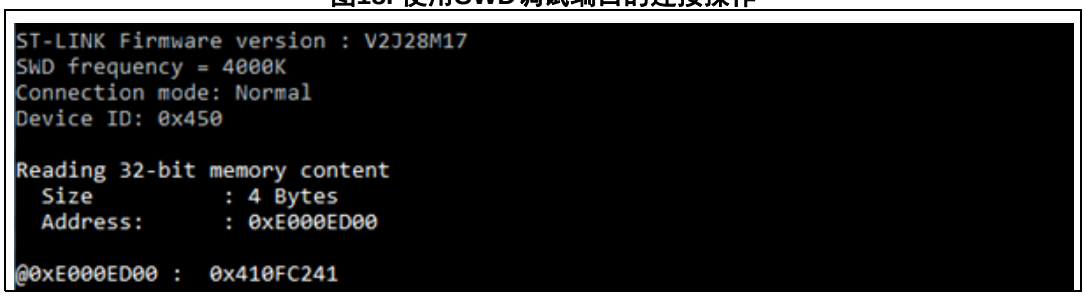
要使用端口连接模式<JTAG / SWD>与ST-LINK工具进行连接，至少需要使用连接命令说明端口名称（例如：-c port=JTAG）

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）。

*图 18*中示例显示了STM32连接到设备ID 0x450的示例：

图18. 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=3 –r32 0xE000ED00 0x4

在连接命令中 (-c port=SWD freq=3900 ap=3)

- <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。*

本例中要连接到访问端口3，如果使用了ap参数，那么连接命令之后使用的任何命令都要通过所选定的访问端口来建立。如本例所示，32位存储器读命令用来通过访问端口3读取地址0xE000ED00。

注： *连接到设备时会显示ST-LINK工具固件版本。*

请确保您拥有ST-LINK固件V2J28M17的最新版本，它在ST网站上有提供（STSW-LINK005）。

3.2.2 Erase指令

-e, --erase

描述：根据给定的参数，该命令可擦除存储器的指定扇区或擦除整个Flash存储器。完成此操作可能需要1秒或更长时间，具体时长取决于所擦除的存储器的大小。

语法：-e [all] [sectorsCodes]

[all]	：擦除全部闪存。
[sectorsCodes]	：仅擦除指定扇区。
[<[start end]>]	擦除从“start”到“end”代码的所有扇区。

示例：

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

此命令仅擦除扇区2和4。

3.2.3 下载命令

-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，然后下载闪存。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

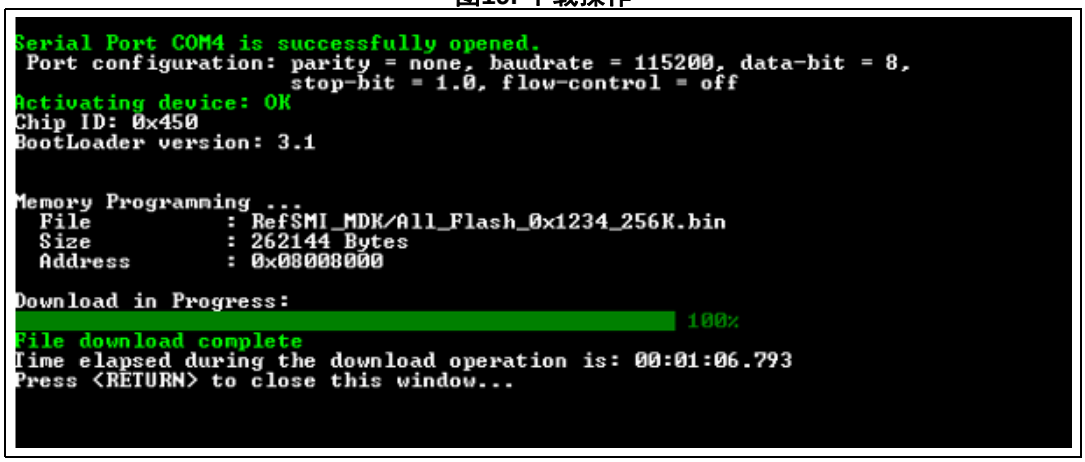
示例：

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

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

此示例的结果如*图 19*中所示。

图19. 下载操作



注： *要验证下载是否成功，您可以在写入命令后调用验证选项（-v或-verify），否则将忽略验证选项。*

3.2.4 下载32位数据命令

-w32

说明： 将指定的32位数据从指定地址开始下载到闪存中。

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。数据应使用空格进行分割

示例：

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

注： *该命令允许32位数据（0x12345678，0xAABBCCFF，0x12AB34CD）从地址0x08000000开始写入闪存*

3.2.5 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”

此命令可以从地址0x20007000开始读取2000个字节，并将其上传到二进制文件 *“/local/benayedh/Binaries/read2000.bin”*

-r32

说明：读取32位数据存储器。

语法： -r32 <start_address> <size>

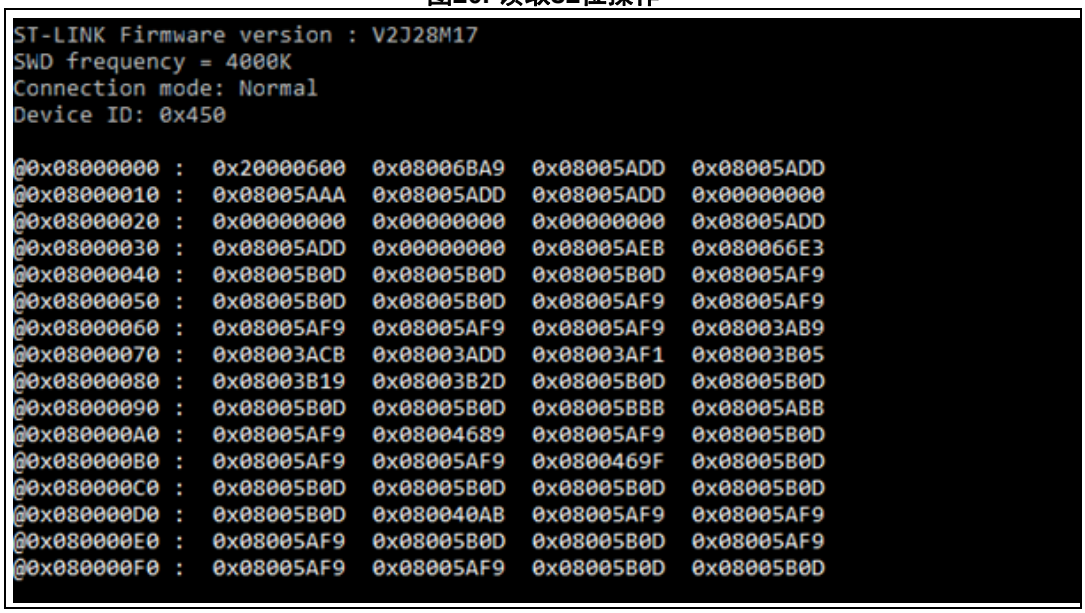
<start_address> ：读取的起始地址。

<size> ：要读取的存储器内容大小。

示例：

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

图20. 读取32位操作



注： *-r32命令最大支持32KB。*

3.2.6 start指令

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

说明：该命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

该命令运行在0x08000000处指定的代码。

3.2.7 List指令

-l, -list

说明：此命令列出所有可用的RS232串行端口。

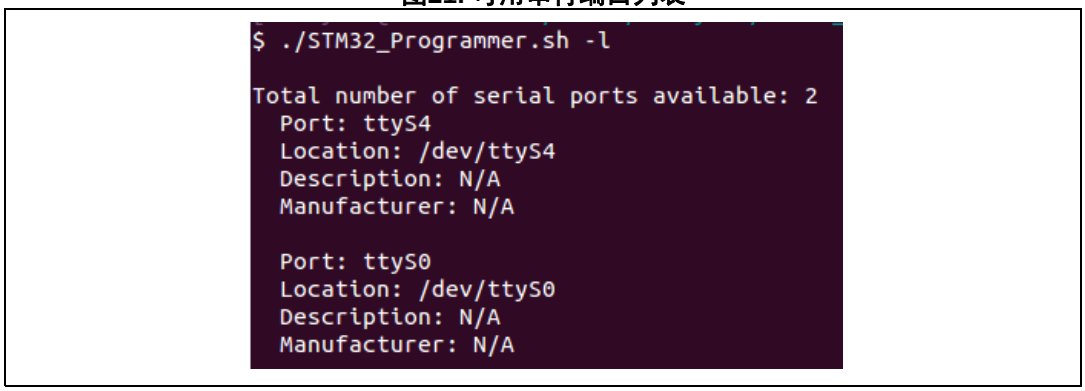
语法： -l, --list

示例：

./STM32_Programmer.sh --list

此示例的结果如*图 21：可用串行端口列表*中所示：

图21. 可用串行端口列表



注： *JTAG/SWD 调试端口不支持该命令。*

3.2.8 QuietMode指令

-q, --quietMode

说明：该命令在下载和读取命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

3.2.9 Verbosity命令

-vb, --verbosity

说明该命令允许显示更多消息，以便更加详细。

语法： -vb <level>

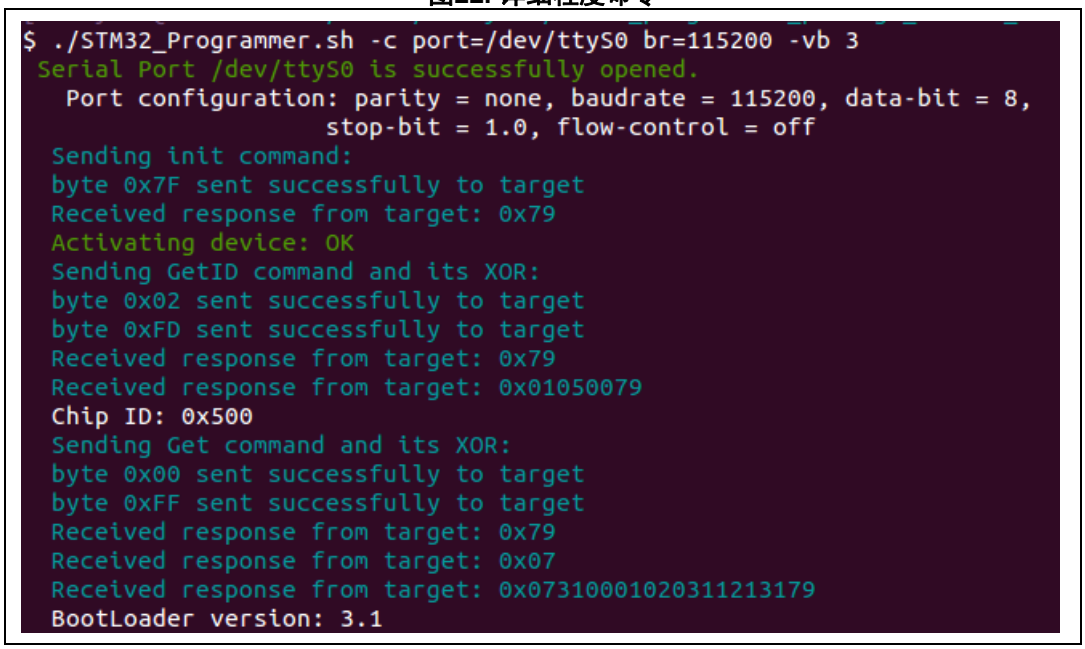
<level> ：详细级别，值为{1, 2, 3} 默认值 vb=1

示例：

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

此示例的结果如*图 22：详细程度命令*中所示：

图22. 详细程度命令



3.2.10 Log命令

-log, --log

说明：此可追溯性命令允许将整个流量（详细级别最高）存储到日志文件中。

语法： -log [filePath.log]

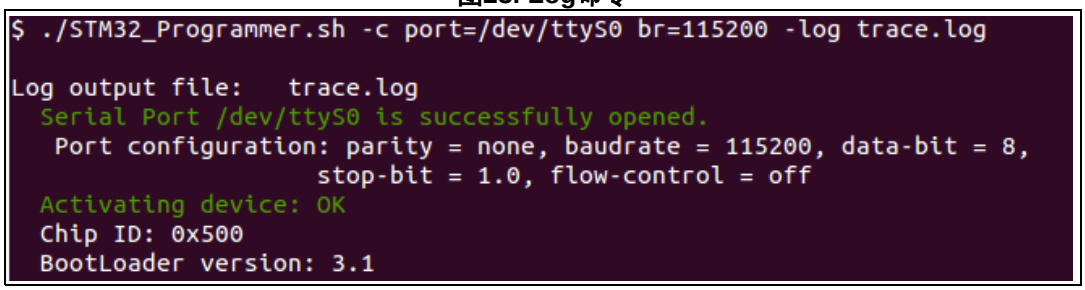
[filePath.log] ：日志文件路径，默认路径为 $HOME/.STM32CubeProgrammer/trace.log

示例：

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

此示例的结果如日志命令和*图 23：Log命令*中所示：

图23. Log命令



日志文件trace.log包含有详细信息，如下：

图24. 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
```

3.2.11 外部加载命令

-el

说明：该命令允许输入外部存储器加载程序的路径，使用外部存储器执行编程、写入擦除和读取操作。

语法： -el [externalLoaderFilePath.stldr]
[externalLoaderFilePath.stldr] 外部加载文件的绝对路径。

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”

注： *仅有SWD/JTAG端口支持该命令。*

3.2.12 Read Unprotect

-rdu, --readunprotect

说明：此命令通过将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

3.2.13 选项字节命令

-ob, --optionbytes

说明：该命令允许用户通过显示或修改设备的选项字节来控制它们。

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

[displ] ： 该选项允许用户显示整套选项字节。

[OptByte=<value>]： 该选项允许用户编程给定的选项字节。

示例：

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

注： 关于设备选项字节的更多详细信息，请参阅 www.st.com网站上设备闪存编程手册和参考手册中的选项字节部分。

3.2.14 Safety lib命令

-sl, --safelib

说明：该命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。

支持格式为：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> ：闪存起始地址

<end_address> ：闪存结束地址

<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 25：Safety lib命令*中：

图25. Safety lib命令

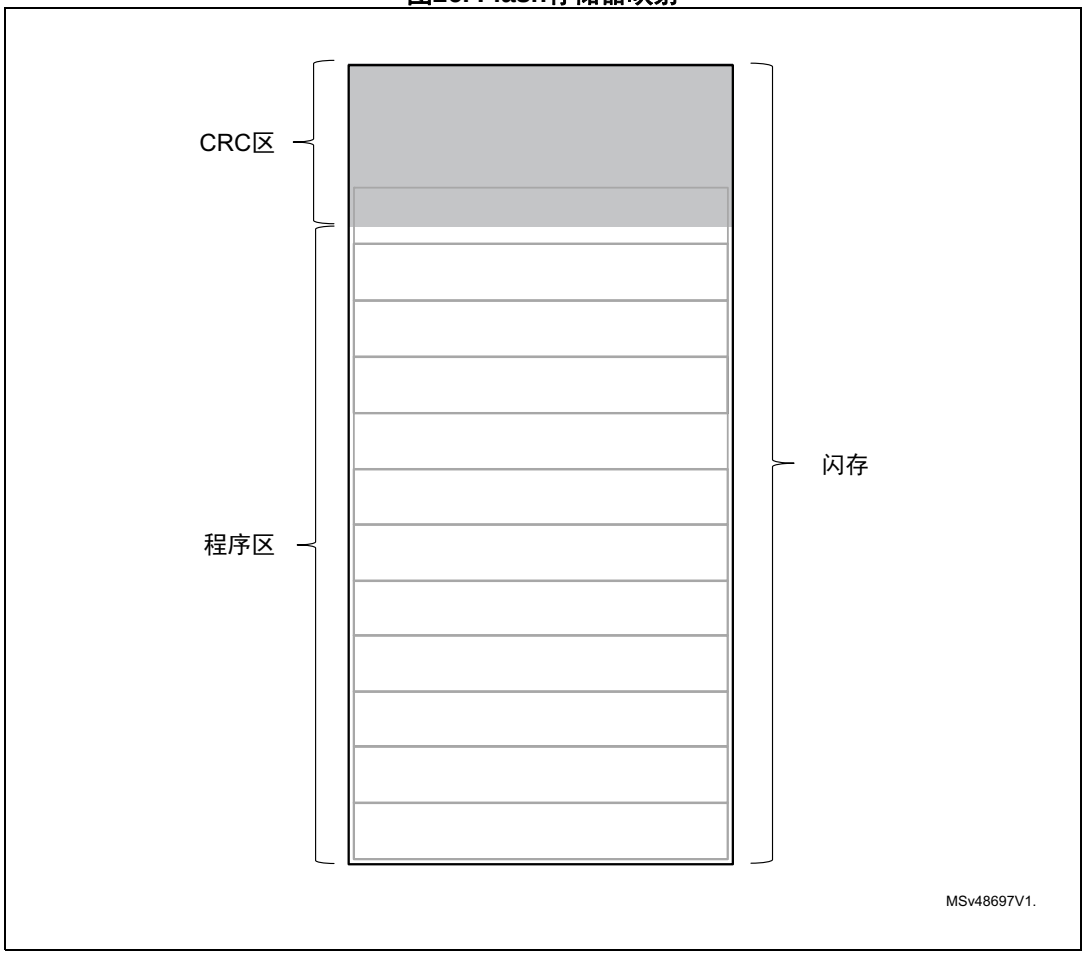
```
C:\bin>STM32_Programmer_CLI.exe -sl TestCRC.axf 0x8000000 0x8010000 0x400
-------------------------------------------------------------------
                        STM32CubeProgrammer v0.4.0-RC1
-------------------------------------------------------------------

Warning: The ELF file will be overwritten
CRCs area injected succesfully
```

闪存程序存储器分为多个片段（如上例所示，片段大小作为安全库命令的参数给出）。

对于每个片段，分别计算CRC值并将其置于CRC区域中。

CRC区位于闪存末尾，如*图 26：Flash存储器映射*所示：

图26. Flash存储器映射



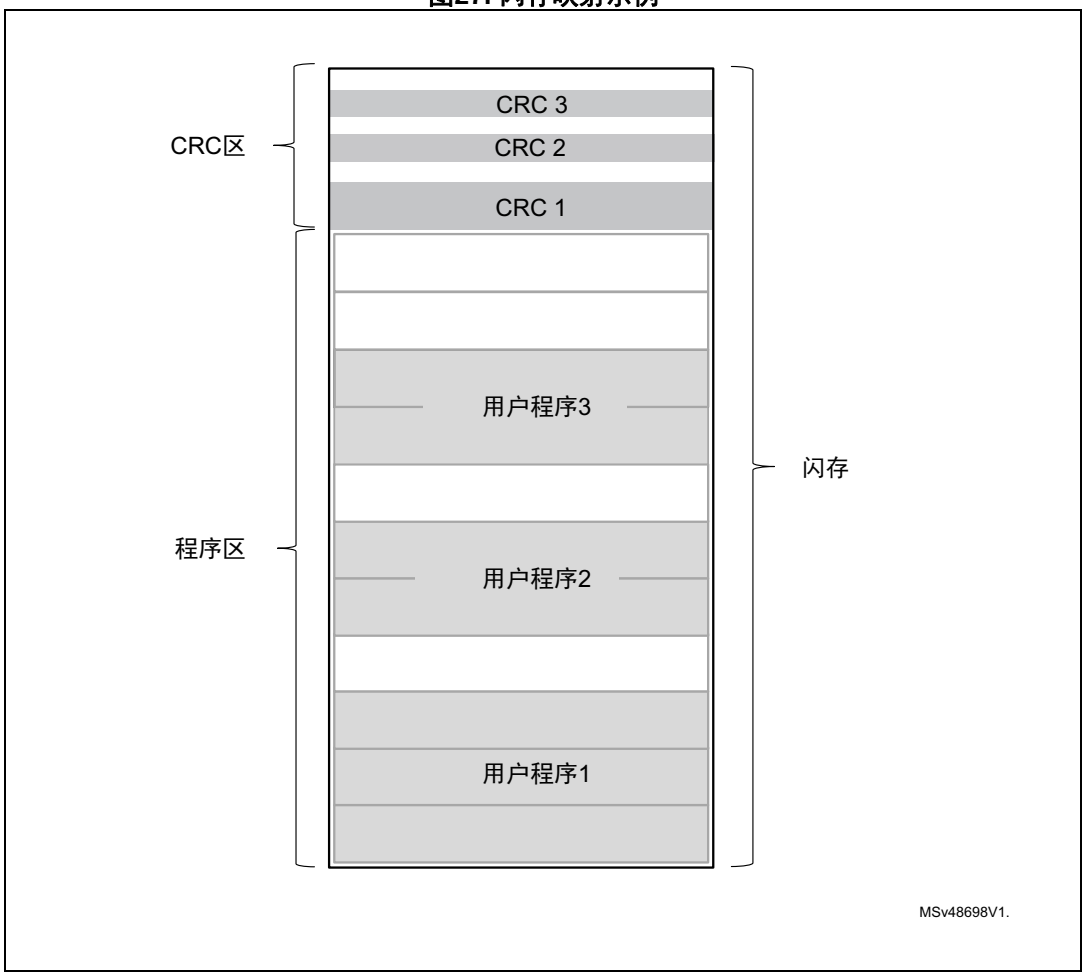
CRC区域的地址和大小确定如下：

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值根据闪存中用户程序的位置放置，参见*图 27：闪存映射示例*:

图27. 闪存映射示例



CRC区域内的CRC范围地址计算如下：

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

4 版本历史

表1. 文档版本历史

日期	版本	变更
2017年12月15日	1	初始版本。

表2. 中文文档版本历史

日期	版本	变更
2018年9月3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。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。本文档的中文版本为英文版本的翻译件，仅供参考之用；若中文版本与英文版本有任何冲突或不一致，则以英文版本为准。