ACM32F0x0/FP0X 系列芯片开发快速上手

目录

1. 开发板资源概述

1.1. 开发板板载资源

- MCU：ACM32F070，LQFP64(10mm*10mm)，FLASH：128KB，SRAM：32KB；
- 1个电源指示灯，1个Link指示灯，1个用户指示灯。
- 1个电源供应/接入口。
- 1个启动模式选择跳帽，选择芯片启动模式。
- 1个系统复位按钮，用于复位MCU。
- 1个用户功能按钮。
- 提供了CMSIS-DAP方式下载、调试，USB虚拟串口打印功能。
- 所有IO口全部引出，包括晶振占用的IO口。
- 开发板支持外接14个触摸按键和段码式显示屏，开发包中有原理图。

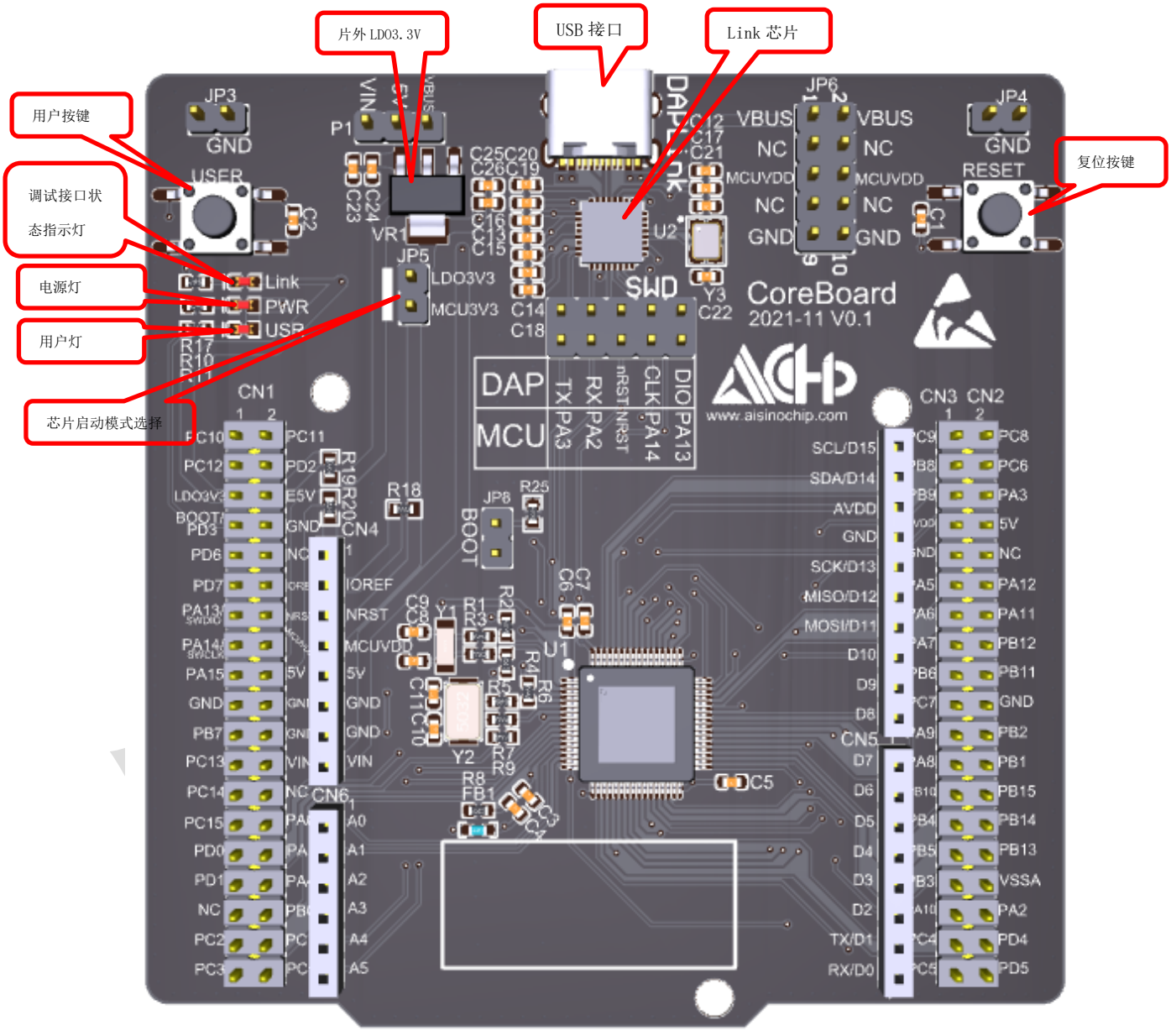


图 1-1 ACM32F070 Layout

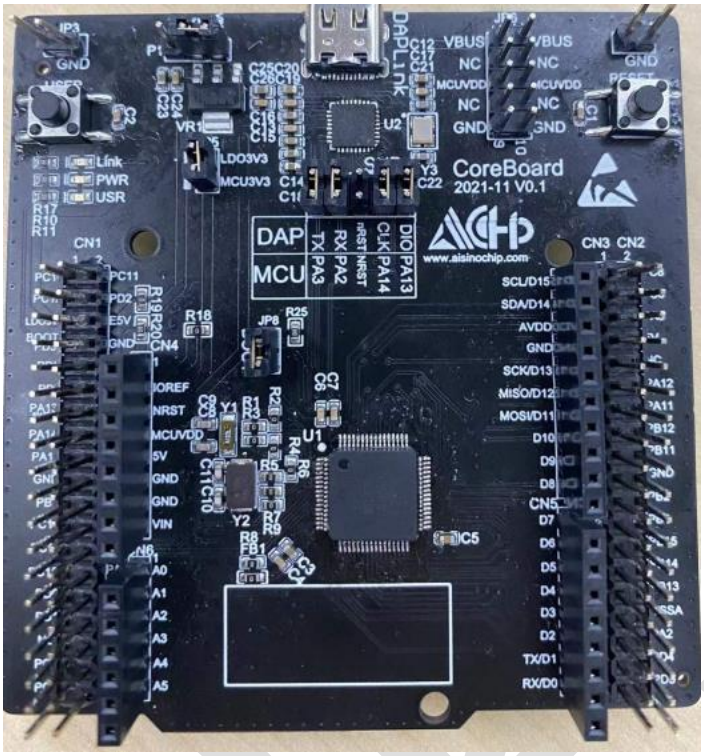


图 1-2 ACM32F070 实物图

2. 板载硬件介绍

2.1. MCU 之 ACM32F070

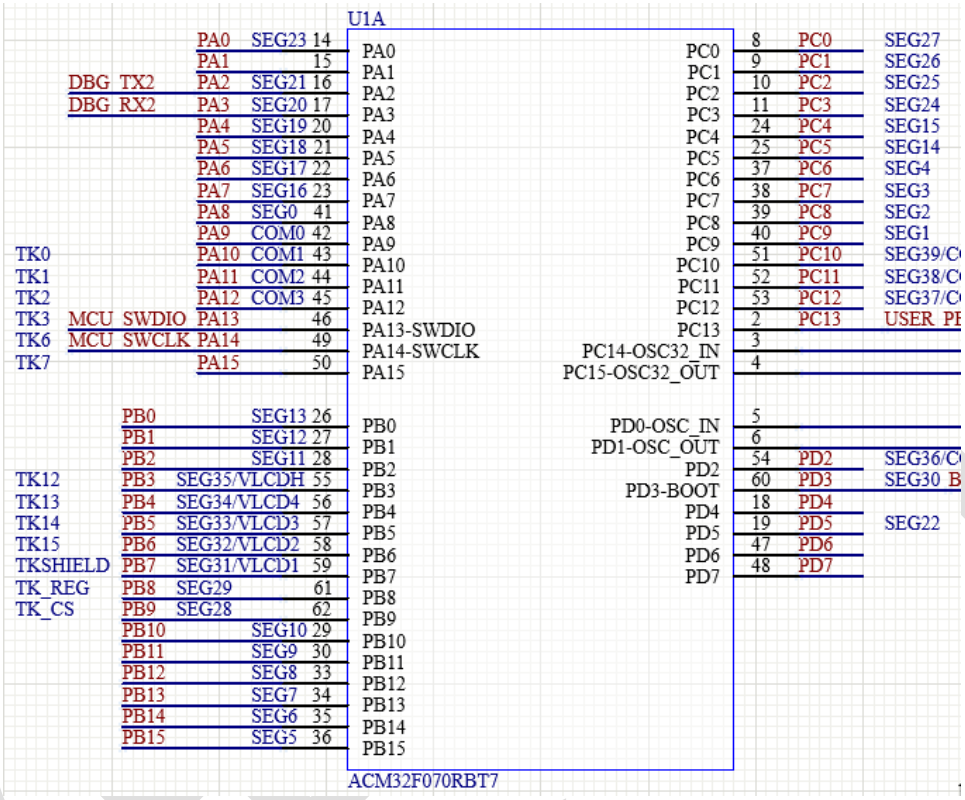


图 2-1 MCU（1）

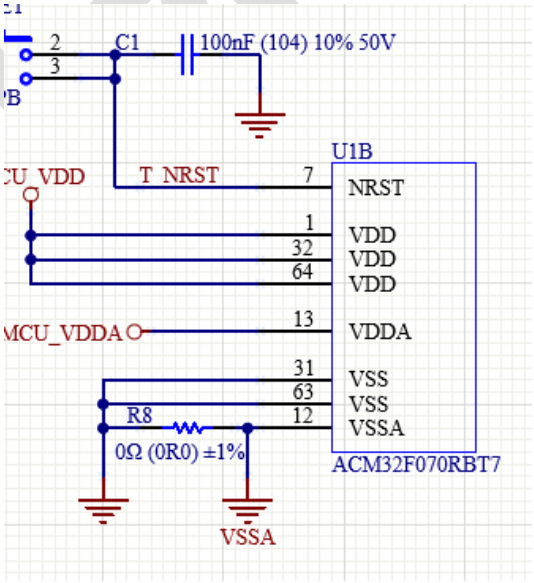


图 2-2 MCU（2）

2.2. CMSIS-DAP 调试下载接口/虚拟 USB 转串口

采用 ACH512 芯片作为下载调试/串口打印芯片，当用户通过 USB 线将开发板与 PC 机连接后，调试/下载程序时在 KEIL 中选择 CMSIS-DAP Debugger 模式，并且在设备管理器中可以找到航芯虚拟串口端口。(后续会具体介绍开发环境搭建)

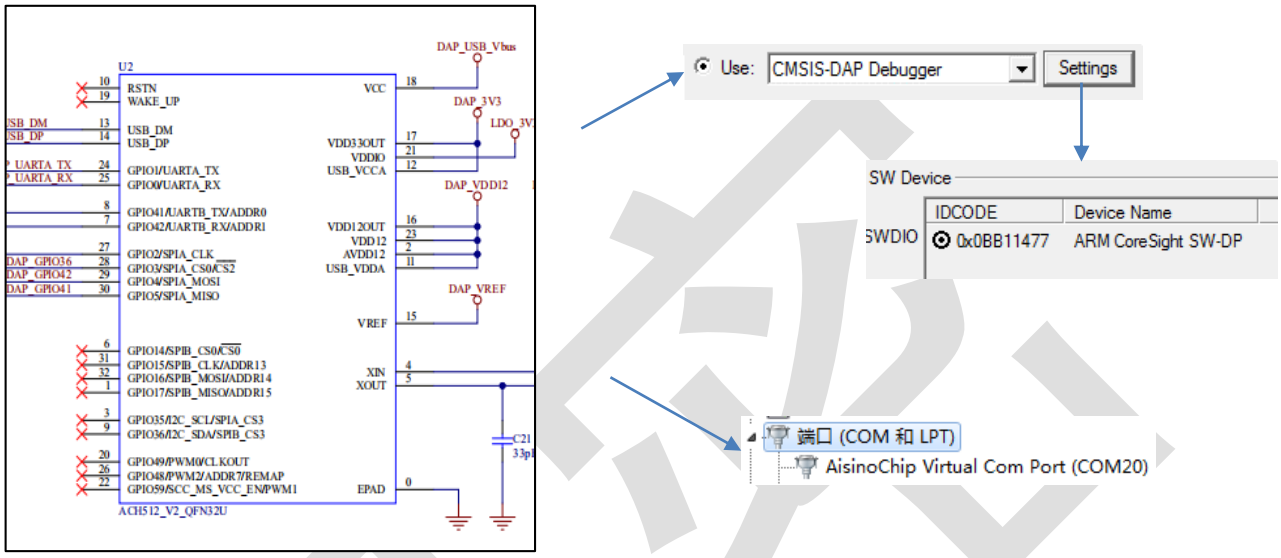


图 2-3 下载调试接口芯片

2.3. 板载 LED

用户程序灯(USR)，绿色，辅助用户查询/调试程序。

电源指示灯(PWR)，绿色，MCU 上电时亮，下电时熄灭。

Link 指示灯(Link) ，红色，反映芯片的下载/调试状态。

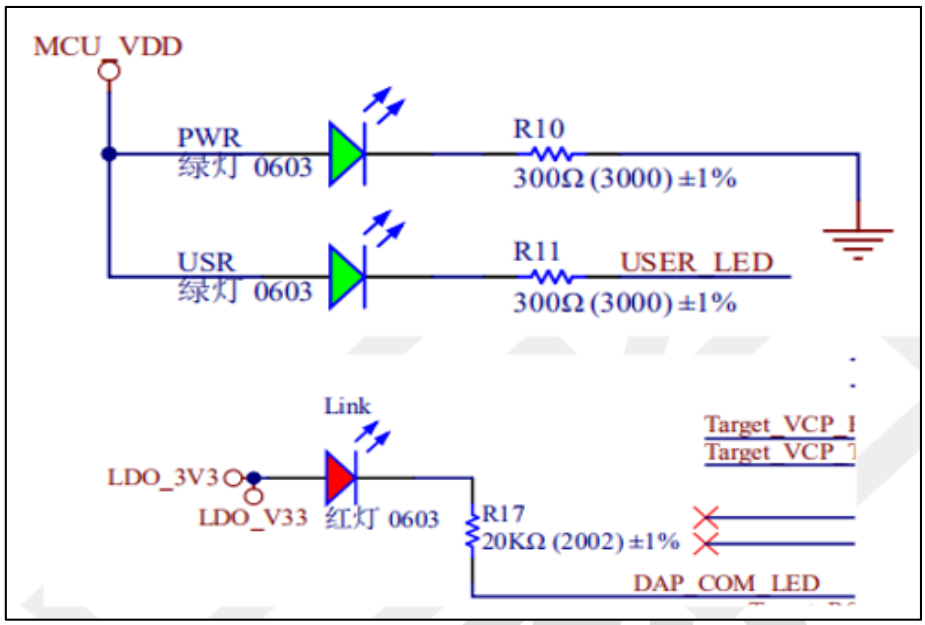


图 2-4 板载 LED

2.4. 按键和 BOOT 跳帽

MCU 复位按键(RstnPB)红色，用于复位主控芯片，按下按键芯片复位。

用户按键(UserPB)黑色，人机交互按键。

芯片启动模式选择跳帽，连接跳帽，芯片会运行用户代码（eFlash）；断开跳帽，芯片只会运行自有 Boot 程序(ROM 中）。

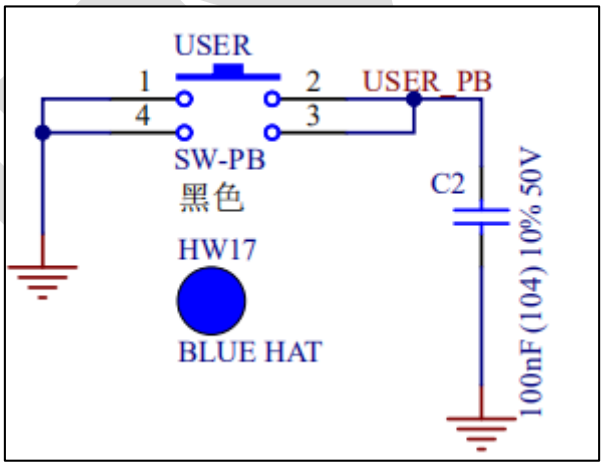


图 2-4 用户按键

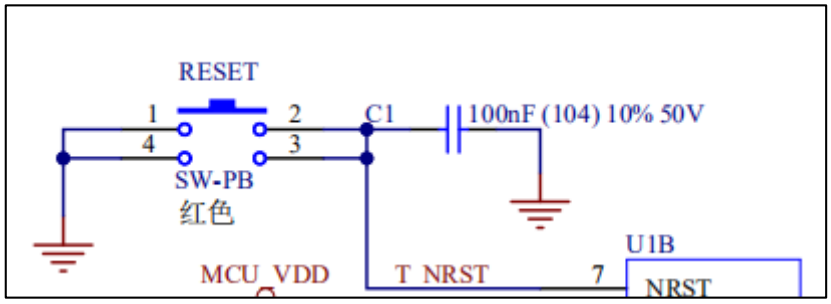


图 2-5 复位按键

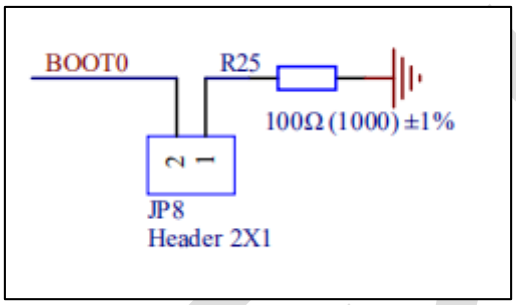


图 2-6 跳帽选择

2.5. 电源输入输出

通常使用 USB 供电，板载 LDO 将 5V(USB_Vbus)转成 3.3V，JP5 默认用跳帽短接。当不使用 USB 供电时，可以断开 JP5，通过排针从外部接入电源给板载 MCU 供电。MCU 的 VDD 输入电压范围是 1.7V~3.6V。

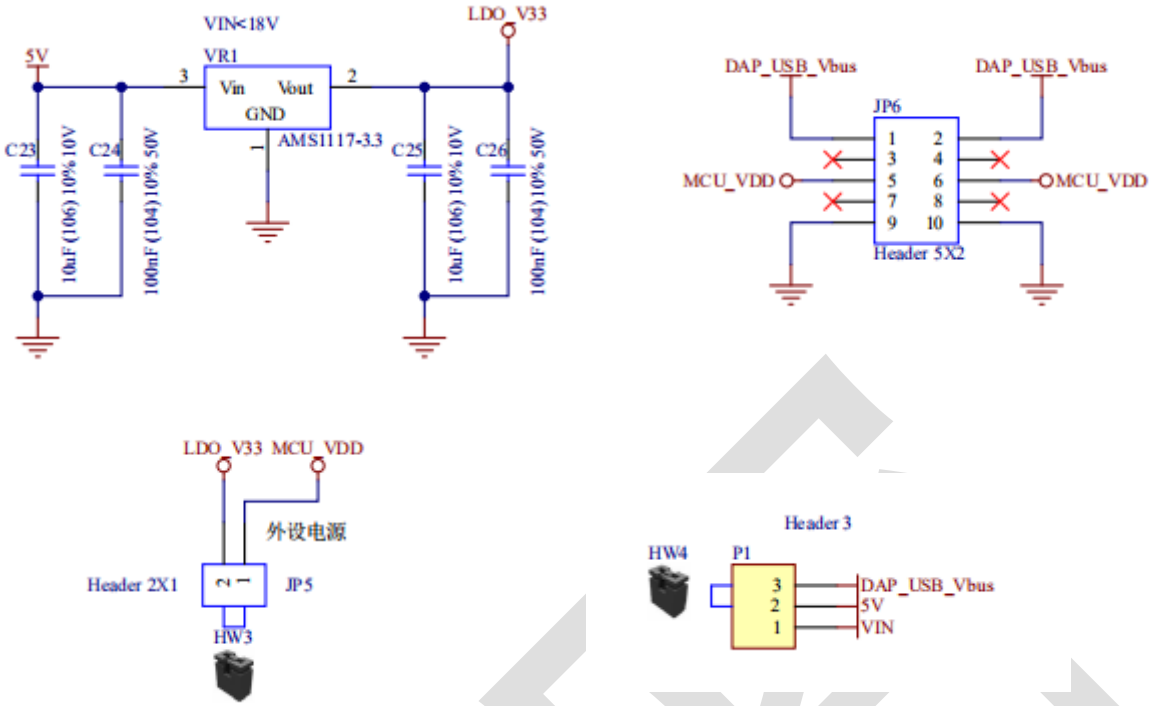


图 2-7 电源

2.6. 板载时钟

板上默认有外部高速 8MHz 无源晶振，外部低速 32.768KHz 无源晶振。程序可以使用片内的 RC64M 时钟作为系统时钟，也可以使用 8MHz 的外部晶振加 PLL 输出的时钟作为系统时钟。32.768KHz 的外部晶振时钟可用作 RTC 的工作时钟。

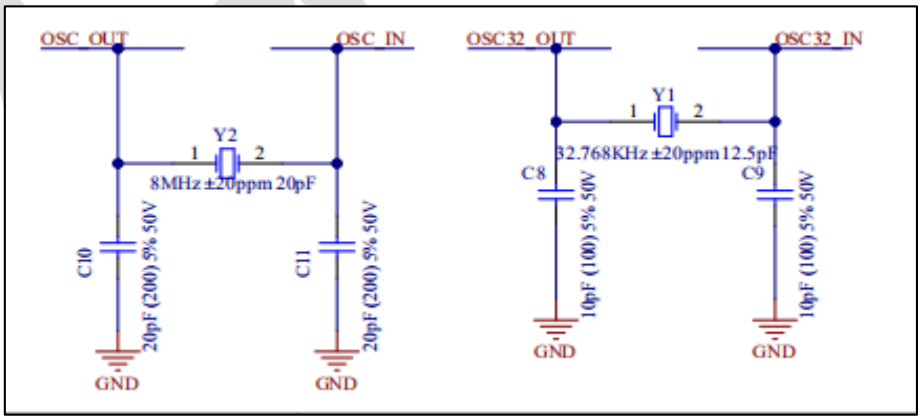


图 2-8 片外时钟

2.7. IO 口

芯片所有 IO 引出到排针 CN1、CN2、CN3 上。并且 JP3、JP4、JP5、JP6 上都提供电源/地，

可以灵活给接插的扩展子板输入/输出电源。

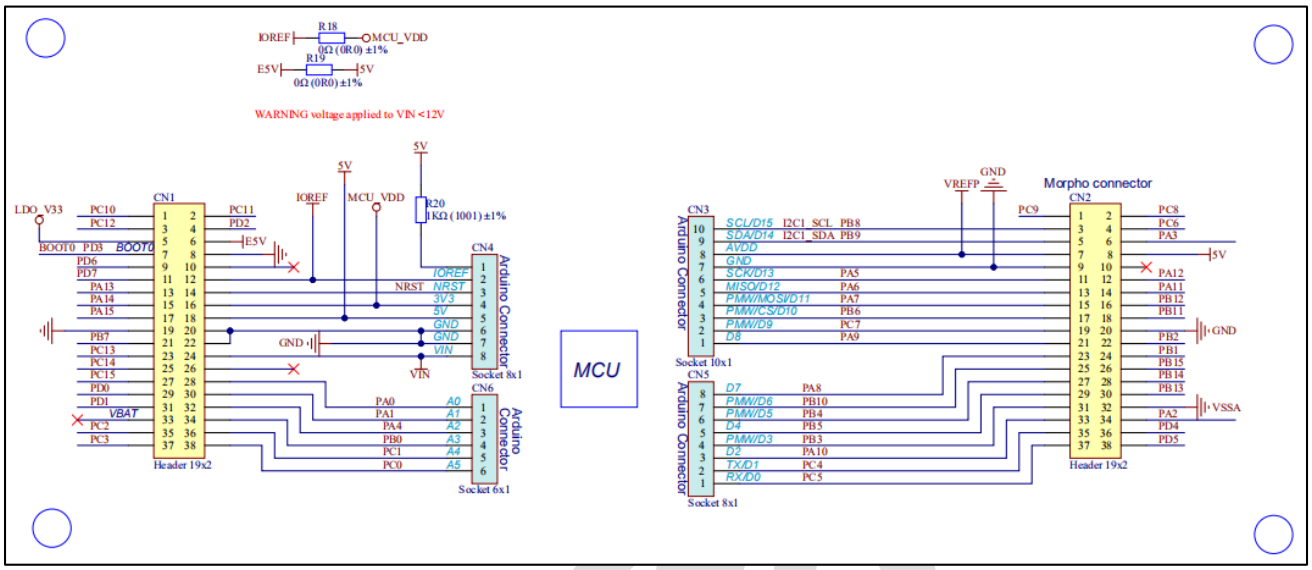
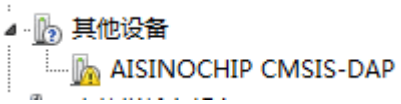


图 2-9 扩展 IO 接口

3. 开发环境搭建

3.1. 设备驱动安装

用户第一次将开发板与 PC 机的 USB 端口相连，Win7 系统用户需要安装 Link 芯片的驱动(Win10 系统用户不需要)。



选择此设备，右键更新驱动，直接点下面的->从驱动列表中选择(COM 类)

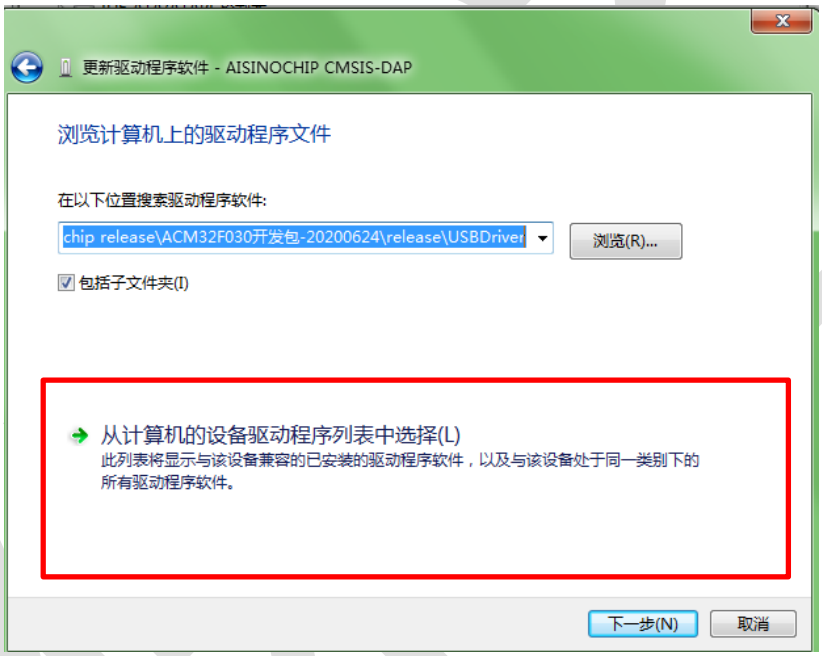


图 3-1 选择驱动程序

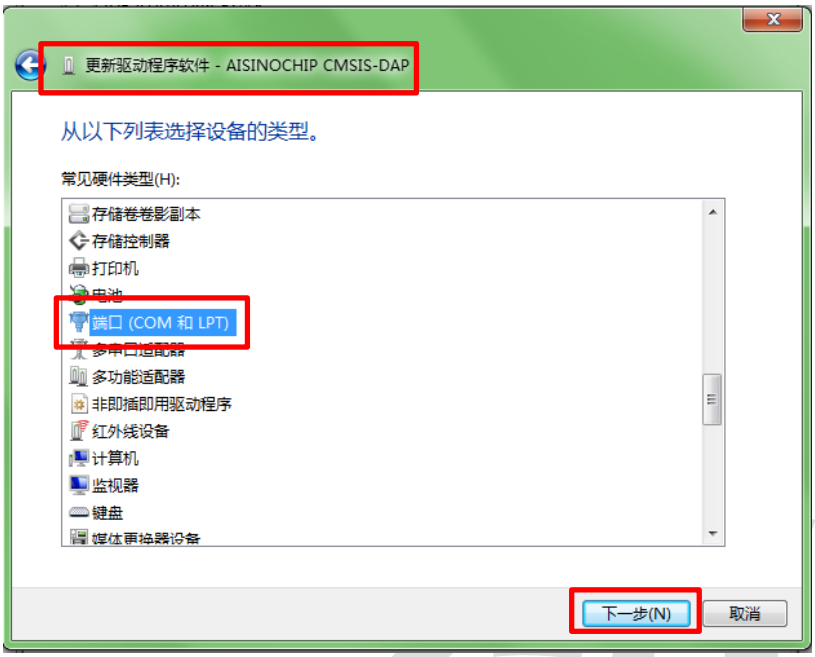


图 3-2 选择驱动程序

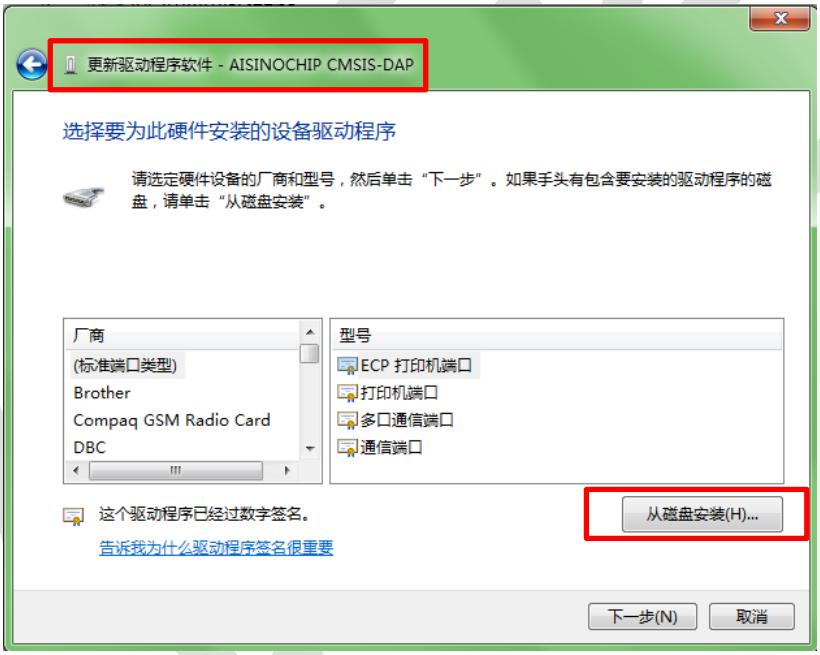


图 3-3 选择驱动程序

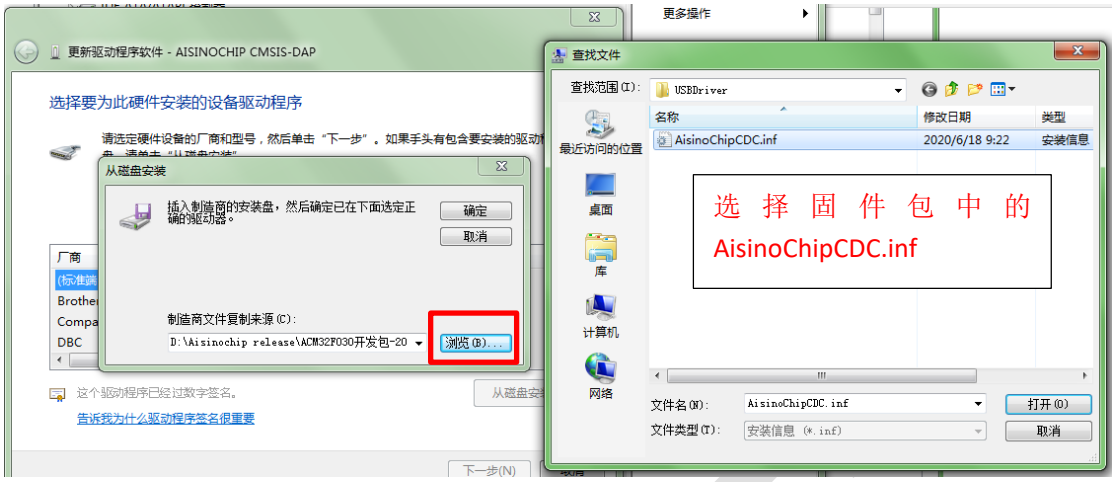


图 3-4 选择驱动程序

忽略警告提示，继续安装，安装完毕。后续调试程序可以选择此 COM 口获得打印信息。

图 3-5 选择驱动程序

3.2. 仿真环境搭建

3.2.1. MCU 选择

安装航芯开发包里的 pack 包”Aisinochip.ACM32F0X0.1.0.0.pack”，安装完成，选择所使用的芯片具体型号，如图所示：

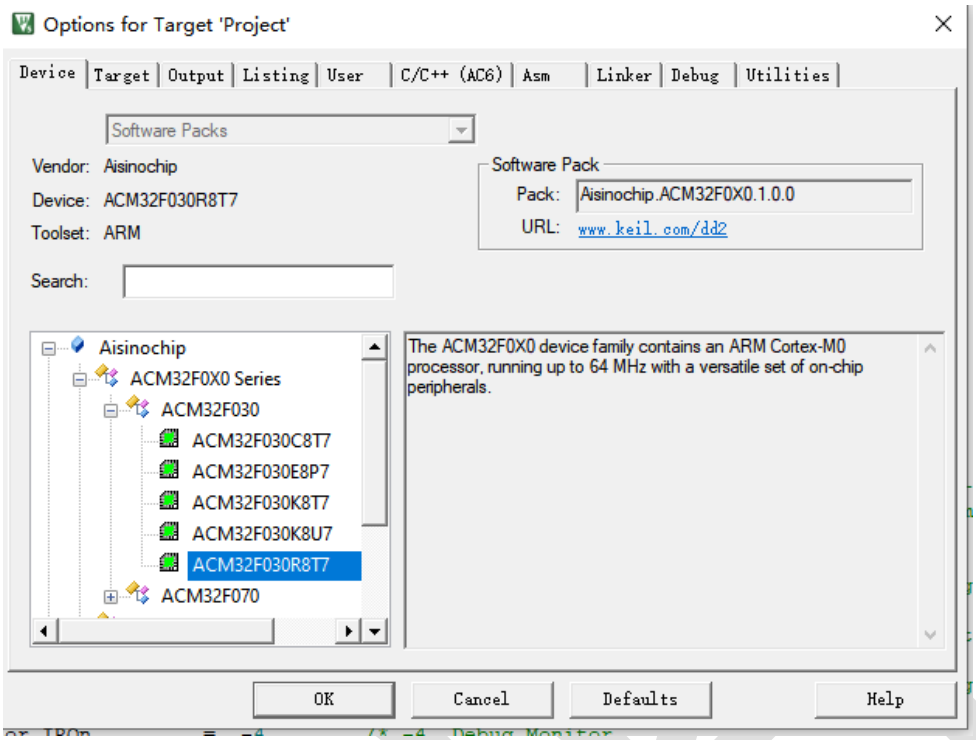


图 3-6 MCU 选择

3.2.2. 下载口选择

连接调试器：如图所示，切换到 Debug 页面后在下拉框中选择 CMSIS-DAP Debugger，然后点击 Settings 按钮，显示图 3-7 的调试器连接情况。

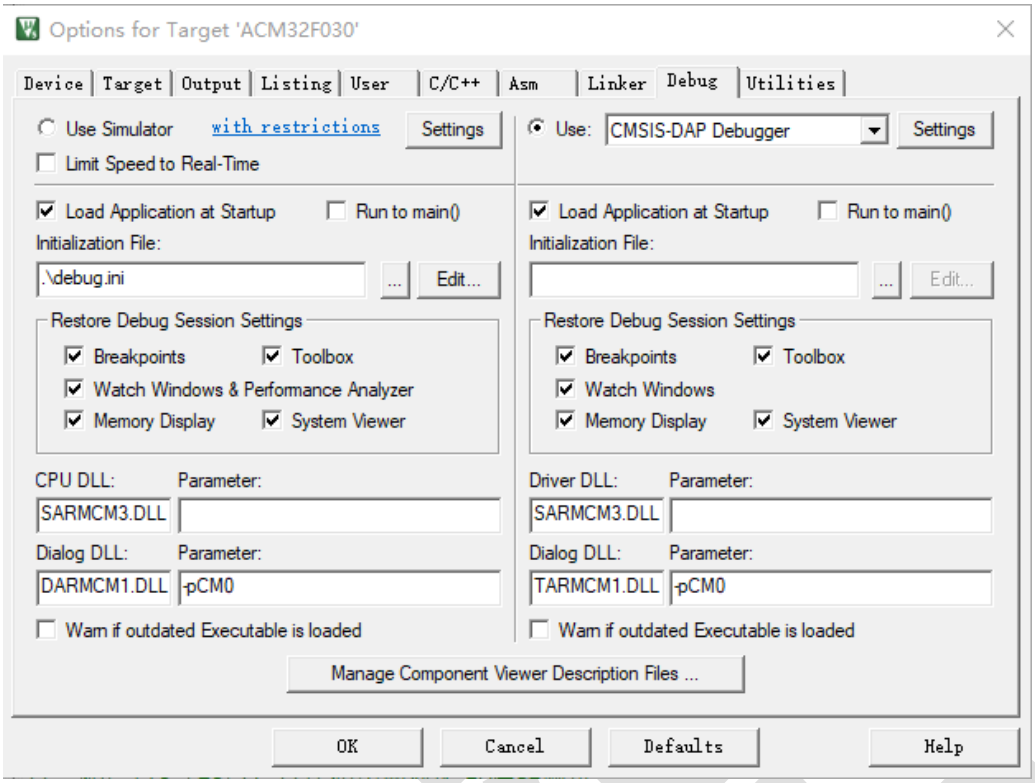


图 3-7 调试器类型

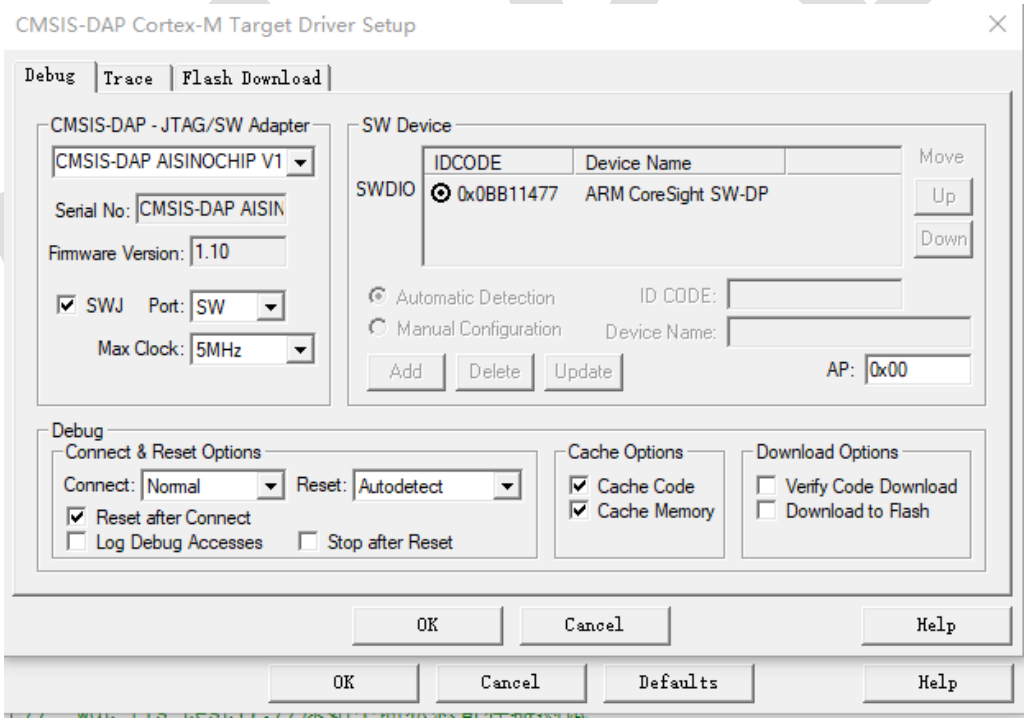


图 3-8 调试器连接情况

3.2.3. 下载算法选择

3.2.1 中的 pack 包包含了算法文件，安装完后会自动将算法文件拷贝到正确的路径。也可

以手动操作，将开发包中的 ACM32F0x0_eflash.flm 拷贝到 C:\KEIL\ARM\Flash 目录下。然后在图的基础上点击“Flash Download”按钮进入插件选择界面，选择烧录插件。

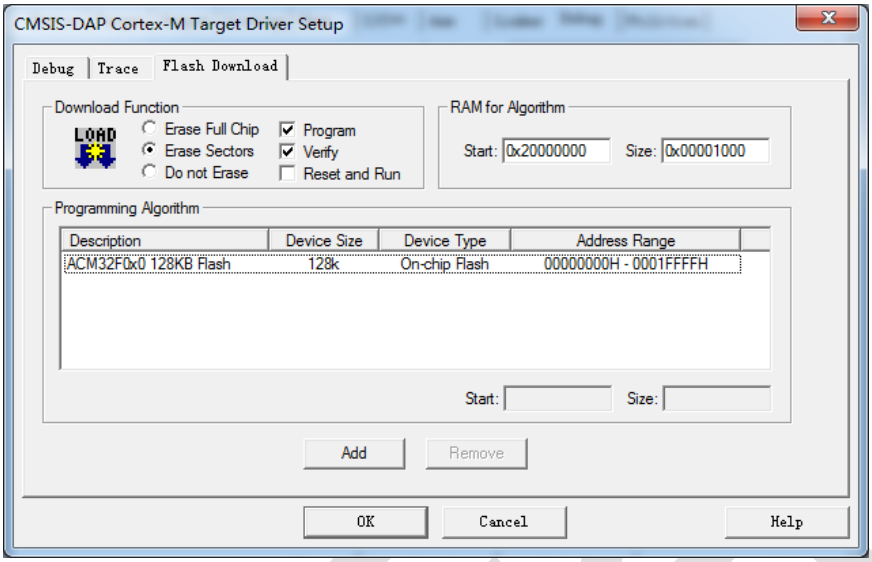


图 3-9 添加烧录插件

3.2.4. 调试模式配置

按如图所示的配置，就可以在调试前先下载程序到 eFlash 中然后开始调试程序。

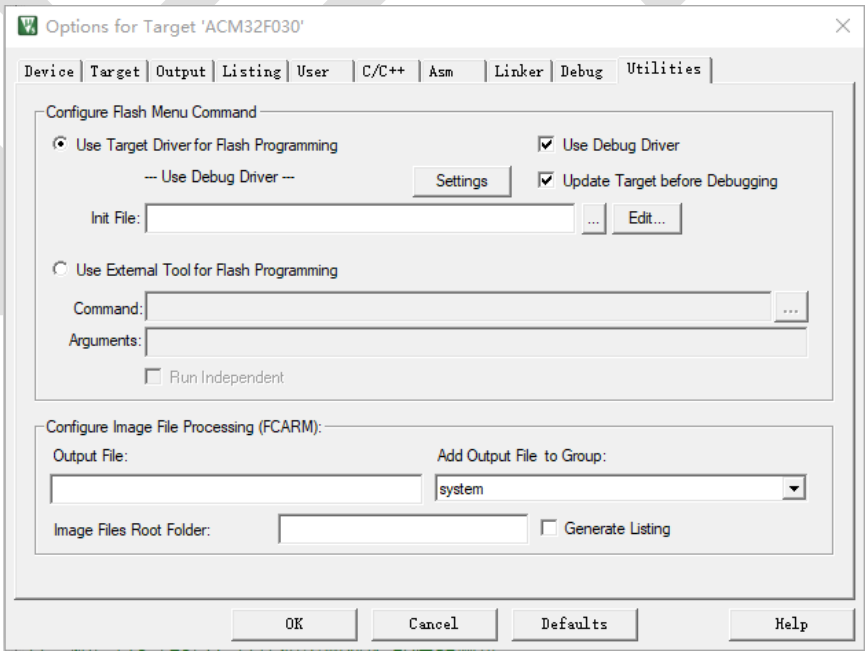


图 3-10 调试前先下载程序

3.2.5. BOOT 引脚配置

ACM32F0x0/FP0X 两种启动模式：ROM 启动和 eFlash 启动。系统上电时，芯片会读取安全序列字段和系统寄存器 WMR 的 BootMode 标志位，决定是将 eFlash 还是将 ROM 映射到 0x0 起始逻辑地址。BootMode 标志位由上电时 BOOT 引脚(PF3)的高低电平决定。图描述了芯片启动模式选择过程。

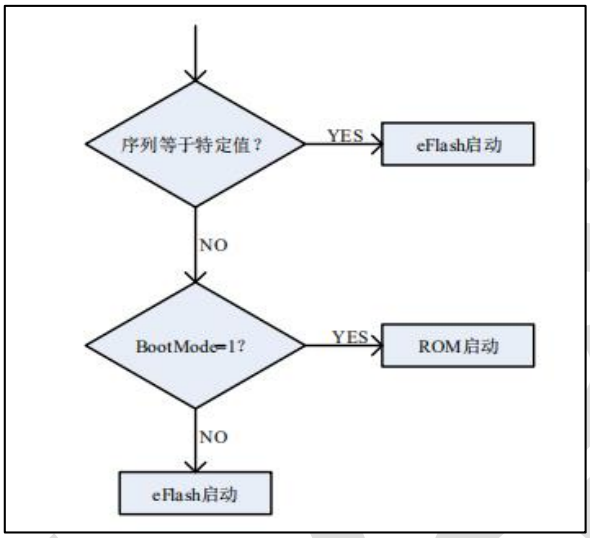
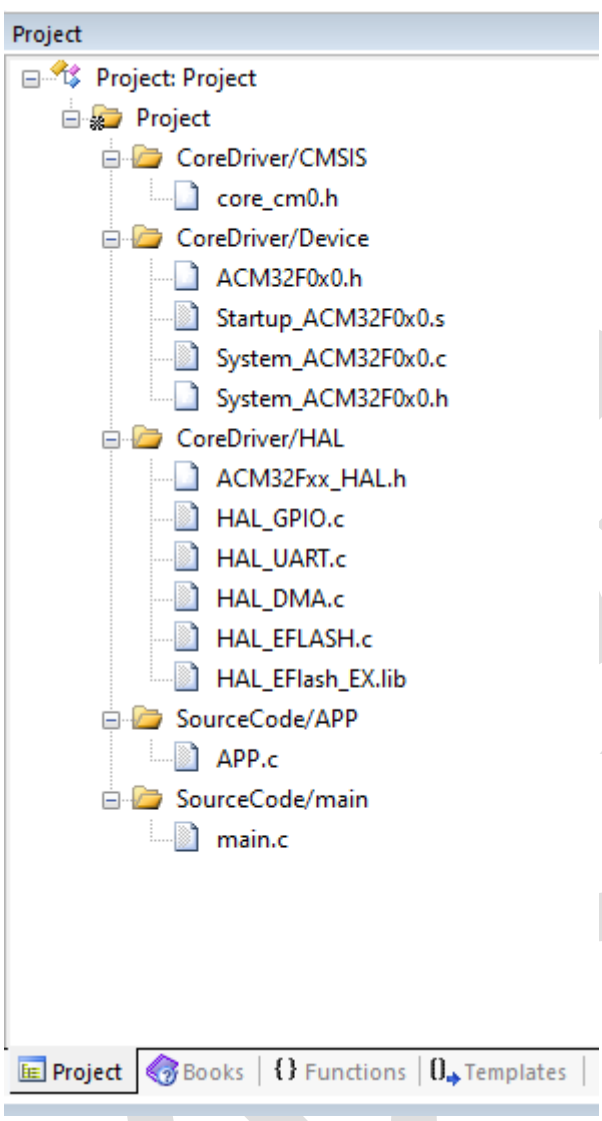


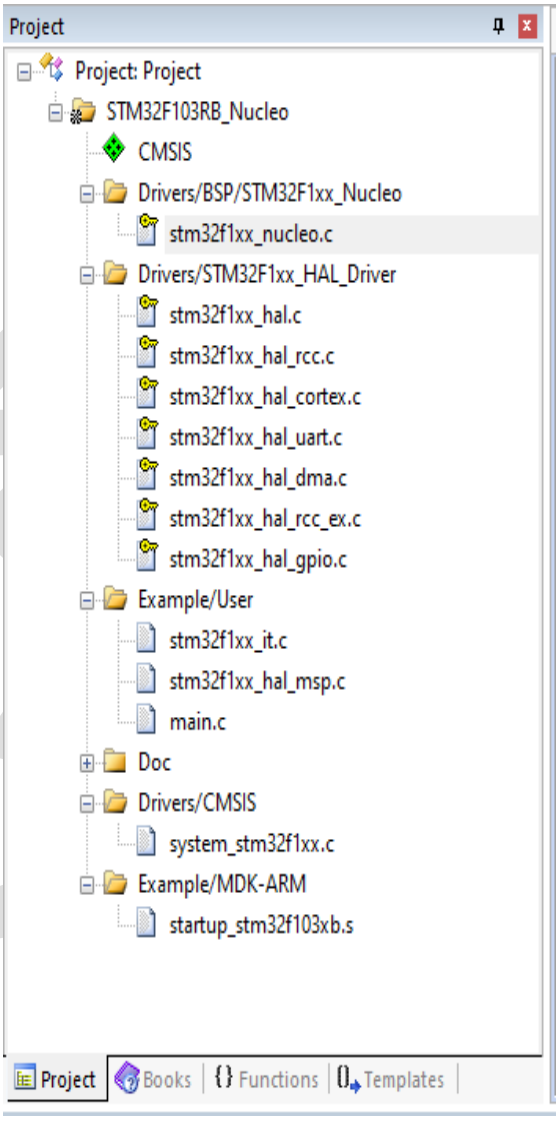
图 3-11 航芯启动模式选择

故需要在不写入安全序列的情况下，将 BOOT 引脚拉低，使 ACM32F0x0/FP0X 从 eflash 启动。

3.3. KEIL 工程移植

本章节描述如何从 STM32F103 的 UART Keil 工程移植到航芯的 ACM32F0/FP0 芯片上。对比一下 STM32F103 和 ACM32F0/FP0 工程的异同：





我们看看 STM32F103 的 UART 工程和航芯工程的对应关系以及移植方法。

1)、STM32F103 的 Drivers/HAL_Driver 对应航芯工程的 CoreDriver/HAL，里面放着的是芯片厂商提供的外设驱动库。

2)、STM32F103 中 CMSIS 对应航芯工程 CoreDriver/CMSIS 的 core_cm0.h 服务，此文件是 ARM 官方提供的。

3)、STM32F103 中 Example/MDK 下的.s 启动文件对应航芯工程中 CoreDriver/Device 中的.s 文件，主要是中断向量表。STM32F103 中的 system_stm32f1xx.c 对应的是航芯工程中的 System_ACM32F0x0.c，此文件是和芯片系统相关的，System_ACM32F0x0.c 中主要是时钟和复位

功能。

4)、STM32F103 中的 User 下的 main.c 对应航芯工程中的 SourceCode/main.c 和 SourceCode/APP.c，是应用层的逻辑，APP.c 实现的是串口的收发，将收到的数据全部发回。

5)、STM32F103 中的 stm32f10x_it.c 对应 ACM32F0x0 中 SourceCode/APP.c 中的 UART1_IRQHandler，是中断处理函数。

如上所述，航芯的 demo 分层清晰：应用层、硬件抽象层、系统层、ARM 公共文件。应用层的 main.c 文件中主要是系统时钟初始化，systick 初始化，串口打印初始；而 APP.c 是 uart demo 应用层的代码，是业务相关、应用相关的逻辑处理。HAL 下的文件是硬件抽象层，如 I2C/SPI/UART 等的 HAL 文件都包含了外设初始化和通信服务，所有和外设通信、数据交互的服务都在 HAL_xxx.c 文件中提供好了。系统层是芯片系统相关的代码，如时钟和复位。ARM 公共文件则是处理器内核相关的代码，是 ARM 提供的标准化接口，如 NVIC 接口，systick 初始化等。

串口 Demo 移植介绍：

打开 ACM32F0/FP0 UART demo 中的 MDK_Project 文件夹下的工程文件。

1）MCU 选择，Device 选择对应的 ACM32F0/FP0。

2）如图 3-12 所示，IROM1 起始地址填 0，是 eflash 的起始地址，大小根据产品型号的参数填写（见数据手册），最大为 128KB，开发板上 F070R8T7 芯片为 128KB 即 0x20000。IRAM1 起始地址填 0x20000000，为 SRAM 的起始地址，大小根据产品型号的参数填写。**编译器选择 version 5**。

3）如图 3-13 所示，可以生产.bin 文件。

4）如图 3-14 所示，将 One ELF Section Per Function 的√打上，链接时未被调用到的函数不会被链接进去，能大大减少生成镜像文件的大小。编译优化选项见图 3-15，链接配置见图 3-14。

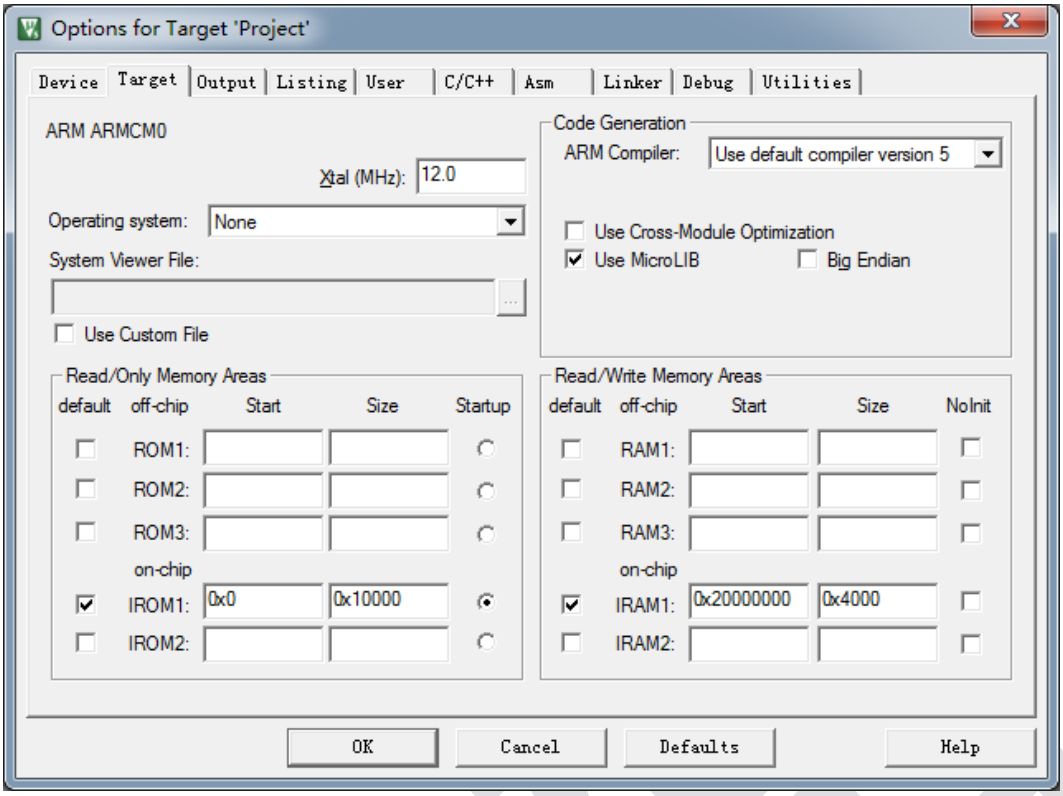


图 3-12 Target 配置

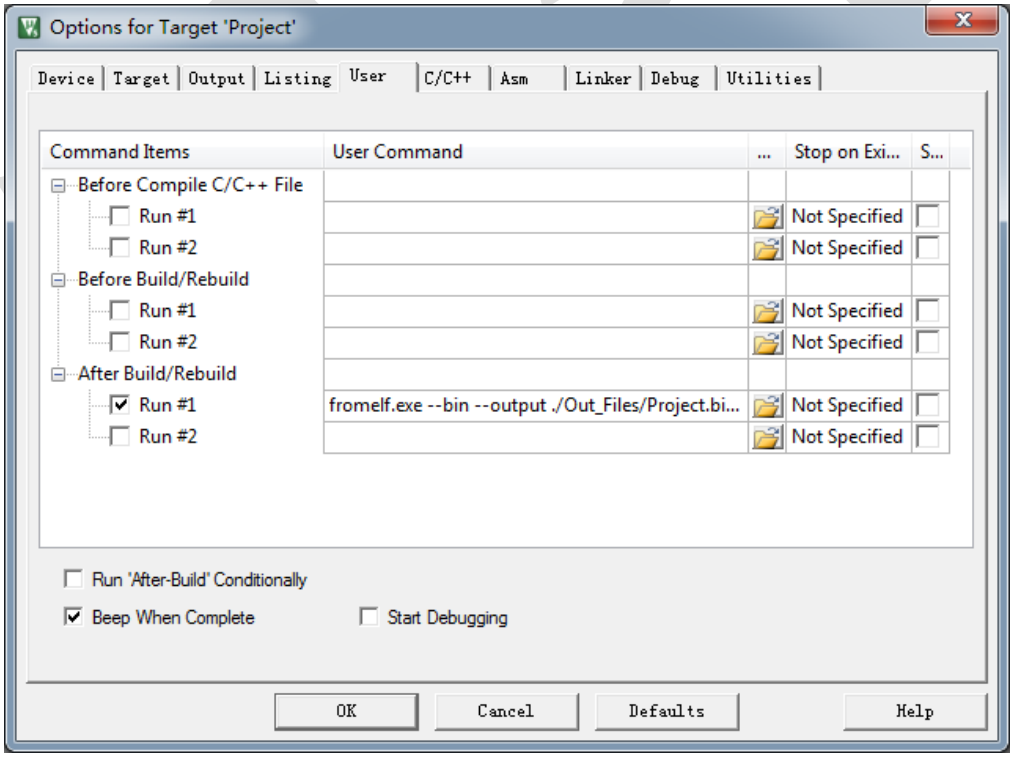


图 3-13 User 页面

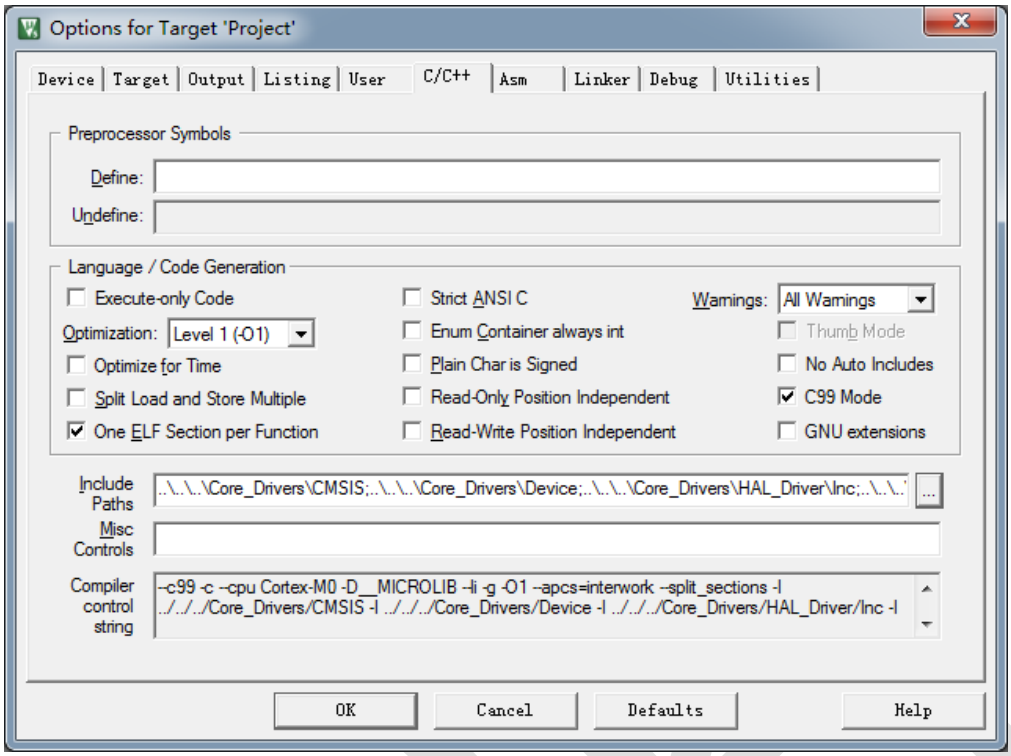


图 3-14 编译链接选项

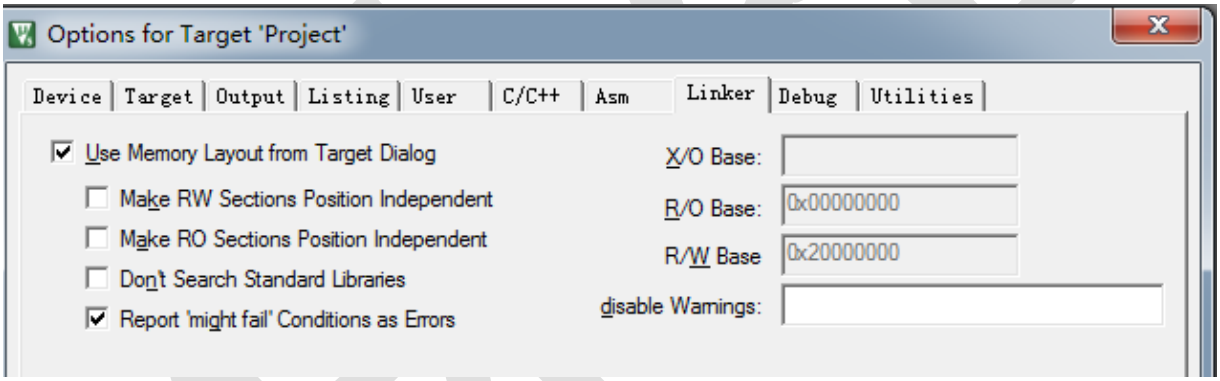


图 3-15 链接加载

5），代码层面上，保持 CoreDriver/CMSIS、CoreDriver/Device、CoreDriver/HAL 不变，将 main 文件进行替换。系统部分的替换内容如图 3-16 所示，串口部分的替换如图 3-17 和 3-18 所示，高度相似，都是通过 HAL 接口完成。头文件使用 APP.h 替换 main.h。去除图 3-19 的内容。去除 PUTCHAR_PROTOTYPE 以及之后的所有内容。

图 3-16 系统配置修改

```
  /*##-1- Configure the UART peripheral ######################################*/
  /* Put the USART peripheral in the Asynchronous mode (UART Mode) */
  /* UART configured as follows:
      - Word Length = 8 Bits (7 data bit + 1 parity bit) : BE CAREFUL : Program 7 c
      - Stop Bit    = One Stop bit
      - Parity      = ODD parity
      - BaudRate    = 9600 baud
      - Hardware flow control disabled (RTS and CTS signals) */
  UartHandle.Instance        = USARTx;

  UartHandle.Init.BaudRate   = 9600;
  UartHandle.Init.WordLength = UART_WORDLENGTH_8B;
  UartHandle.Init.StopBits   = UART_STOPBITS_1;
  UartHandle.Init.Parity     = UART_PARITY_ODD;
  UartHandle.Init.HwFlowCtl  = UART_HWCONTROL_NONE;
  UartHandle.Init.Mode       = UART_MODE_TX_RX;
  if (HAL_UART_Init(&UartHandle) != HAL_OK)
  {
    /* Initialization Error */
    Error_Handler();
  }
```

图 3-17 STM32F103 串口初始化

```
#define UART_BAUD_RATE  115200

UART_HandleTypeDef UART1_Handle;

/*****************************************************************************
 * function   : Uart_Init
 * Description: Uart Initiation.
 *****************************************************************************/
void Uart_Init(void)
{
    UART1_Handle.Instance                = UART1;
    UART1_Handle.Init.BaudRate   = UART_BAUD_RATE;
    UART1_Handle.Init.WordLength = UART_WORDLENGTH_8B;
    UART1_Handle.Init.StopBits   = UART_STOPBITS_1;
    UART1_Handle.Init.Parity     = UART_PARITY_NONE;
    UART1_Handle.Init.Mode       = UART_MODE_TX_RX_DEBUG;
    UART1_Handle.Init.HwFlowCtl  = UART_HWCONTROL_NONE;

    HAL_UART_Init(&UART1_Handle);
}
```

图 3-18 ACM32F0/FP0 的串口初始化

```
/* Private typedef -----------------------------------------------------------*/
/* Private define ------------------------------------------------------------*/
/* Private macro -------------------------------------------------------------*/
/* Private variables ---------------------------------------------------------*/
/* UART handler declaration */
UART_HandleTypeDef UartHandle;

/* Private function prototypes -----------------------------------------------*/
#ifdef __GNUC__
/* With GCC, small printf (option LD Linker->Libraries->Small printf
   set to 'Yes') calls __io_putchar() */
#define PUTCHAR_PROTOTYPE int __io_putchar(int ch)
#else
#define PUTCHAR_PROTOTYPE int fputc(int ch, FILE *f)
#endif /* __GNUC__ */
void SystemClock_Config(void);
static void Error_Handler(void);

/* Private functions ---------------------------------------------------------*/
```

图 3-19 STM32F103 去除的部分

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
Website： www.aisinochip.com

版本维护

版本	日期	作者	描述
V0.1	2021-01-07	Aisinochip	初始版
V0.2	2021-02-19	Aisinochip	核心开发板芯片为 F070R8T7
V0.3	2021-03-19	Aisinochip	增加KEIL工程移植章节
V0.4	2021-03-26	Aisinochip	STM32F103到航芯芯片的HAL移植
V0.5	2022-02-04	Aisinochip	以V2开发板为基础，更新本文档