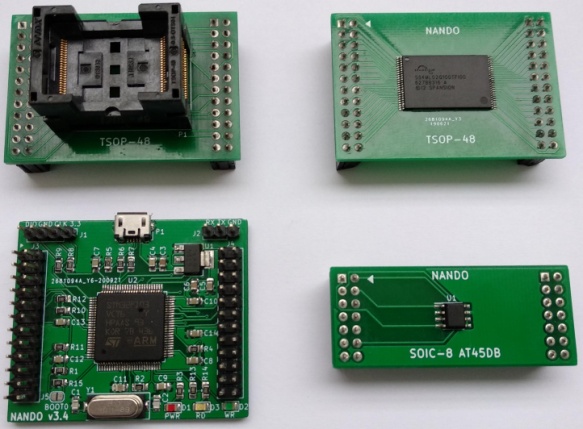
NANDO （NANDopen） 编程器

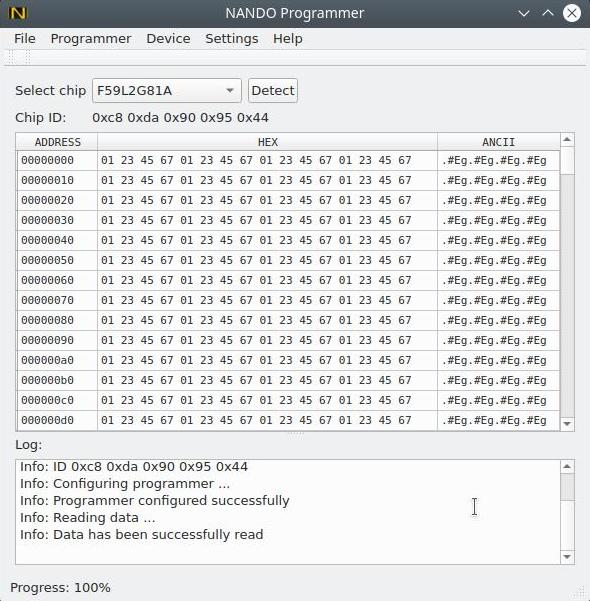
## 常规

NANDO是基于STM32处理器的开源NAND编程器。它支持并行 NAND 和 SPI 闪存编程。

电路板：

[](https://github.com/bbogush/nand_programmer/blob/master/img/board.jpeg)

应用：

[](https://github.com/bbogush/nand_programmer/blob/master/img/host_app.jpeg)

## 特征

* 接口
* 适用于 Linux 和 Windows 的 PC 客户端软件。
* 用于 NAND 芯片的 TSOP-48 插座适配器（与 TL866 适配器兼容）
* 用于 NAND 芯片的 TSOP-48 焊接适配器
* 8 位并行 NAND 接口
* SPI 接口
* 3.3V 电源
* NAND 读取、写入和擦除支持
* 芯片 ID 的 NAND 读取支持
* 坏块的 NAND 读取
* NAND 坏块跳过选项
* NAND 包括备用区域选项
* Open KiCad PCB & Diagram
* 开源代码
* 读写指示灯指示
* 可扩展的芯片数据库
* 芯片自动检测
* 固件更新

### 支持的芯片

#### 并行 NAND：

K9F2G08U0C、HY27US08121B、TC58NVG2S3E、F59L2G81A、MX30LF2G18AC、MX30UF2G18AC等。

查看支持的芯片[qt/nando\_parallel\_chip\_db.csv](https://github.com/bbogush/nand_programmer/blob/master/qt/nando_parallel_chip_db.csv) 的完整列表

#### SPI 闪存

AT45DB021D、MX25L8006E、W25Q16JV等。

查看支持的芯片[qt/nando\_spi\_chip\_db.csv](https://github.com/bbogush/nand_programmer/blob/master/qt/nando_spi_chip_db.csv) 的完整列表

## 发布二进制文件

您可以下载用于 Ubuntu/Windows 安装程序的主机应用程序 deb 包，固件二进制文件 STM32，来自[谷歌驱动程序](https://drive.google.com/drive/folders/1d5hP6MKbkEBGQe_xIRl4-A5LPt_-9QIN)的原理图和gerber文件。

### 许可证

一般来说，sorce 代码、PCB 和原理图都遵循 GPLv3 许可证，但有以下限制：

固件/libs/spl/CMSIS/License.doc

固件/libs/spl/STM32\_USB-FS-Device\_Driver/ -<http://www.st.com/software_license_agreement_liberty_v2>

固件/usb\_cdc -<http://www.st.com/software_license_agreement_liberty_v2>

## 维基

查看[WiKi](https://github.com/bbogush/nand_programmer/wiki)页面以获取更多信息。

**目录**

1. [常见问题](https://github.com/bbogush/nand_programmer/wiki#faq)
2. [问题和限制](https://github.com/bbogush/nand_programmer/wiki#issues_and_limitations)
3. [印刷电路板](https://github.com/bbogush/nand_programmer/wiki/PCB)
4. [电路板组件](https://github.com/bbogush/nand_programmer/wiki/Components)
5. [焊接](https://github.com/bbogush/nand_programmer/wiki/Soldering)
6. [刻录固件](https://github.com/bbogush/nand_programmer/wiki/BurnFirmware)
7. [添加新的并行 NAND 芯片](https://github.com/bbogush/nand_programmer/wiki/AddNANDChip)
8. [添加新的 SPI 芯片](https://github.com/bbogush/nand_programmer/wiki/AddSPIChip)
9. [并行 NAND 芯片连接](https://github.com/bbogush/nand_programmer/wiki/NANDChipConnection)
10. [SPI 芯片连接](https://github.com/bbogush/nand_programmer/wiki/SPIChipConnection)
11. [程序员设置](https://github.com/bbogush/nand_programmer/wiki/Settings)
12. [程序员操作](https://github.com/bbogush/nand_programmer/wiki/Operations)
13. [软件构建](https://github.com/bbogush/nand_programmer/wiki/Build)
14. [印刷电路板变更 （KiCad）](https://github.com/bbogush/nand_programmer/wiki/KiCad)

[**目录** 3](#_Toc121256405)

[1常见问题 4](#_Toc121256406)

[2问题和限制 4](#_Toc121256407)

[3印刷电路板 4](#_Toc121256408)

[4电路板组件 5](#_Toc121256409)

[5焊接 6](#_Toc121256410)

[6刻录固件 7](#_Toc121256411)

[7添加新的并行 NAND 芯片 10](#_Toc121256412)

[8添加新的 SPI 芯片 13](#_Toc121256413)

[9并行 NAND 芯片连接 17](#_Toc121256414)

[10 SPI 芯片连接 17](#_Toc121256415)

[11编程器设置 18](#_Toc121256416)

[12编程器操作 18](#_Toc121256417)

[13软件构建 19](#_Toc121256418)

[14印刷电路板变更 （KiCad） 20](#_Toc121256419)

# 1常见问题

错误：“无法打开 /dev/ttyACM0：权限被拒绝”

以 root 身份运行 NANDO 或添加以下 udev 规则以进行永久配置：  
/etc/udev/rules.d/50-nando.rules

SUBSYSTEMS=="usb", ATTRS{idVendor}=="0483", ATTRS{idProduct}=="5740", MODE:="0666"

错误：“无法打开 /dev/ttyACM0：设备或资源繁忙”

调制解调器管理器尝试检测 USB 设备。等待一分钟或停止调制解调器管理器：

sudo service ModemManager stop

读取 ID 显示 0xFF 0xFF...， 0xFE 0xFE...或其他模式

芯片未正确连接到主板。问题可能是以下之一：

- TSOP-48 插座无法为未焊接的 NAND 提供可靠的连接，因为引脚不平坦。 尝试使用“焊接适配器”或调整插座中的 NAND。

- MCU 和适配器/NAND 之间没有连接。尝试通过测试仪检查连接。

- 杂散电容。确保焊接后没有助焊剂残留物。

- 不支持芯片。支持检查引脚排列和电压。

- 配置错误。如果添加了新芯片，请确保配置正确。

哪些MCU项目支持？

目前仅支持一个单片机STM32F103VCT6。该项目可以移植到包含FSMC和SPI接口，256MB闪存和64个SRAM的MCU。

# 2问题和限制

Windows 10 上的闪存写入速度低。可能是驱动程序有问题。

不支持 ECC 计算。为了恢复某些设备固件，需要带有备用区域的转储。

# 3印刷电路板

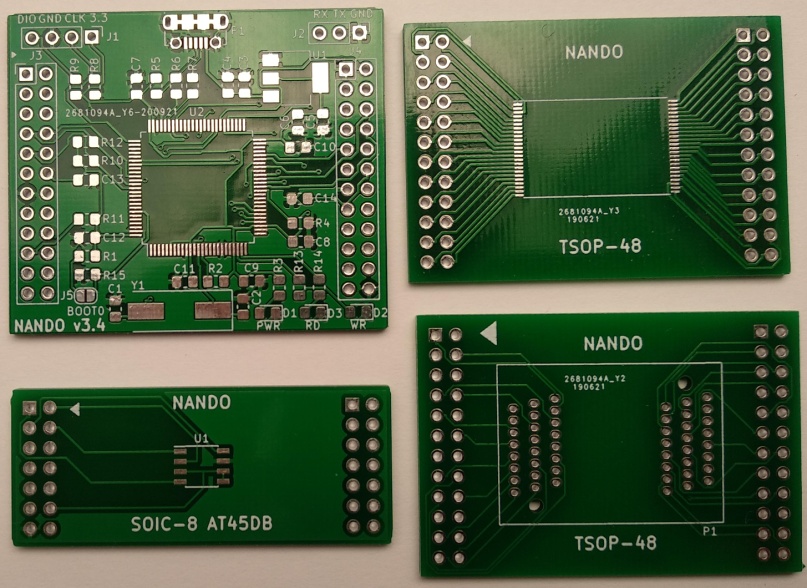
波格丹·博古什编辑了此页面on Apr 19, 2021·[1 修订](https://github.com/bbogush/nand_programmer/wiki/PCB/_history)版

* 从[Google Drive](https://drive.google.com/drive/folders/1zLtebfLW6CxaWMq0rr4EfEPmeXfGk-7n?usp=sharing)release/nando-x.x.x/PCB/gerber/ 下载 Gerber 文件。
* 例如，在PCB制造网站上注册 https://jlcpcb.com/
* 创建订单并加载 gerber 文件。

配置PCB参数，如下例所示：



收到的电路板：



# 4[电路板组件](https://github.com/bbogush/nand_programmer/wiki/Components)

Google 云端硬盘上的发布目录包含一个 BOM.html（物料清单）文件。该文件包含设备所需部件的列表。 BOM.html 提供 https://www.farnell.com/ 参考，因此您可以在本网站上按 ID 和订单查找零件，或在本地商店查找模拟零件。

该器件基于采用LQFP100封装的STM32F103VCT6，具有256KB闪存和48KB SRAM。 不支持其他 MCU，因为固件 HAL 仅支持当前系列。 在某些情况下，由于闪存大小（<256 KB）或缺少FSMC，理论上不可能移植到另一个MCU，如STM32F103Vx.

晶体采用HC49-SD封装，8MHz和18pF负载电容。 如果缺少具有相同负载电容的晶体，则可以使用另一个晶体。负载电容器的值应固定如下：

Cl1 = Cl2 = 2 \* （Cl - Cs） = 2 \* （18pF - 3pF） = 30pF Cl1 = Cl2 = 2 \* （Cl - Cs） = 2 \* （16pF - 3pF） = 26pF。

其中Cl1，Cl2 - 晶体的负载电容器值;Cl - 数据表中的负载电容;Cs - PCB的杂散电容，近似值。

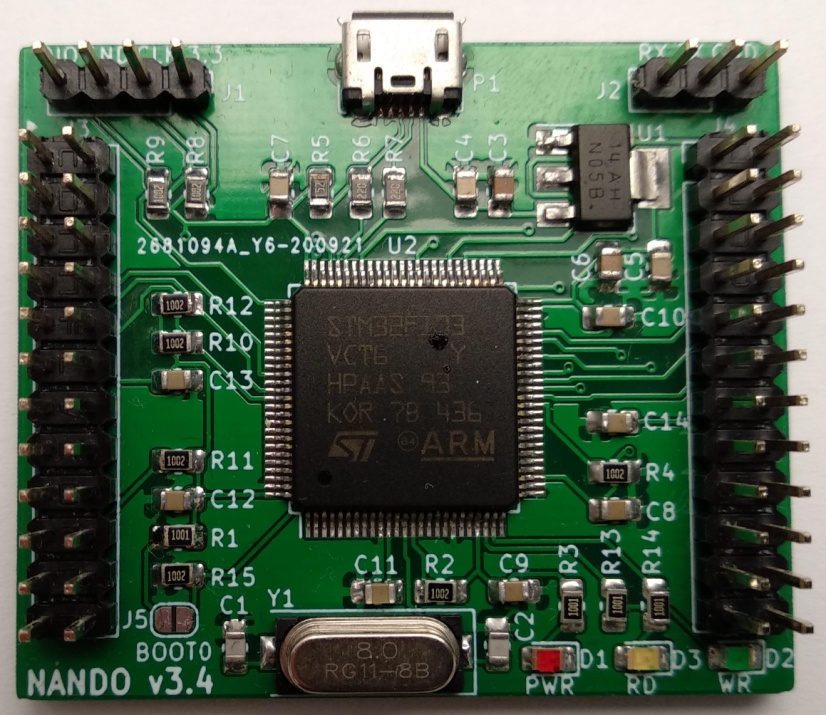
电阻器、电容器、LED 采用 0805 封装。针座，步长为 2.54mm。

适配器的部件不包括在 BOM 文件中，但它只是几个引脚接头和 TSOP-48 插座。TSOP-48 插座可以在 https://www.ebay.com 上购买。 另一种选择是为 TL866 订购 TSOP-48 适配器。



# 5[焊](https://github.com/bbogush/nand_programmer/wiki/Soldering)接

该板可以使用普通烙铁进行焊接.唯一的要求是用“kontakt u”或超声波浴等化学品清洁助焊剂（即使它是“免清洗的”）。 否则将存在高杂散电容，并且设备将无法按预期工作。 该原理图可在谷歌云端硬盘发布目录中找到。

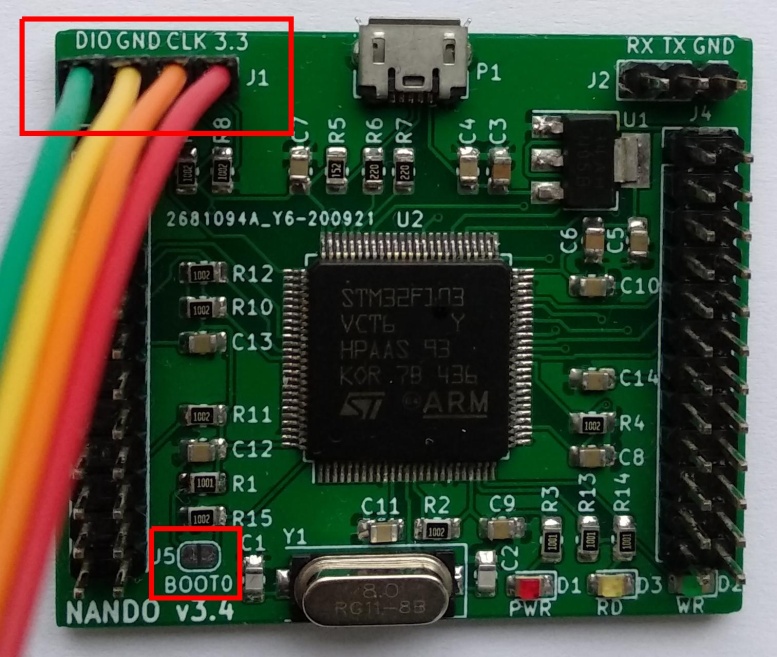


# 6[刻录固件](https://github.com/bbogush/nand_programmer/wiki/BurnFirmware)

STM32 MCU固件可以使用ST-Link或USB-UART适配器进行刻录。STM32F103不支持通过USB编程。

## 6.1 ST-Link （SWD）

设备固件使用 ST-Link V2 刻录。



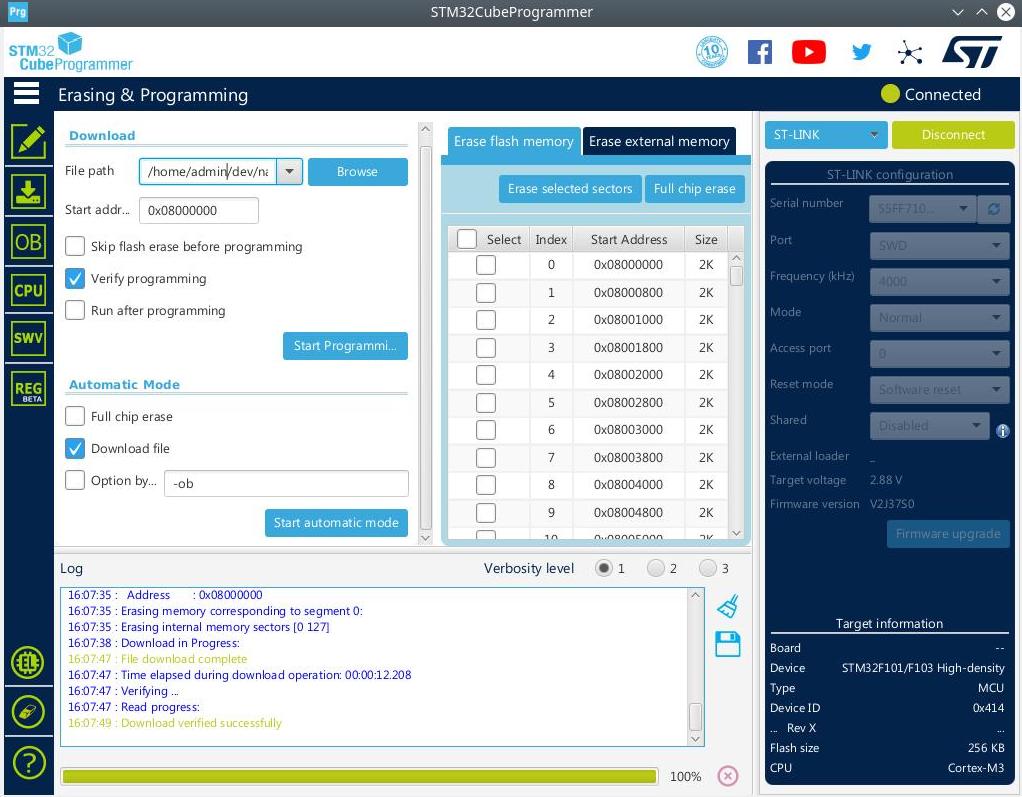
ST-Link连接到电路板的SWD引脚，如下所示。 全新的MCU可能需要使用连接BOOT0引脚上电，有时还需要使用RST引脚（在新PCB中提供）。 编程后，BOOT0 和 RST 引脚必须断开。

## 1）ST-Link Ubuntu软件

* sudo apt-get install libusb-1.0-0-dev
* git clone https://github.com/texane/stlink.git
* cd stlink/
* mkdir -p build && cd build
* cmake -DCMAKE\_BUILD\_TYPE=Release ..
* sudo make install
* sudo cp ../etc/udev/rules.d/\* /etc/udev/rules.d/
* sudo udevadm control --reload-rules
* sudo udevadm trigger
* Connect ST-Link to board.
* Download firmware from Google Driver release directory.
* st-flash write nando\_fw.bin 0x8000000

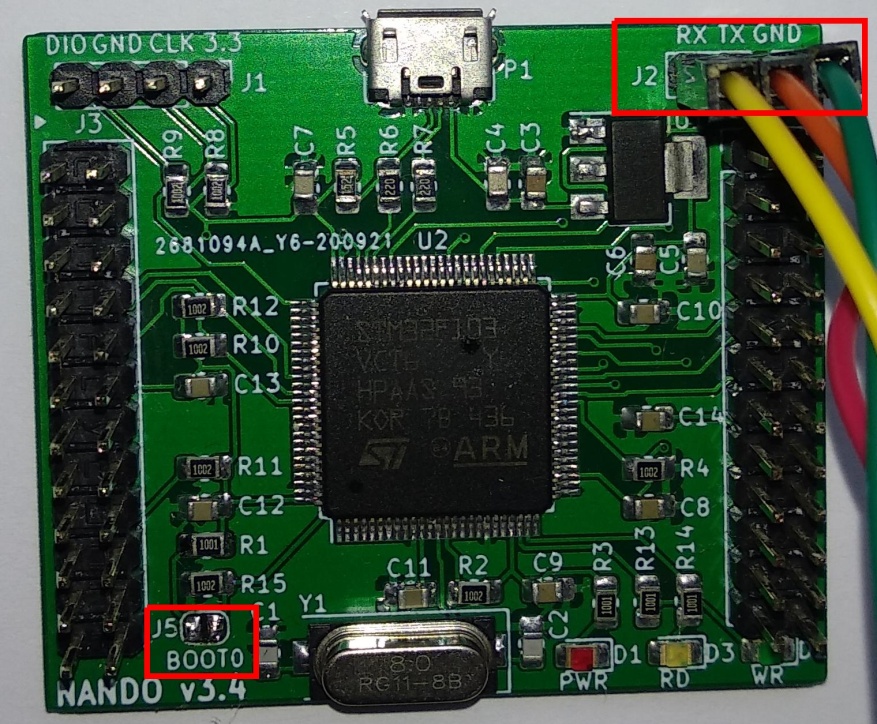
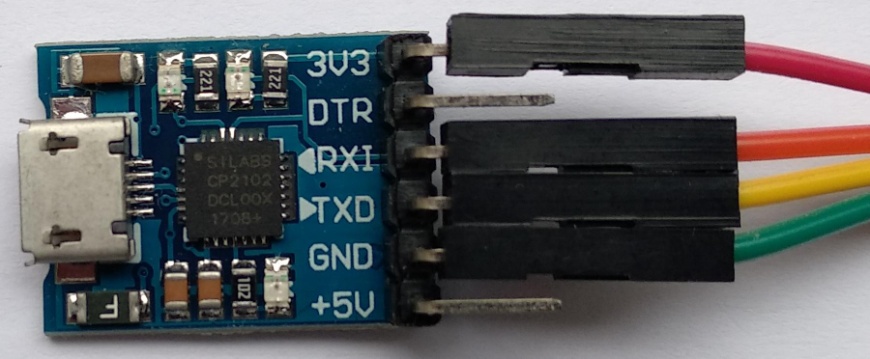
## 2）ST-Link Ubuntu/Windows/Mac STM32Cube Programmer软件

* 从 https://www.st.com 或谷歌驱动程序工具目录下载并安装STM32CubeProgrammer。
* 从谷歌驱动程序发布目录下载固件。
* 使用STM32CubeProgrammer编程器编写固件。



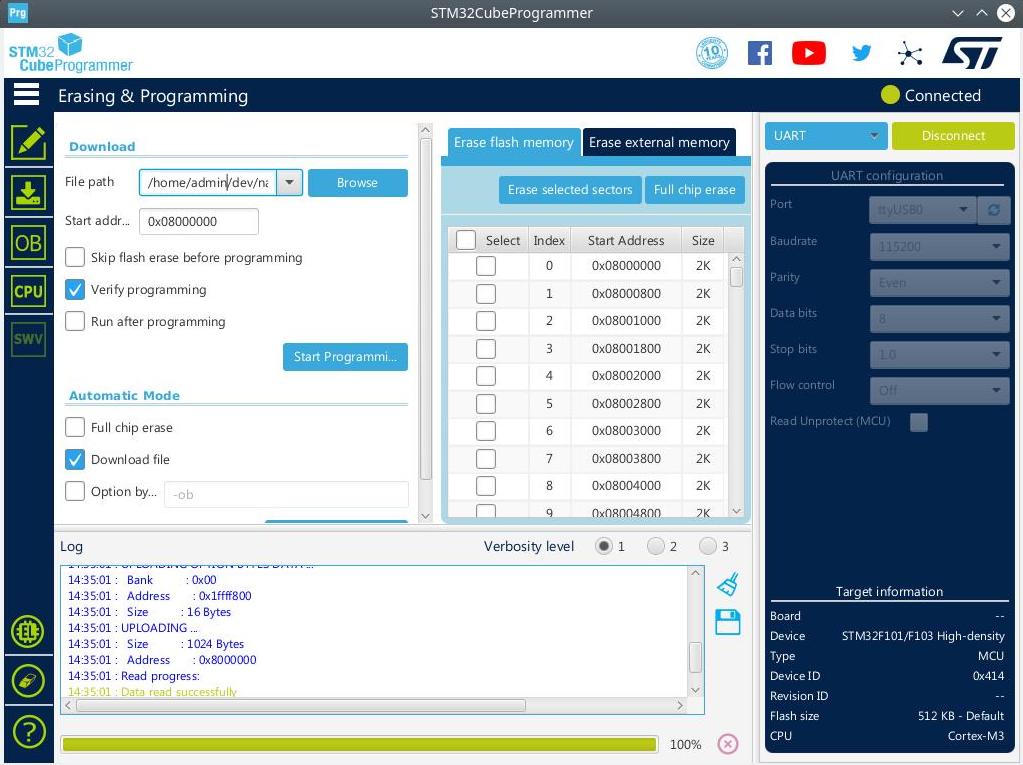
## 6.2 USB-UART 适配器

USB-UART 适配器连接到主板的 UART 引脚，如下所示。电路板需要使用连接BOOT0 引脚上电，有时还需要使用 RST 引脚（在新 PCB 中提供）。 编程后，BOOT0 和 RST 引脚必须断开。

## USB-UART 适配器 Ubuntu/Windows/Mac STM32 Cube Programmer 软件

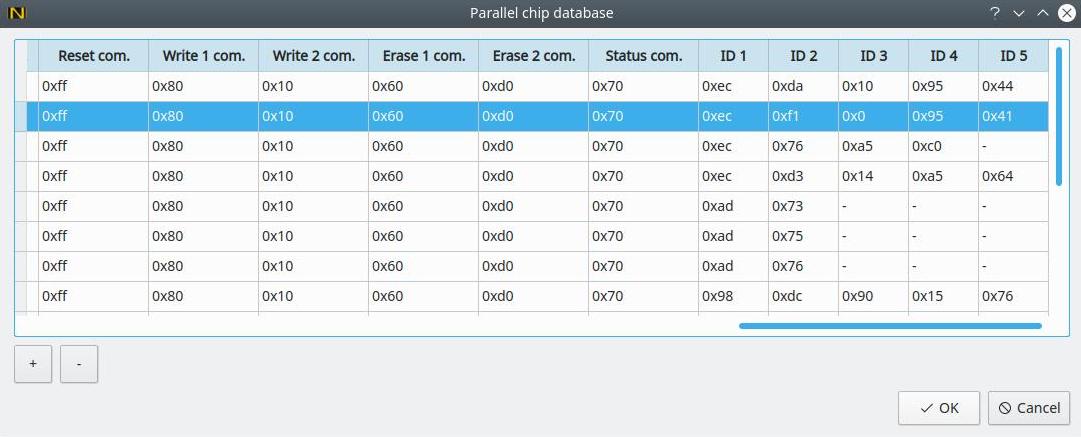
* 从 https://www.st.com 或谷歌驱动程序工具目录下载并安装STM32 Cube Programmer。
* 从谷歌驱动程序发布目录下载固件。
* 使用STM32 Cube Programmer编程器编写固件。
* 在 Linux 上，必须授予 /dev/ttyUSBx 的权限（sudo chmod 777 /dev/ttyUSB0）。

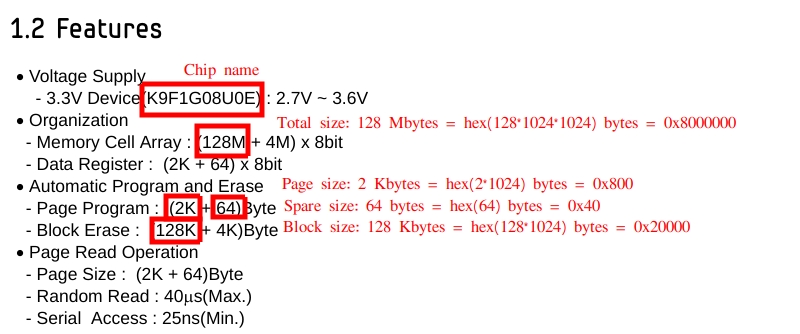


# 7[添加新的并行 NAND 芯片](https://github.com/bbogush/nand_programmer/wiki/AddNANDChip)

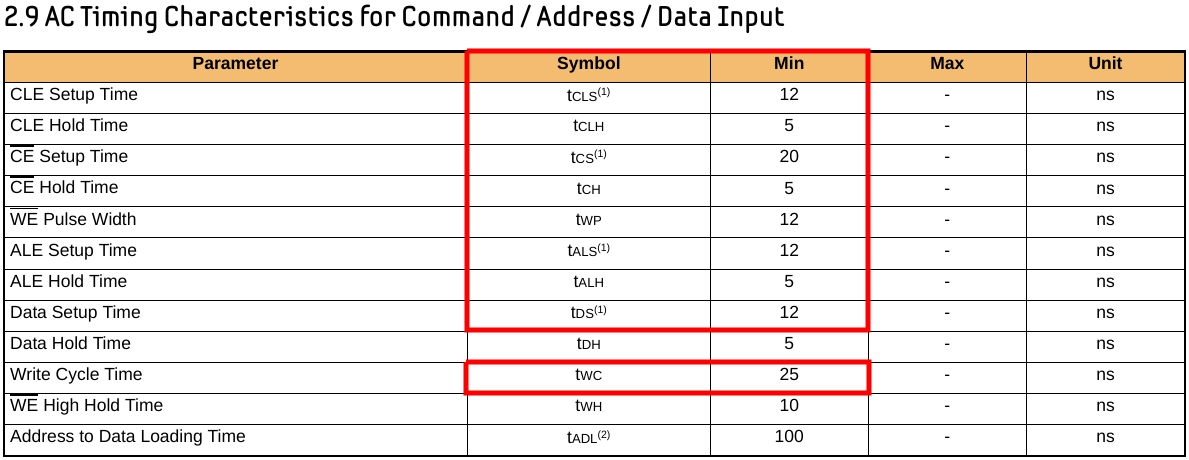
打开*设置 - 并行芯片数据库*，并从数据表添加芯片参数。

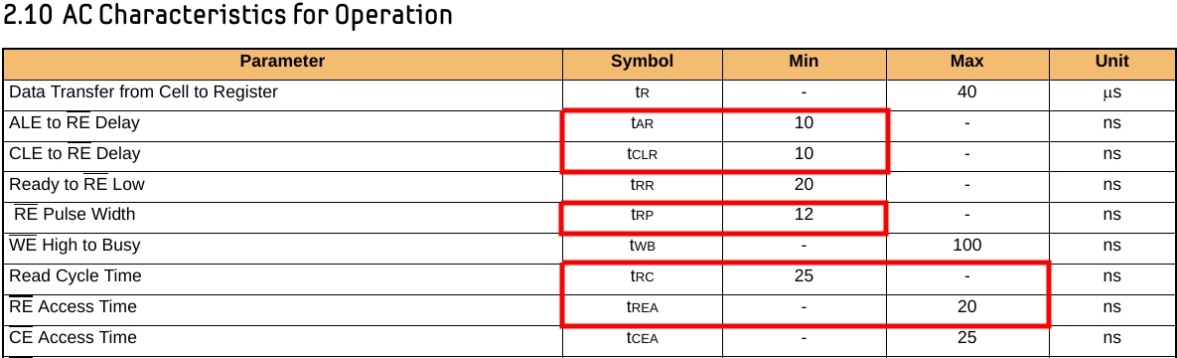
* *名称* - 芯片的名称。
* *页面大小* - 不带空闲区域的页面大小（以字节为单位）（十六进制）。
* *块大小* - 块大小（以字节为单位，不含备用区域）（十六进制）。
* *总大小* - 总大小（以字节为单位），不含备用区域（十六进制）。
* *备用大小* - 以字节为单位的备用大小（十六进制）。
* *BB标记关闭。*- 包含坏块标记的备用区域中字节的偏移量。如果字节包含非0xFF值，则该块被视为坏块。偏移量从 0 开始。
* *tCS*- 芯片使能 （CE） 设置时间 （ns）。
* *tCLS* - 命令锁存器启用 （CLE） 设置时间 （ns）。
* *tALS*- 地址锁锁启用 （ALE） 设置时间 （ns）。
* *tCLR* - 命令锁存器启用 （CLE） 以读取启用 （RE） 延迟 （ns）。
* *tAR*- 地址锁存器使能 （ALE） 到读取使能 （RE） 延迟 （ns）。
* *tWP*- 写使能 （WE） 脉冲宽度 （ns）。
* *tRP-* 读取使能 （RE） 脉冲宽度 （ns）。
* *tDS*- 数据建立时间（纳秒）。
* *tCH*- 芯片使能 （CE） 保持时间 （ns）。
* *tALH*- 地址锁存器使能 （ALE） 保持时间 （ns）。
* *tWC*- 写入周期时间（纳秒）。
* *tRC*- 读取周期时间（纳秒）。
* *tREA*- （RE） 访问时间 （ns）。
* *行周期* - 对行地址进行编程所需的时钟周期数。
* *列周期* - 编程列地址所需的时钟周期数。
* *阅读 1 个。*- 读取页面第一个时钟周期（十六进制）的命令。
* *阅读 2 com。*- 读取第 2 个时钟周期（十六进制）的命令。可选 （-）。
* *阅读 spr.com。*- 读取备用区域（十六进制）的命令。可选 （-）。
* *读取 ID com。*- 读取芯片 ID 的命令（十六进制）。
* *写 1 个。*- 命令对页面进行编程的第一个循环（十六进制）。
* *写 2 个。*- 命令编程页面第二个循环（十六进制）。可选 （-）。
* *擦除 1 com。*- 命令擦除块第一个循环（十六进制）。
* *擦除 2 com。*- 命令擦除块第二个循环（十六进制）。可选 （-）。
* *状态通信*- 状态命令（十六进制）。
* *ID1*- 第一个芯片ID - 制造商（十六进制）。
* *ID2* - 第二个芯片 ID - 设备 ID（十六进制）。可选 （-）。
* *ID3*- 第三芯片ID（十六进制）。可选 （-）。
* *ID4*- 第 4 个芯片 ID（十六进制）。可选 （-）。
* *ID5*- 第 5 个芯片 ID（十六进制）。可选 （-）。

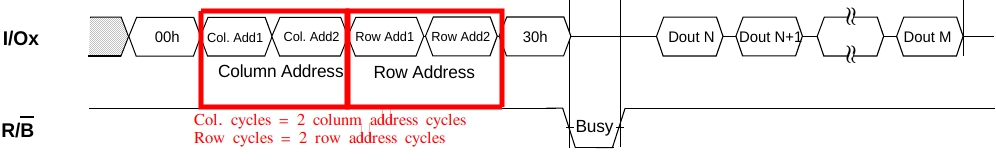
三星 K9F1G08U0E 示例：  


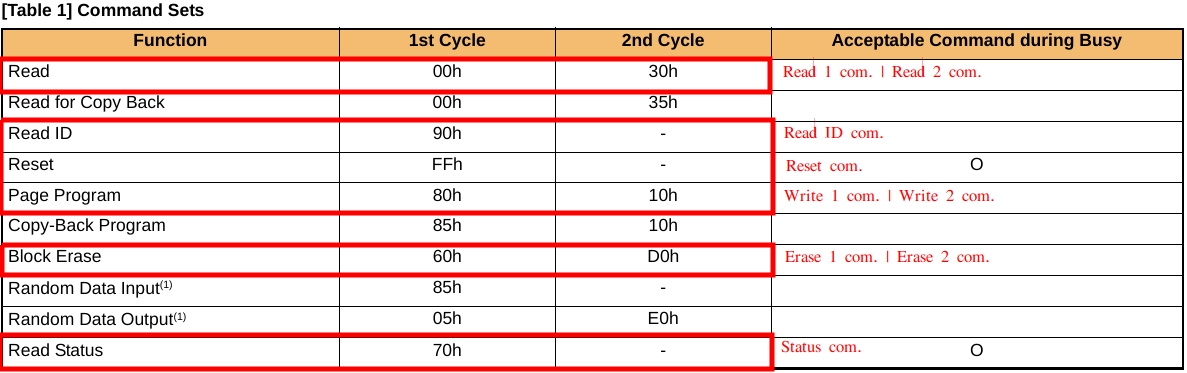


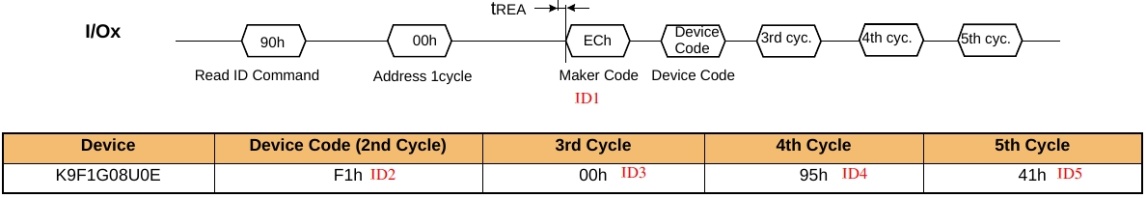








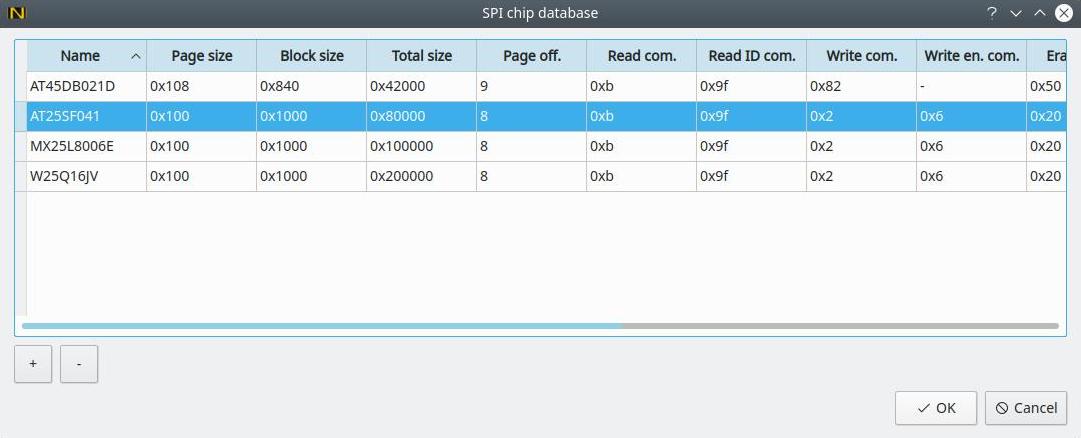


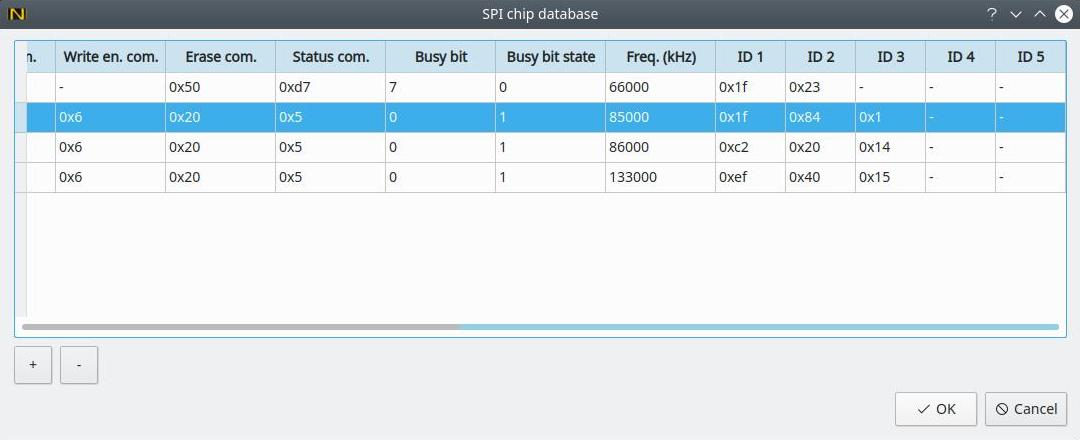


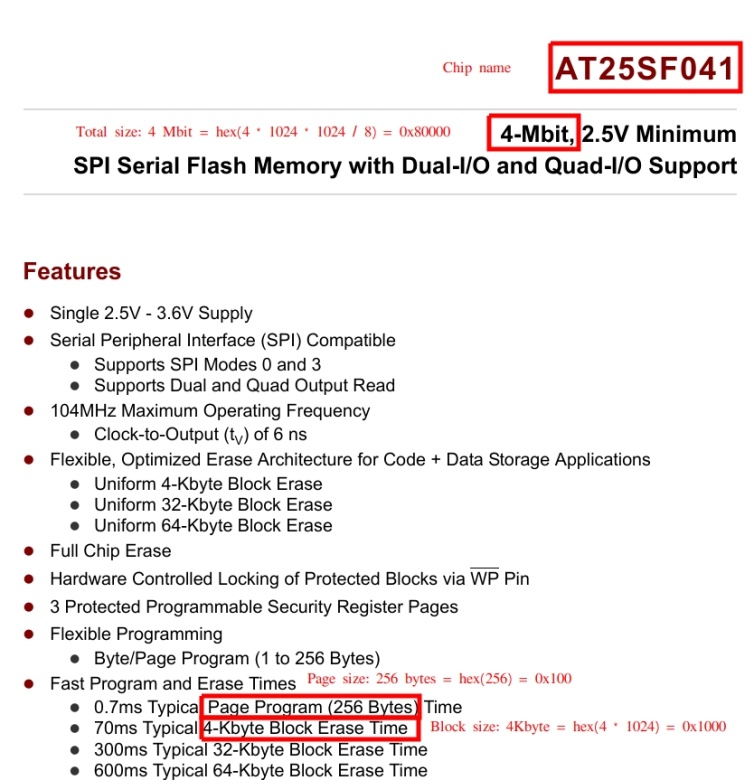
# 8[添加新的 SPI 芯片](https://github.com/bbogush/nand_programmer/wiki/AddSPIChip)

打开*设置 - SPI 芯片数据库*，并从数据表添加芯片参数。

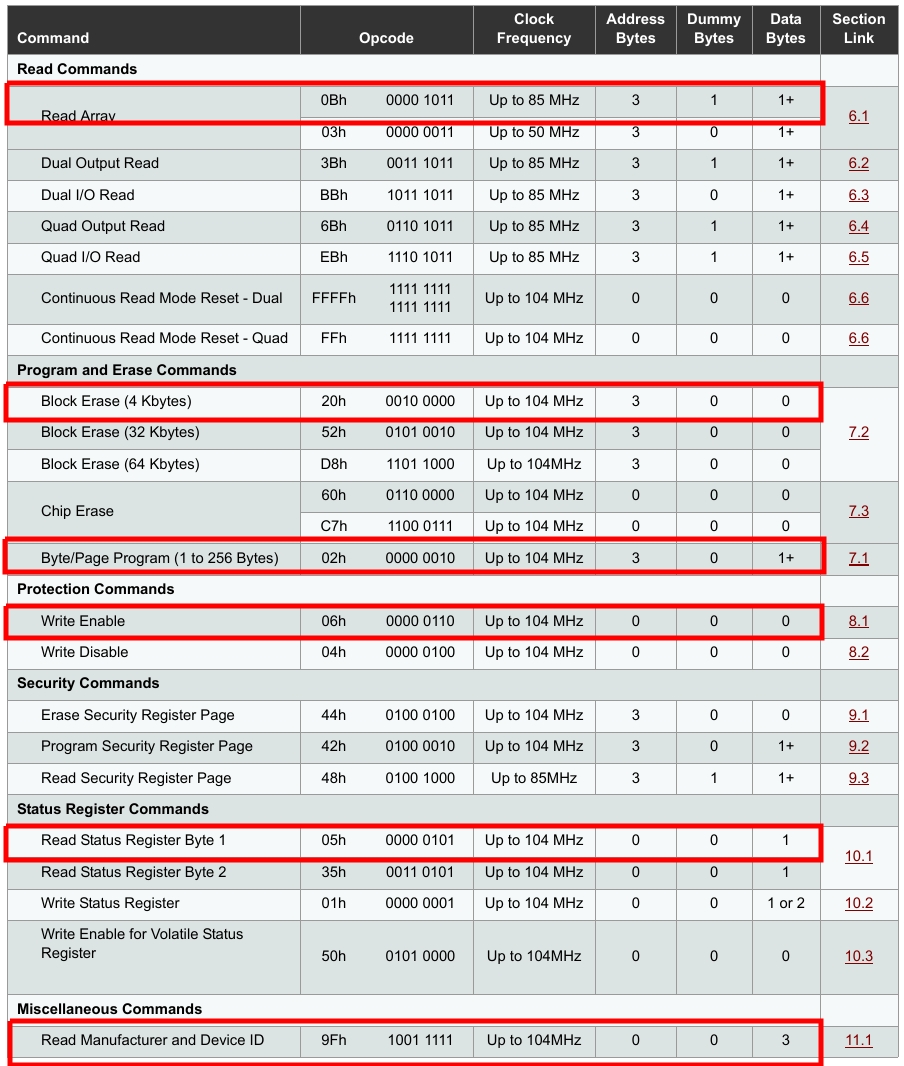
* *名称* - 芯片的名称。
* *页面大小* - 以字节为单位的页面大小（十六进制）。
* *块大小* - 以字节为单位的块大小（十六进制）。
* *总大小* - 以字节为单位的总大小（十六进制）。
* *页面关闭。*- 整个地址中的页面地址偏移量。整个地址由页面地址和字节地址组成。
* *阅读com。*- 读取页面的命令（十六进制）。
* *读取 ID com。*- 命令到芯片 ID（十六进制）。
* *写com。*- 写入页面的命令（十六进制）。
* *写.com。*- 启用页面写入的命令（十六进制）。可选 （-）。
* *擦除通信。*- 擦除块（十六进制）的命令。
* *状态通信*- 读取状态的命令（十六进制）。
* *忙位* - 状态寄存器中忙位的位数（从 0 开始）。
* *忙位状态* - 当芯片处于繁忙状态时，状态寄存器中忙位的状态（0/1）。
* *频率 （kHz）* - 所有命令的最大时钟频率 （kHz）。
* *ID1*- 第一个芯片ID - 制造商（十六进制）。
* *ID2* - 第二个芯片 ID - 设备 ID（十六进制）。可选 （-）。
* *ID3*- 第三芯片ID（十六进制）。可选 （-）。
* *ID4*- 第 4 个芯片 ID（十六进制）。可选 （-）。
* *ID5*- 第 5 个芯片 ID（十六进制）。可选 （-）。

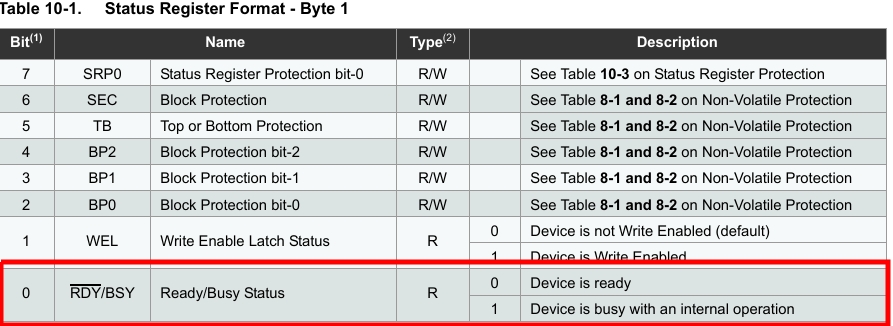
示例三星 AT25SF041：  


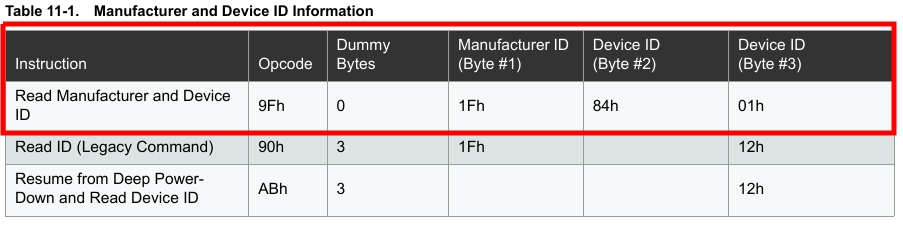






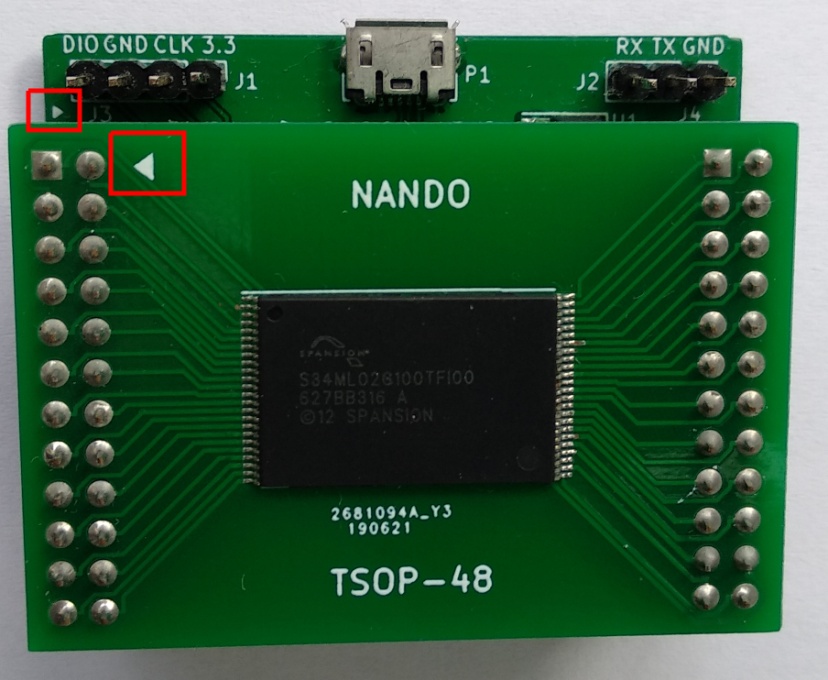


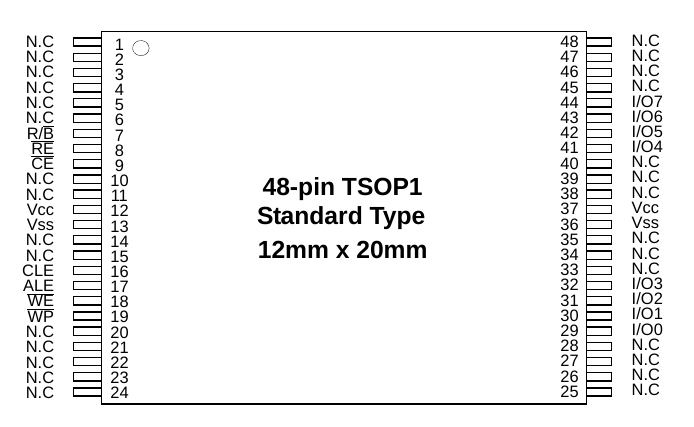




# 9[并行 NAND 芯片连接](https://github.com/bbogush/nand_programmer/wiki/NANDChipConnection)

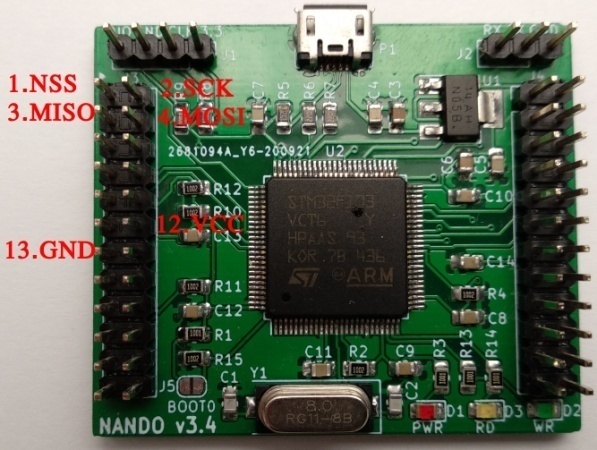
并行NAND芯片可以使用插座适配器或焊接适配器连接到NANDO。焊接适配器提供更可靠的连接。在大多数情况下，简单的电线连接是行不通的。

确保适配器和板卡的按键匹配：  
目前NANDO支持3.3V 8位并行NAND，具有以下引脚排列：

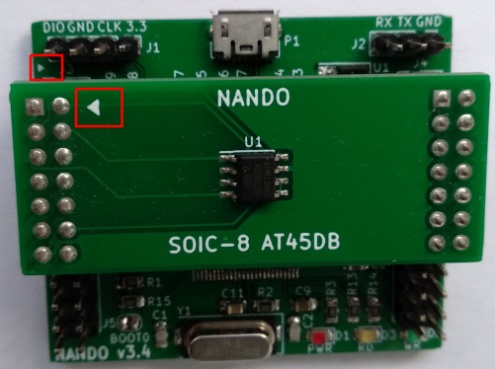
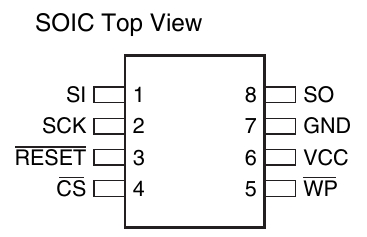


# 10 [SPI 芯片连接](https://github.com/bbogush/nand_programmer/wiki/SPIChipConnection)

NANDO支持3.3V SPI芯片编程。芯片可以通过电线或使用适配器连接，根据以下架构

：确保RESET和WP连接到VCC。

目前，一种焊接适配器可用于以下引脚排列：

# 11[编程器设置](https://github.com/bbogush/nand_programmer/wiki/Settings)

* USB 设备名称 - 操作系统提供给程序员的串行 *USB 设备*的名称（/tty/USBx、COMx）。
* *跳过坏块* - 跳过坏块的读/写/擦除。坏块在读/写/擦除操作之前为芯片读取一次。 如果特定页面备用区域字节包含非0xFF值，则坏块被视为坏块。
* *包括备用区域* - 读/写页面备用区域以及主要区域。

# 12[编程器操作](https://github.com/bbogush/nand_programmer/wiki/Operations)

* 读取 ID-*读取芯片 ID*，包括制造商、设备和其他信息。
* *擦除* - 全芯片擦除。
* *读取* - 全芯片读取。
* *将*缓冲区写入芯片。
* *读取坏块* - 读取所有坏块。

# 13[软件构建](https://github.com/bbogush/nand_programmer/wiki/Build)

## 13.1要求

Linux x32/x64， Windows x64.

## 13.2固件

### 1）Linux

* 转到[谷歌司机](https://drive.google.com/drive/folders/1zLtebfLW6CxaWMq0rr4EfEPmeXfGk-7n)。
* 按“全部下载”。
* 姆克迪尔 ~/dev/
* cd ~/dev/
* mv ~/下载/nand\_programmer\*.zip ./
* 解压缩nand\_programmer-\*.zip
* 光盘nand\_programmer/编译器/
* tar -xvf gcc-arm-none-eabi-4\_9-2015Q1-20150306-linux.tar.bz2
* 光盘..
* Git 克隆 https://github.com/bbogush/nand\_programmer.git
* 光盘nand\_programmer/固件
* 做

### 2）Windows

* 转到[developer.arm.com](https://developer.arm.com/-/media/files/downloads/gnu-rm/10-2020q4/gcc-arm-none-eabi-10-2020-q4-major-win32.exe?revision=9a4bce5a-7577-4b4f-910d-4585f55d35e8&rev=9a4bce5a75774b4f910d4585f55d35e8&hash=9770A44FEA9E9CDAC0DD9A009190CC8B)。
* 下载并安装编译器（gcc-arm-none-eabi-10-2020-q4-major-win32.exe）和[make](https://zenlayer.dl.sourceforge.net/project/gnuwin32/make/3.81/make-3.81.exe?viasf=1)（make-3.81.exe）。
* 将已安装二进制文件（arm-none-eabi-gcc.exe，make.exe）的路径添加到PATH环境变量。
* Git 克隆 https://github.com/bbogush/nand\_programmer.git
* 光盘nand\_programmer/固件
* 做

## 13.3主机应用程序

### 1)Linux

* sudo apt-get install libboost-all-dev mesa-common-dev libglu1-mesa-dev
* 安装 Qt5.x
* 使用 QtCreator 打开 ~/dev/nand\_programmer/nand\_programmer/qt/qt.pro。
* 转到“项目”->“生成->生成步骤”->“添加生成步骤->生成”。将“安装”添加到“制作参数”。
* 构建>运行 qmake。
* 构建>全部构建。
* 构建>运行。

### 2) Windows

* 在 C：/boost/ 下从[GoogleDriver](https://drive.google.com/drive/u/0/folders/15pFG1sRmd8cxjD9x2B9aag1P6Wlx8U34)存档中安装 [boost](https://archives.boost.io/release/1.82.0/source/boost_1_82_0.zip) 库 （boost.zip） 或编译。[Boost 在 Windows 上的入门指南 - Boost C++ 函数库](https://boost.ac.cn/doc/libs/latest/more/getting_started/windows.html)
* 安装 Qt5.x。
* 使用 QtCreator 打开 ~/dev/nand\_programmer/nand\_programmer/qt/qt.pro。
* 转到“项目”->“生成->生成步骤”->“添加生成步骤->生成”。将“安装”添加到“制作参数”。
* 构建>运行 qmake。
* 构建>全部构建。
* 构建>运行。

# 14[印刷电路板变更 （KiCad）](https://github.com/bbogush/nand_programmer/wiki/KiCad)

.