嵌入式系统与接口技术
（信工电科教学班）
实验报告

实验 4　UART BootLoader 实验

成员姓名：强陶

完成时间：2024 年 5 月 29 日

上海交通大学 电子信息与电气工程学院

目　　录

1. 实验目的

了解 BootLoader 的工作原理。

能够利用 TI 的相关例程进行 BootLoader 及应用 APP 的操作。

最终能够将自己的 APP 项目修改为能够被 BootLoader 调用的应用 APP。

2. 系统总述

LM Flash Programmer_1613，是一款免费的闪存编程实用程序，与 Texas Instruments Stellaris 或 Tiva C 系列微控制器一起使用。通过 bootloader 下载更新固件需要用到 LMFlashProgrammer。

TivaWare_C_Series-2.2.0.295，TI TivaWare C 系列评估、开发和演示软件

软件包安装完成后例程位于 SW-TM4C-2.1.4.178\examples\boards\ek-tm4c1294xl\ 下。与 bootloader 相关的有五个目录，分别对应串口更新与网口更新两个部分，对应串口的是 boot_serial，boot_demo1 和 boot_demo2。其中 boot_serial 是 bootloader 本身，boot_demo1 和 boot_demo2 是两个跑马灯例程，一个闪 LED1，一个闪 LED2，主要就是演示下载不同固件用的。

名称	项目位置	说明
Boot_serial	C:\ti\TivaWare_C_Series-2.2.0.295\examples\boards\ek-tm4c1294xl\boot_serial	Bootloader 程序
Boot_demo1	C:\ti\TivaWare_C_Series-2.2.0.295\examples\boards\ek-tm4c1294xl\boot_demo1	闪烁 D1 应用程序
Boot_demo2	C:\ti\TivaWare_C_Series-2.2.0.295\examples\boards\ek-tm4c1294xl\boot_demo2	闪烁 D2 应用程序

三个项目中，在各自的子目录.\rvmdk 下，分别有如下三个 bin 文：Boot_serial.bin、Boot_demo1.bin、Boot_demo2.bin

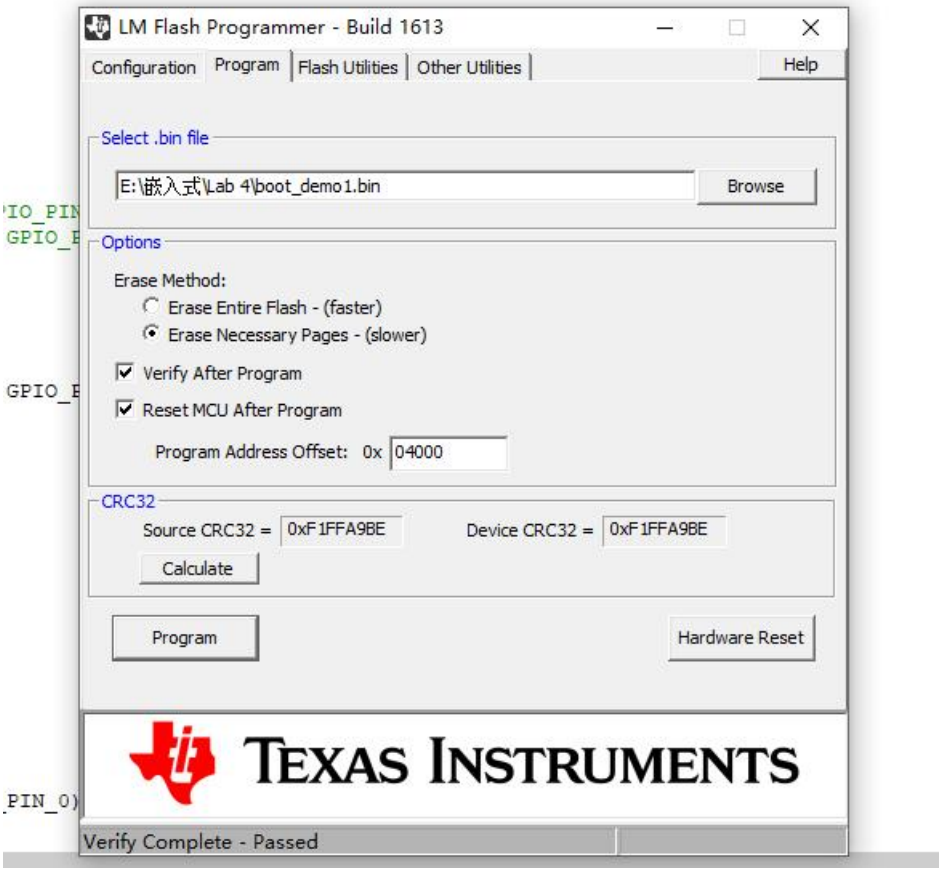
axf 文件、hex 文件与 bin 文件都是可以运行在我们的芯片上的，它们都存储了编译器根据源代码生成的机器码，根据应用场合的不同，它们又有所区别：

axf 文件：包含调试信息。axf 文件是编译默认生成的文件，不仅包含代码数据，而且还包含着调试信息，在 MDK 里进行 debug 调试用的就是这个文件。

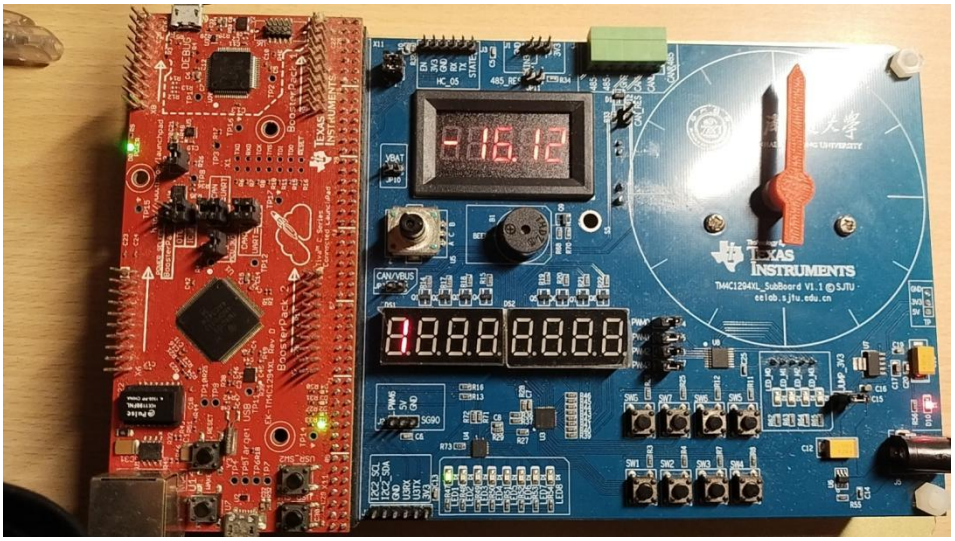
hex 文件：包含地址信息。hex 文件是一种使用十六进制符号表示的代码记录， 记录了代码应该存储到 FLASH 的哪个地址，下载器可以根据这些信息辅助下载。

bin 文件：最直接的代码映像。bin 文件就是最小的可以运行的文件了，其包含最直接的代码映像。

3. 实验思路与结果

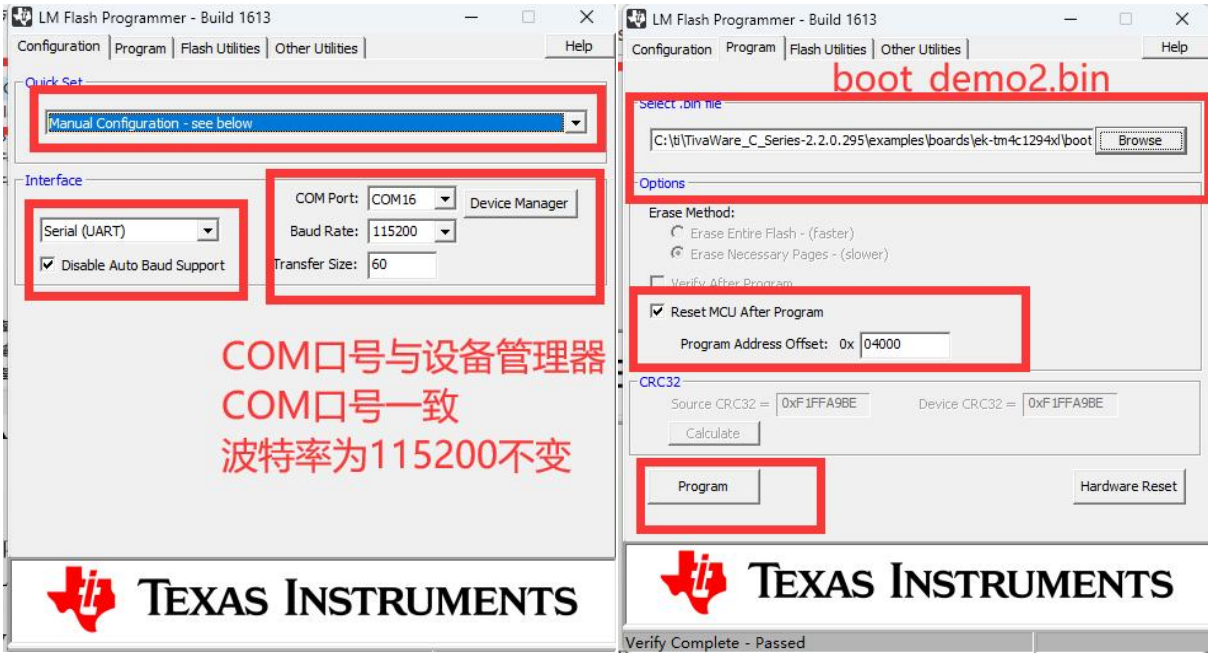


将地址设为 0x04000,选中 boot_demo1.bin



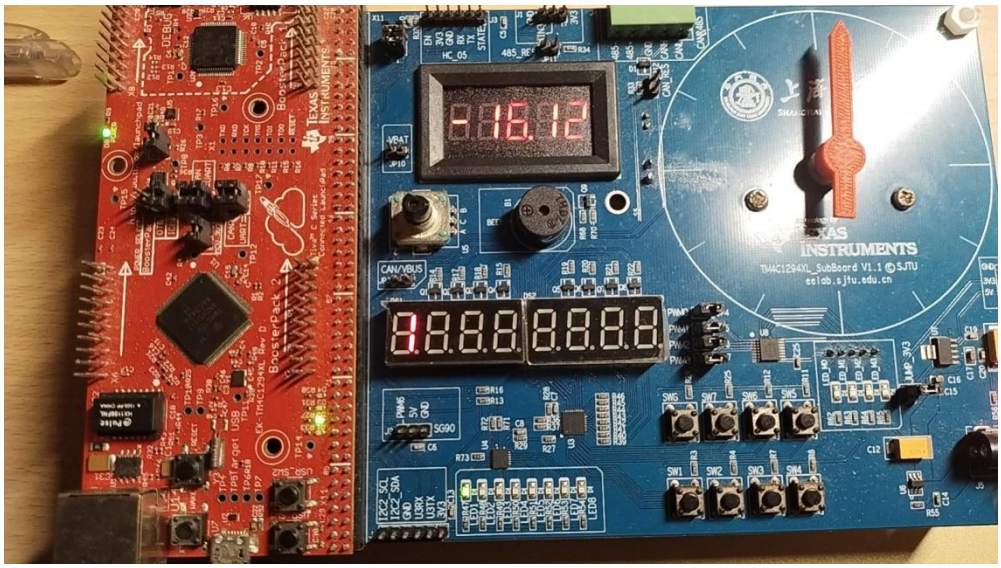
图一、D1 闪烁

boot_serial 和 boot_demo1 分别烧入 0x0 和 0x4000 开始的位置。系统复位后，先进入 boot_serial，因为没有按下 SW1，因此 bootloader 程序自动跳转到预置的 0x4000 处执行 boot_demo1 程序。

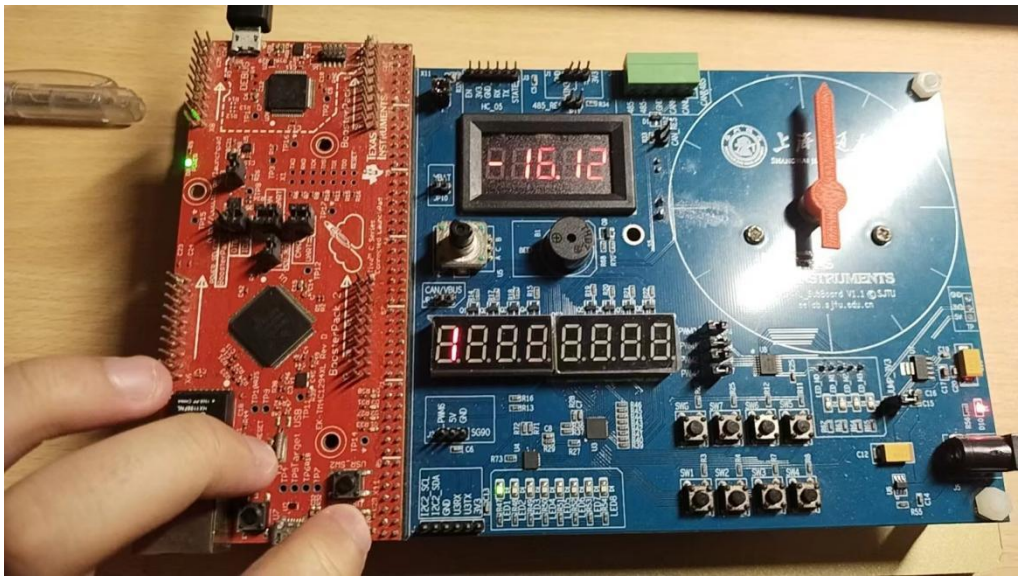


在 Program 前，按住 SW1 按键，然后再按 Program 键。强置 bootloader 程序进行 UART 通讯，将 boot_demo2.bin 文件通过 UART 传送到 MCU 中。然后按一下 LaunchPad 板上的 RESET 按键，进行复位。就能看到 D2 闪烁。

如果在按下 RESET 时，同时按下 SW1 键，则进入 Bootloader 状态，并且不会跳转到 boot_demo2，因此没有 LED 闪烁。程序始终在 bootloader 中。此时如果需要跳转到 boot_demo2，则需要再按一次 RESET 复位即可。当没有 SW1 按下时，程序自动从 bootloader 跳转到应用 APP 中。即 boot_demo2。



图二、D2 闪烁



图三、处于 Bootloader 状态

4. 讨论题

请参考 boot_demo2 项目，将你自己的项目修改为适合 boot_serial 调用的应用 APP。注意如下 两点即
1、 偏移地址为 0x4000
2、 需要使用 fromelf 将 axf 文件生成为 bin 文件供调用。

1: 修改偏移地址为 0x4000

(1). 打开的 Keil 项目。

(2). 选择 "Options for Target 'Target 1'"（或你的目标名称）。

(3). 在 "Target" 选项卡中，找到 "IROM1" 的配置，设置起始地址为 0x4000，并相应调整大小。可以设置为：

Start: 0x4000

Size: 0x7C000 (512KB - 16KB)

2: 使用 fromelf 将 axf 文件生成为 bin 文件

(1). 打开 Keil 的命令行工具（Command Prompt）或者在 Windows 的命令提示符中，导航到你的项目输出目录。

(2). 使用 fromelf 工具将 .axf 文件转换为 .bin 文件:

fromelf --bin --output your_project.bin your_project.axf

1. 打开 Keil，加载项目 MyApp。

2. 进入 Options for Target 'MyApp'，在 Target 选项卡中，设置如下：

- IROM1
  - Start: 0x4000
  - Size: 0x7C000

3. 编译项目，生成 MyApp.axf 文件。

4. 打开命令提示符，导航到项目输出目录。

5. 将 .axf 文件转换为 .bin 文件：

C:\Keil_v5\ARM\ARMCC\bin\fromelf --bin --output MyApp.bin MyApp.axf

5. 感想与收获

1、Bootloader 的原理：

Bootloader（引导加载程序）是一段在系统启动时运行的代码，用于加载操作系统或应用程序。它的主要功能是在设备上电或复位时初始化硬件，加载和执行应用程序或操作系统。

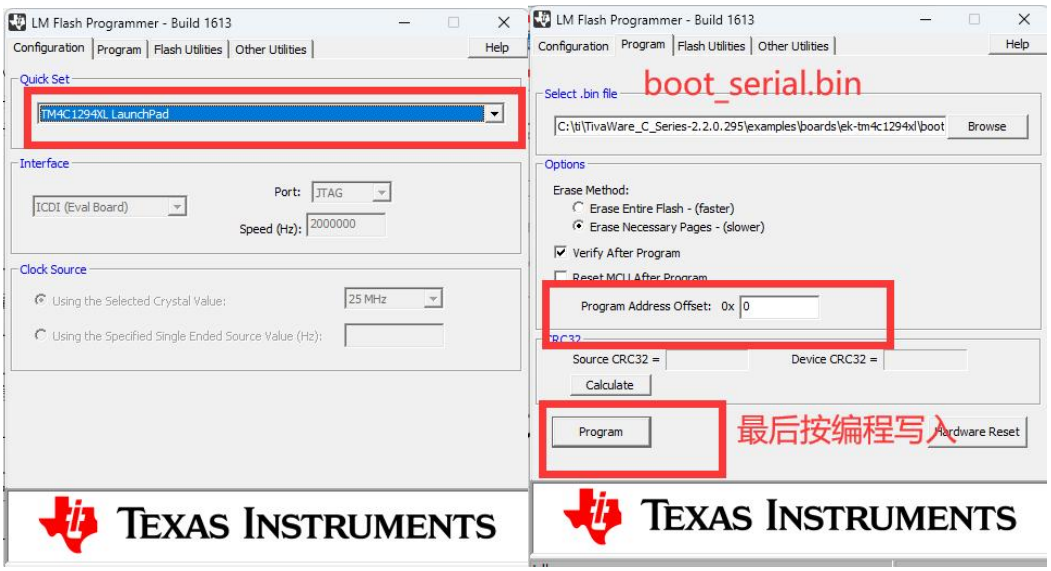
(1).启动硬件初始化：当设备上电或复位时，处理器会从一个预定义的地址开始执行代码。这段代码通常位于只读存储器（如 ROM 或闪存）中，称为 BootLoader 的第一阶段（Stage 1）。

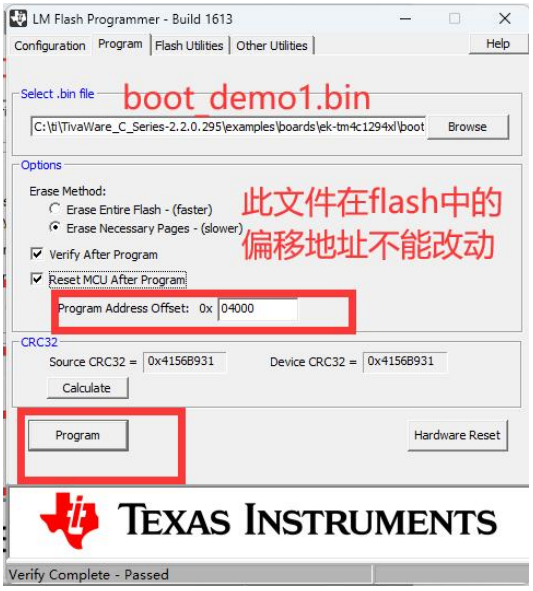
在这一阶段，BootLoader 执行最基本的硬件初始化，例如设置时钟、配置堆栈指针、初始化内存控制器等，以便系统能够正常工作。

(2).检查和选择启动源：BootLoader 通常会检查一些启动源（如闪存、SD 卡、网络等），以确定从哪里加载下一阶段的代码。有些 BootLoader 还会根据特定条件（如引导引脚状态、配置寄存器等）来选择不同的启动路径。

(3).加载和执行应用程序：BootLoader 从预定的存储位置（如闪存中的特定偏移地址）加载应用程序代码到内存中。这一过程可能包括解压缩和校验代码的完整性。一旦应用程序加载到内存中，BootLoader 将跳转到应用程序的入口地址，开始执行应用程序的代码。

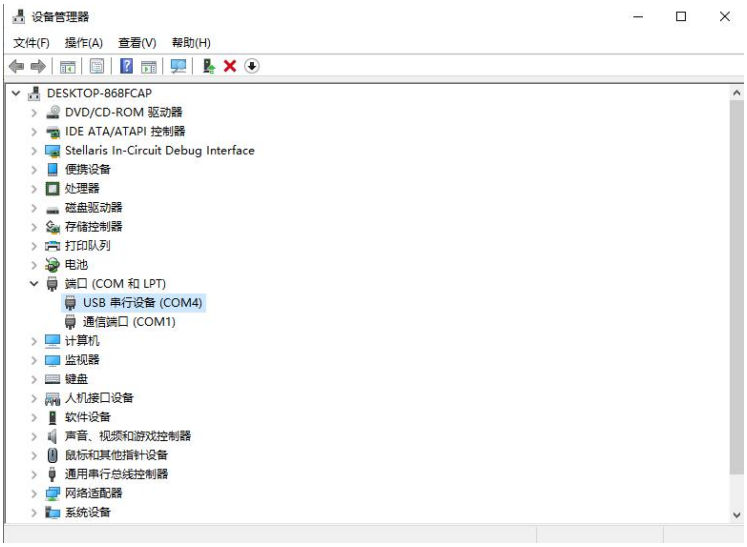
2、学会了程序写入的操作：



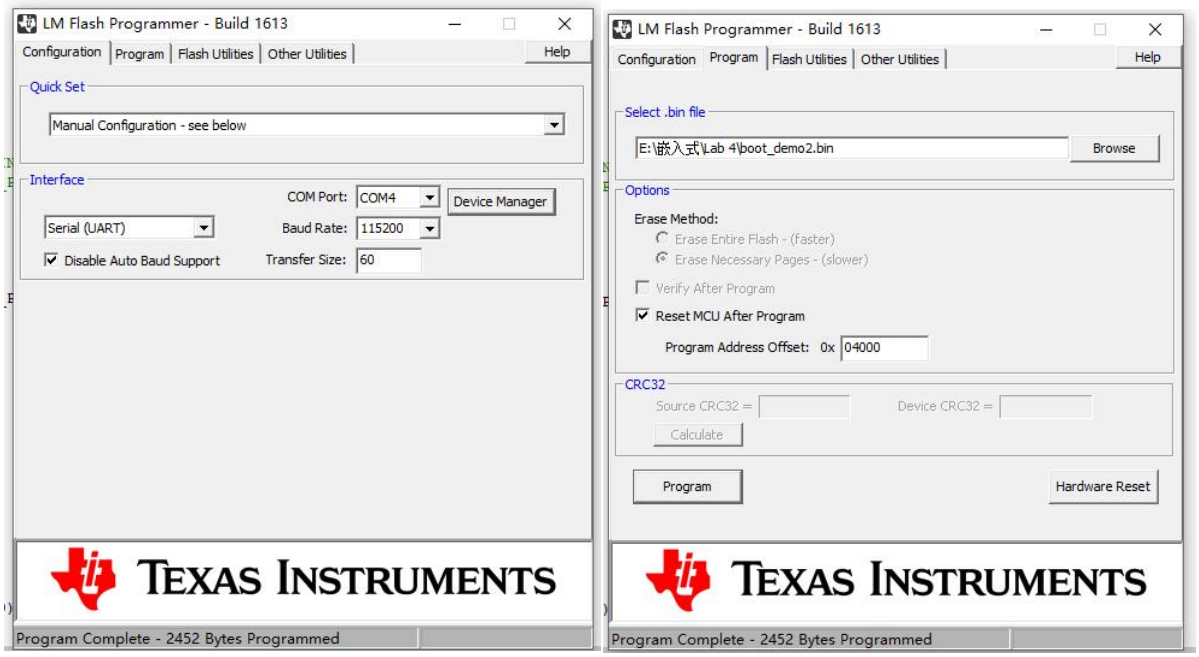


将已有的 bin 文件录入到板子中并正常运行。

3、将 bin 文件通过 UART，利用 BootLoader 写入到 FLASH 的 0x4000 地址：



在虚拟机中 USB 串口对应的端口号为 COM4



在 Program 前，按住 SW1 按键，然后再按 Program 键。强置 bootloader 程序进行 UART 通讯，将 boot_demo2.bin 文件通过 UART 传送到 MCU 中。然后按一下 LaunchPad 板上的 RESET 按键，进行复位。就能看到 D2 闪烁。

如果在按下 RESET 时，同时按下 SW1 键，则进入 Bootloader 状态，并且不会跳转到 boot_demo2，因此没有 LED 闪烁。程序始终在 bootloader 中。此时如果需要跳转到 boot_demo2，则需要再按一次 RESET 复位即可。当没有 SW1 按下时，程序自动从 bootloader 跳转到应用 APP 中。即 boot_demo2。

从而便可手动载入 bin 文件。