

## 介绍

本应用笔记介绍如何将 ACM32F0/FP0 的 MDK 工程移植到 GCC 编译环境。这里选取 UART 工程作为示例。

# 1. 移植前的准备工作

## 1.1. Windows 下建立 GCC 环境

Windows 下使用 GCC, 需要搭建 GUN 环境。这里我选用 MinGW, 下载地址: [www.mingw.org](http://www.mingw.org)。下载完成后安装。然后在 cmd 命令行里输入 `gcc -v`, 出现如下版本信息说明安装成功。

```
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=c:/mingw/bin/./libexec/gcc/mingw32/6
Target: mingw32
Configured with: ../src/gcc-6.3.0/configure --build=x86_64-w
--with-mpfr --with-mpc=/mingw --with-isl=/mingw --prefi
generic --enable-languages=c,c++,objc,obj-c++,fortran,ada
able-shared --enable-threads --with-dwarf2 --disable-sjlj
onv-prefix=/mingw --with-libintl-prefix=/mingw --enable-l
Thread model: win32
gcc version 6.3.0 (MinGW.org GCC-6.3.0-1)
```

## 1.2. 准备 STM32 的 GCC 工程

利用 STM32Cube 生成 STM32F003 的 GCC 工程, 我们的移植很多地方需要参考 ST 的工程。

## 1.3. 几点说明

- 1) 我们实际使用的 GCC 编译器是 `arm-none-eabi-gcc`, 属于 GCC 的一个分支;
- 2) 不同的编译器, 内核文件 `cm33.h` 所依赖的头文件不一样, 在 MDK 下依赖 `cmsis_armclang.h`, 在 GCC 平台依赖 `cmsis_gcc.h`;
- 3) GCC 平台主要修改的有 `makefile` 文件、`.ld` 文件和 `.s` 文件。

## 2. 移植过程

新建一个文件夹，将原有的 MDK 平台下的 UART 工程拷贝到该文件夹下，编译通过后删除 MDK\_Project 文件夹，只保留需要的源文件。然后按照以下步骤移植：

- 1) 拷贝 ST 工程下 CMSIS 文件夹下的 cmsis\_gcc.h 文件到 UART 工程的 CoreDriver/CMSIS 文件夹下，gcc 编译器需要依赖这个文件；
- 2) 参考 ST 的.s 文件修改 ACM32F0 的启动文件，主要是修改中断向量表以及去除一些不同项；
- 3) 将 ST 工程下的 makefile 文件和.ld 文件拷贝到 UART 工程根目录下；
- 4) 修改 makefile 文件：

C\_SOURCES: 源文件路径

ASM\_SOURCES: 启动文件路径

CPU: 所用 CPU 的内核

C\_INCLUDES: 头文件路径

LDSCRIPT: 链接脚本 ld 文件路径

另外需要注意 LIBS = -lc -lm -lnosys 不能删除，这句话表明链接了一些标准库，删除编译会报错；

- 5) 修改 ld 文件（可以理解为 MDK 下的 Sct 文件）：主要是修改堆栈信息；
- 6) 修改 UART 工程主时钟为 64M,关闭 EFlash；
- 7) 修改 HAL\_UART.c,gcc 里使用 printf 需要依赖 write 函数而不是 fputc；

```
int _write(int fd, char *ptr, int len)
{
    HAL_UART_Transmit(&Uart_Debug, (uint8_t*)ptr, len, 0xFFFF);
    return len;
}
```

- 8) 然后开始编译，使用 make 命令进行编译，如果看到生成.bin 等文件说明编译成功，如果报错需要针对具体的错误提示修改。

```
rm-none-eabi-gcc build/main.o build/APP.o build/System_ACM32F0x0.o build/HAL_DMA.o build/HAL_GPIO.o build/HAL
ld/startup_ACM32F0.o -mcpu=cortex-m0 -mthumb -specs=nano.specs -TACM32F0_FLASH.ld -lc -lm -lnosys -Wl,-Ma
32F0_UART.map,--cref -Wl,--gc-sections -o build/ACM32F0_UART.elf
rm-none-eabi-size build/ACM32F0_UART.elf
text    data    bss     dec     hex filename
11224   156     4020   15400   3c28 build/ACM32F0_UART.elf
rm-none-eabi-objcopy -O ihex build/ACM32F0_UART.elf build/ACM32F0_UART.hex
rm-none-eabi-objcopy -O binary -S build/ACM32F0_UART.elf build/ACM32F0_UART.bin
:\E11sion\MyJob\Job_branch\GCC\ACM32F0\GCC_UART>_
```

- 9) 使用 boot 上位机工具下载生成的 bin 文件，观察是否能正常打印信息以及通过 PA8 观察时钟输出是否

正常。

如果时钟没有输出或者串口通讯不正常，需要检查.s文件和.ld文件是否配置正确。

### 3. 静态库生成

在 MDK 平台下我们使用的是.lib 库，GCC 平台对应使用的是.a 库，所有我们需要将现有的 HAL\_EFlash\_EX.lib 转化为对应的.a 库。这里提供一种比较快捷的方法，借助 keil 的 GCC 环境生成.a 库。具体步骤如下：

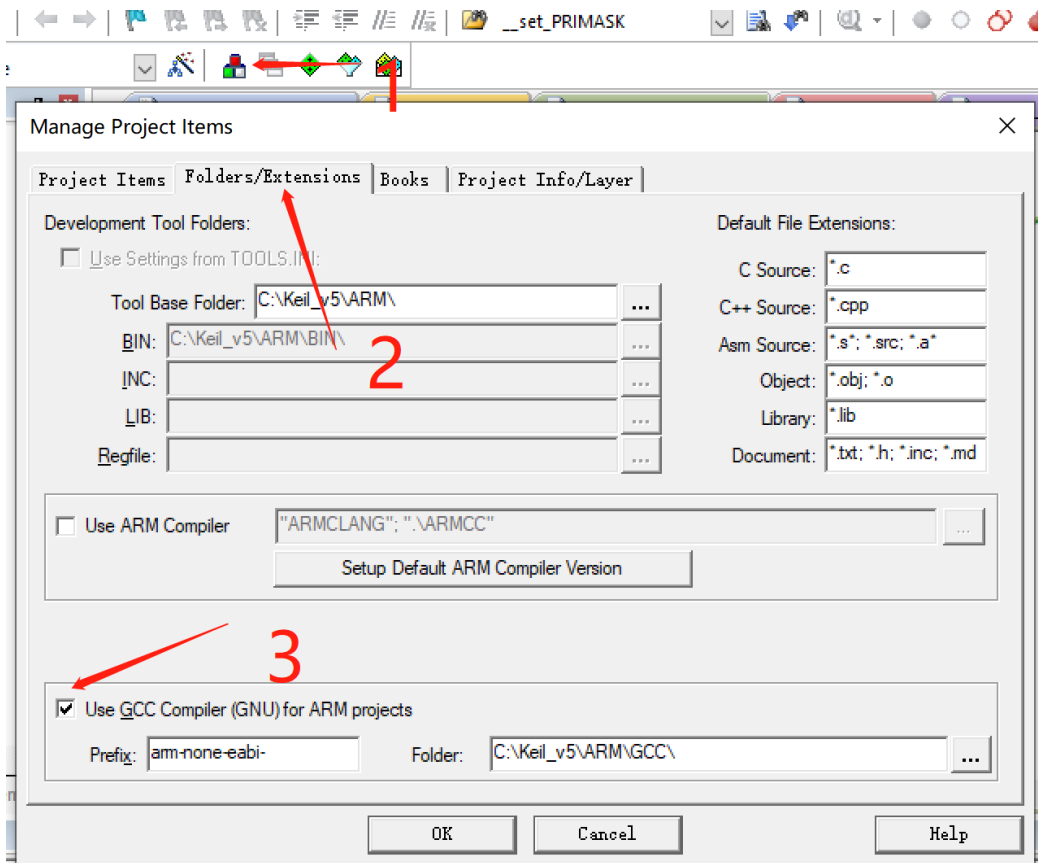
- 1) 下载 ARM\_GCC 编译器，地址：<https://launchpad.net/gcc-arm-embedded/>

解压后将文件夹移动至 keil 安装路径下的 ARM,并重命名为 GCC;

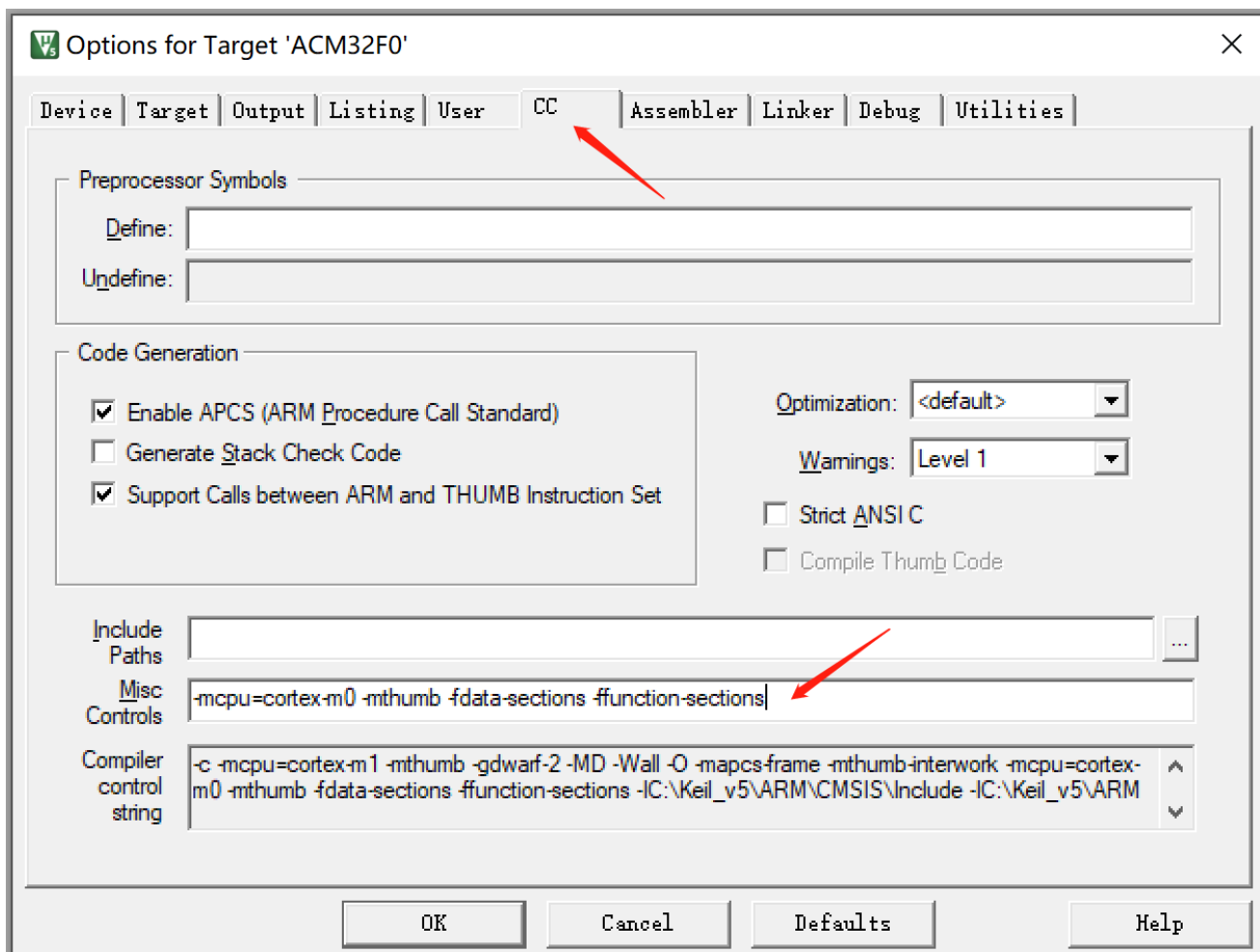
TCAMP (C:) > Keil\_v5 > ARM > GCC

名称	修改日期	类型	大小
arm-none-eabi	2020/11/24 12:04	文件夹	
bin	2020/11/24 12:08	文件夹	
lib	2020/11/24 11:56	文件夹	
share	2020/11/24 12:08	文件夹	

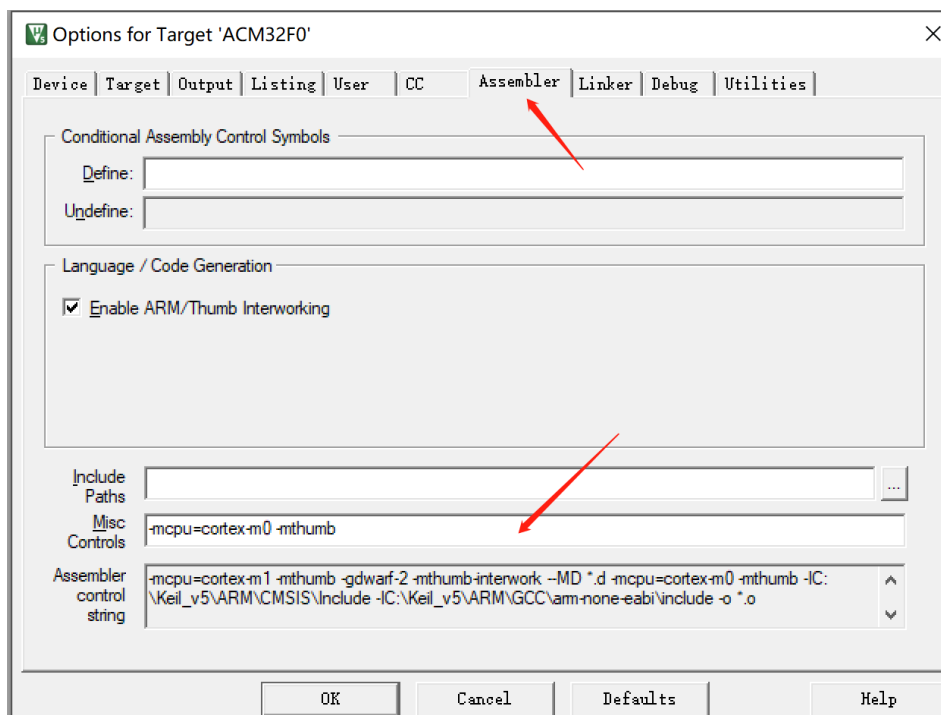
- 2) 打开对应的 MDK 工程，按下面步骤打开并选择 GCC 编译器，配置编译器路径为上一步的路径；



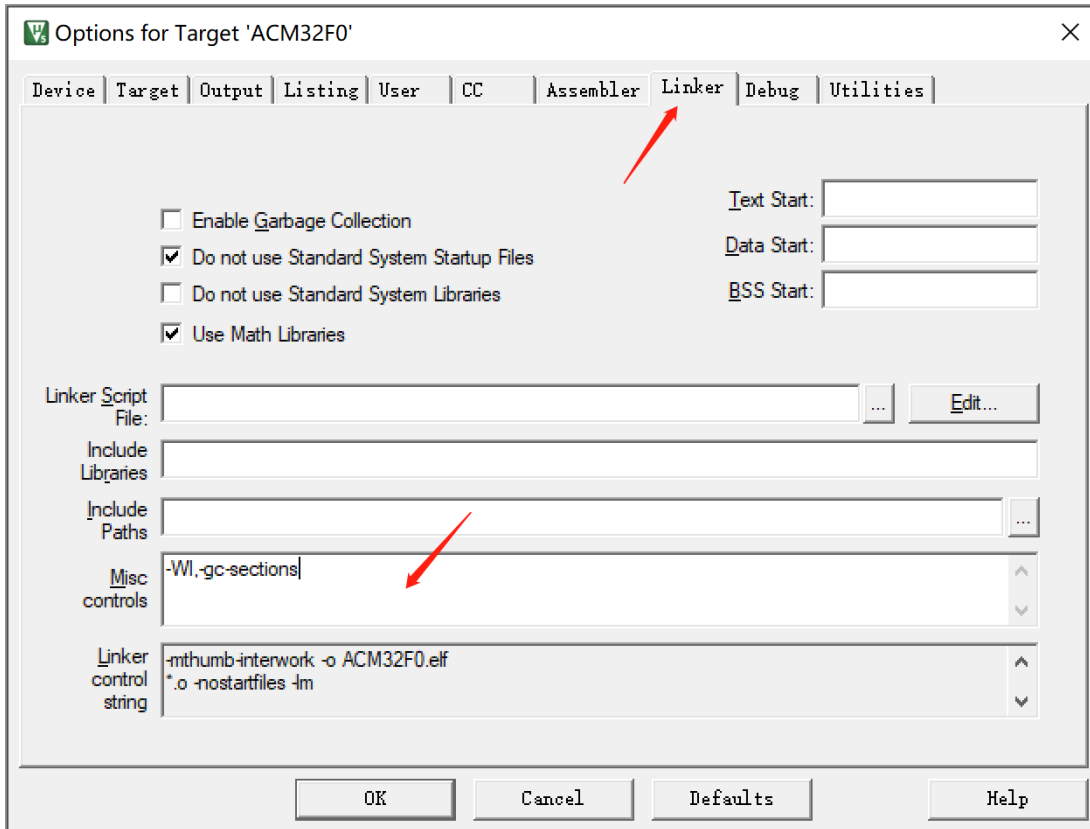
3) 然后点击魔法棒，配置以下几项：  
配置 CC 项，加入 `-mcpu=cortex-m0 -mthumb -fdata-sections -ffunction-sections`



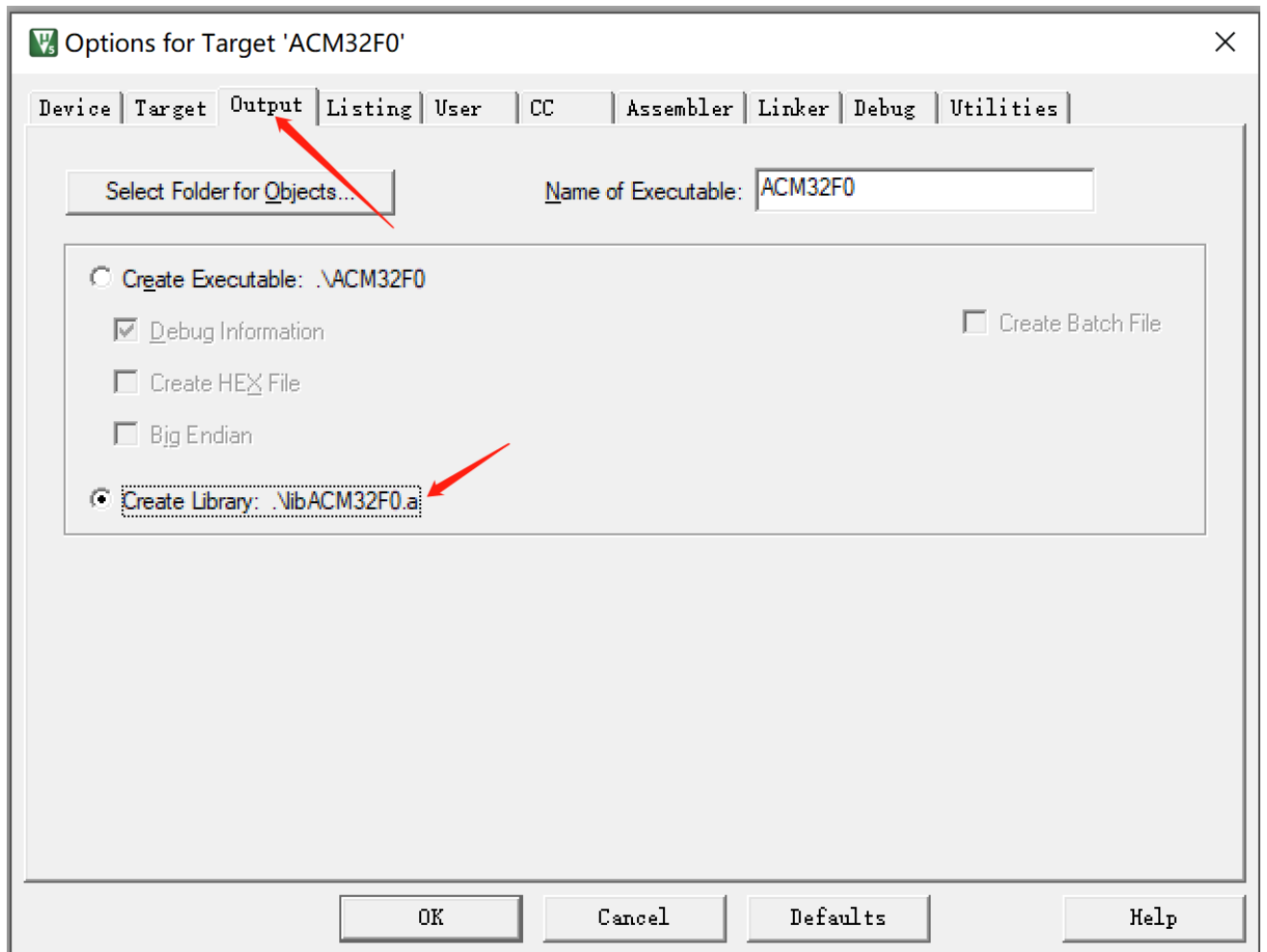
配置 Assembler 项，加入 `-mcpu=cortex-m0 -mthumb`












修改 Linker 项，加入-Wl,-gc-sections，主要是配置警告等级以及优化选项。



4) Output 下选择生成.a 库



- 5) 将工程中引用的 cmsis\_armclang.h 文件改为引用 cmsis\_gcc.h 文件。
- 6) 开始编译，不报错的话会生成对应的.a 库。

 ACM32F0.uvprojx	2022/3/9 10:59
 ACM32F0_ACM32F0.dep	2022/3/9 10:59
 ArInp.bat	2022/3/9 10:59
 ArInp.Scr	2022/3/9 10:59
 Debug_RAM.ini	2022/3/9 10:19
 HAL_EFlash_EX.build_log.htm	2022/3/9 10:59
 hal_eflash_ex.d	2022/3/9 10:59
 hal_eflash_ex.o	2022/3/9 10:59
 libHAL_EFlash_EX.a	2022/3/9 10:59

使用以上方法的好处是生成的.a 是直接基于 cm0 核的，如果使用 arm-none-eabi-ar 命令生成，需要指定对应的内核信息。



## 4. 加入静态库，生成最终工程

- 1) 修改 makefile 文件，加入.a 库：

```
# libraries
LIBS = -lc -lm -lnosys
LIBS += libHAL_EFlash_EX.a
```

- 2) 修改系统时钟为 64M，使能 EFlash,编译工程。

如果出现找不到对应函数或者架构错误，说明生成的.a 库有问题或者加入库的路径出错，如果路径确认无误，需要查看生成.a 库过程中相应的编译选项是否设置正确。

- 3) 如果编译通过，会生成最新的.bin 文件。
- 4) 使用 make clean 命令删除多余的文件。
- 5) 使用 boot 工具烧录 bin 文件，查看对应信息，可以看到系统正常运行。

```
 Geshe Beacon is officially renamed as Geshe Debug Genius which includes a new network authoriz
1 [2022-03-10 13:47:37.146 R]GCC ->MCU is running, HCLK=64000000Hz, PCLK=64000000Hz
2
3 [2022-03-10 13:47:37.176 R]----- UART Test -----
4 Please enter any String/Data
5
6 [2022-03-10 13:47:39.623 T]
7 12123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@
8 ABCDE1234123456789@ABCDE56789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE
9 [2022-03-10 13:47:39.683 R]
10 12123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@
11 ABCDE1234123456789@ABCDE56789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE
12
```

## 联系我们

公司：上海爱信诺航芯电子科技有限公司  
地址：上海市闵行区合川路 2570 号科技绿洲三期 2 号楼 702 室  
邮编：200241  
电话：+86-21-6125 9080  
传真：+86-21-6125 9080-830  
Email: [Service@AisinoChip.com](mailto:Service@AisinoChip.com)  
Website: [www.aisinochip.com](http://www.aisinochip.com)

## 版本维护

版本	日期	作者	描述
V1.0	2022-03-10	Aisinochip	初始版

本文档的所有部分，其著作权归上海爱信诺航芯电子科技有限公司（简称航芯公司）所有，未经航芯公司授权许可，任何个人及组织不得复制、转载、仿制本文档的全部或部分组件。本文档没有任何形式的担保、立场表达或其他暗示，若有任何因本文档或其中提及的产品所有资讯所引起的直接或间接损失，航芯公司及所属员工恕不为其担保任何责任。除此以外，本文档所提到的产品规格及资讯仅供参考，内容亦会随时更新，恕不另行通知。