



ACM32F070 / A070 / FP001 / WB15 系列芯片 MDK 平台移植到 GCC 环境

版本: V1.2 日期: 2025-3-10

上海航芯电子科技股份有限公司

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。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。

```
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_6
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-sjl;
onv-prefix=/mingw --with-libint1-prefix=/mingw --enable-1
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

我们实际使用的 GCC 编译器是 arm-none-eabi-gcc, 属于 GCC 的一个分支;

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;

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:

- 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;
- 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;
- 将 ST 工程下的 makefile 文件和.ld 文件拷贝到 UART 工程根目录下;
- 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;
```

- 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;
- 修改 UART 工程主时钟为 64M,关闭 EFlash;
- 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;

}

● 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。



● 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;

修改日期	类型	大/
2020/11/24 12:04	文件夹	
2020/11/24 12:08	文件夹	
2020/11/24 11:56	文件夹	
2020/11/24 12:08	文件夹	
	修改日期 2020/11/24 12:04 2020/11/24 12:08 2020/11/24 11:56 2020/11/24 12:08	修改日期 类型 2020/11/24 12:04 文件夹 2020/11/24 12:08 文件夹 2020/11/24 11:56 文件夹 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;

← → 🏴 陸 陸 陸 [] 谭 谭 //』 //編 💯set_PRIMAS	SK 📃 🗟 🥐 🍭 - 🌢 🔿 🔗 (
Manage Project Items	×			
Project Items Folders/Extensions Books Project Info/Lay	yer			
Development Tool Folders:	Default File Extensions:			
□ Use Settings from TOOLS.IN:	C Source: *.c			
Tool Base Folder: C:\Keil_v5\ARM\	C++ Source: *.cpp			
	Asm Source: *.s*; *.src; *.a*			
	Object: *.obj; *.o			
LIB:	Library: ^{*_lib}			
<u>R</u> egfile:	Document: *.txt; *.h; *.inc; *.md			
Use ARM Compiler "ARMCLANG"; ".\ARMCC"				
Setup Default ARM Compiler Version	on			
3				
Use GCC Compiler (GNU) for ARM projects				
Prefix: arm-none-eabi- Folder: C:\Keil_v5\ARM	M/GCC/			
OK	.cel Help			

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

	×
Gottons for Target 'ACM32F0'	×
Device Target Output Listing User CC Assem	oler Linker Debug Utilities
Preprocessor Symbols	
Define:	
U <u>n</u> define:	
Code Generation	
Enable APCS (ARM Procedure Call Standard)	<u>O</u> ptimization: <mark><default></default></mark> ▼
Generate Stack Check Code	Warnings: Level 1
✓ Support Calls between ARM and THUMB Instruction Set	
	Compile Thumb Code
Include Paths	
Misc	ections
Controls	
compiler -c -mcpu=cortex-m1 -mthumb -gdwarf-2 -MD -Wall -O control m0 -mthumb -fdata-sections -ffunction-sections -IC:\Ke string	-mapcs+frame -mthumb-interwork -mcpu=cortex- eil_v5\ARM\CMSIS\Include -IC:\Keil_v5\ARM
,	
OK Cancel	Defaults Help

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

			l ag Assell.	let int	1	
evice Tar	get Uutput I	listing User	CC Assemble	r Linker Debug	Utilities	
 Conditiona <u>D</u>efine: 	al Assembly Contro	ol Symbols				
U <u>n</u> define:						
– Language	/ Code Generation	on				
🔽 Enable	- ARM/Thumb Int					
	e Anim/ mumb in	terworking				
	ANM/ INDID IN	terworking				
		terworking				
		terworking				
		terworking				
Include		terworking				
Include Paths		lerworking				
Include Paths Misc	mcpu=cortex-n	n0 -mthumb				
Include Paths <u>M</u> isc Controls	-mcpu=cortex-n	n0 -mthumb				
Include Paths Misc Controls Assembler control string	mcpu=cortex-n \Keil_v5\ARM	n0 -mthumb n1 -mthumb -gdwarf ICMSIS\Include -IC	-2 -mthumb-interwork - :\Keil_v5\ARM\GCC\	-MD *.d -mcpu=cortes am-none-eabi\includ	<m0 -ic:<br="" -mthumb="">e -o *.o</m0>	····
Include Paths <u>M</u> isc Controls Assembler control string	-mcpu=cortex-n \Keil_v5\ARM	n0 -mthumb n1 -mthumb .CMSIS\Include -IC	-2 -mthumb-interwork :\Keii_v5\ARM\GCC\	-MD *.d -mcpu=corte arm-none-eabi\includ	<-m0 -mthumb -IC: e -o *.o	

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

🔣 Options fo	or Target 'ACM32F0'	×
Device Targ	et Output Listing User CC Assembler Linker Debug Utilities	
	Enable Garbage Collection Data Start:	
	Do not use Standard System Startup Files	
Linker <u>S</u> cript File:	<u>E</u> dit	
Include Lib <u>r</u> aries		
Include Paths	···	
<u>M</u> isc controls	-WI,-gc-sections	
Linker control string	-mthumb-interwork -o ACM32F0.elf *.o -nostartfiles -Im	
	OK Cancel Defaults Help	

4) Output 下选择生成.a 库

🔣 Options for Target 'ACM	132F0'				\times
Device Target Output I	isting Vser (C Assembler	Linker Debug	Utilities	
Select Folder for Objects		Name of Executable:	ACM32F0		
C Create Executable: .V	ACM32F0			🗖 Create Batch File	
 Big Endian Create Library: NibAC 	M32F0.a				
	07	Curvel	D. C.J.	11.1	
	40	Lancei	Delanitz	летр	

- 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
💿 Arlnp.bat	2022/3/9 10:59
III Arlnp.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;



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
[2022-03-10 13:47:37.146 R]GCC ->MCU is running, HCLK=6400000Hz, PCLK=6400000Hz
[2022-03-10 13:47:37.176 R]------ UART Test ------
Please enter any String/Data
[2022-03-10 13:47:39.623 T]
12123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE123456789@ABCDE1
```

5. 版本历史

版本	日期	作者	描述
V1.0	2021-04-30	Hangxin	初始版
V1.1	2023-02-10	Hangxin	添加 A070 系列芯片支持
V1.2	2025-03-10	Hangxin	添加 WB15 系列芯片支持

6. 版权声明

本文档的所有部分,其著作产权归上海航芯电子科技股份有限公司(简称航芯科技)所有,未经航芯科技授权 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,航芯科技及所属员工 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。

联系我们

- 公司: 上海航芯电子科技股份有限公司
- 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: <u>service@HangChip.com</u>
- Website: www.hangChip.com