AC7840x 开发板入门指南

文档版本：2.2

发布日期： 2023-08-28

修订信息

版本	日期	作者	修订说明
1.0	2022-05-13	AutoChips	文档初版
2.0	2022-08-15	AutoChips	硬件改版： LED 引脚更换 KEY1/KEY2 引脚更换 ADC 引脚更换，并添加一路 ADC Nor Flash 引脚更换 删除 Touch Pad
2.1	2022-12-16	AutoChips	更改文件名为《AC7840x 开发板入门指南》
2.2	2023-08-28	AutoChips	修改官网链接 更改图 1-1 中“EEPRO”为“EEPROM”

版权声明

本文档包含 AutoChips 公司的机密信息。禁止未经授权使用或披露本文档包含的信息。对因未经 AutoChips 公司授权而全部或部分披露此文档内容而给 AutoChips 公司带来的任何损失或损害，AutoChips 公司将追究责任。

AutoChips 公司保留对此处任何信息进行更改的权利，此处的信息如有变更，恕不另行通知。AutoChips 公司对使用或依赖此处包含的信息不承担任何责任。

本文档的所有信息均“按原样”提供，不提供任何形式的明示，暗示，法定或其他形式的保证。AutoChips 公司明确拒绝对适销性，非侵权性和针对特定用途的适用性方面的的所有暗示保证。AutoChips 公司对本文档可能使用、包含或提供的任何第三方软件不提供任何担保，并且用户同意仅向该等第三方寻求与此相关的任何担保索赔。AutoChips 公司对于根据用户规格或为符合特定标准或公开论坛而产生的任何交付物，也不承担任何责任。

文档目录

插图目录

表格目录

1 开发板平台简介

1.1 AC7840x 通用开发板示意图

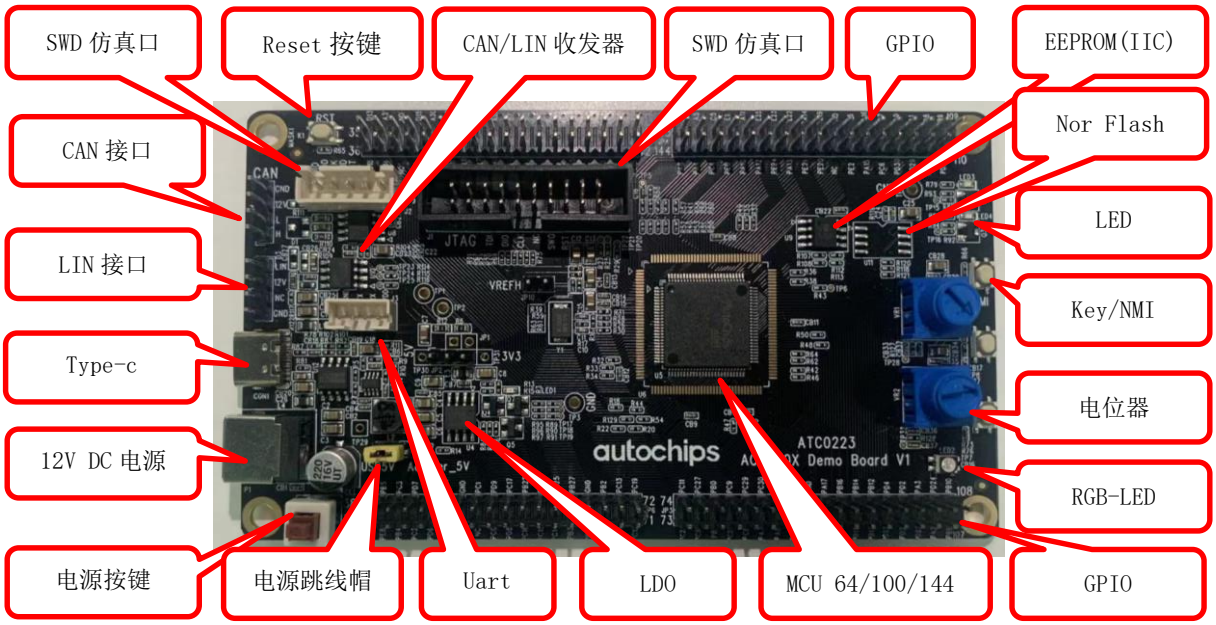

图 1-1 AC7840x 开发板资源图

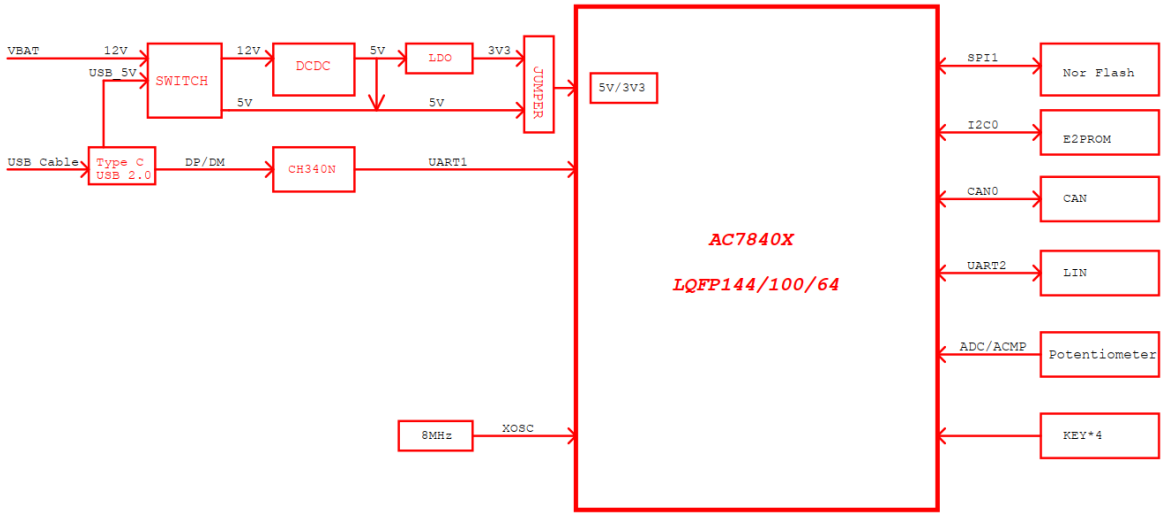

图 1-2 AC7840x 开发板结构框图

1.2 AC7840x 通用开发板资源说明

1. MCU

芯片规格：请参见《ATC_AC7840x_ReferenceManual_CH》表 2-1 AC7840x 模块概述

MCU 部分原理图：

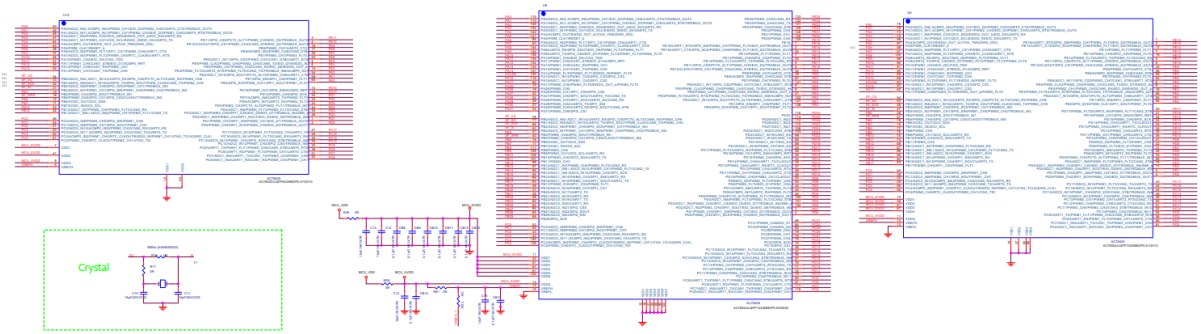

图 1-3 MCU 部分原理图

- AC7840x_demo 板支持两种封装型号的 MCU，分别为 100PIN 和 144PIN，高 PIN 脚对低 PIN 脚兼容，用户可以根据需求选择合适的 demo 板。
- 芯片可选内部 RC 振荡电路或外部无源晶体作为芯片的时钟源，AC7840x_demo 板的外部晶振选择 8M 无源晶体，与芯片内部的 HSI 时钟(8M)同步，在外部晶体出现故障时，切换为 HSI 时钟，可使主时钟的输出保持不变。
- 芯片的供电分为 VDDA 和 VDD，分别用于芯片内部的数字部分和模拟部分供电，二者的供电电压相差不能超过 0.3V。详细信息可以参考《ATC_AC7840x_Datasheet_CH》表 5-4 电压和电流操作额定值。

2. GPIO

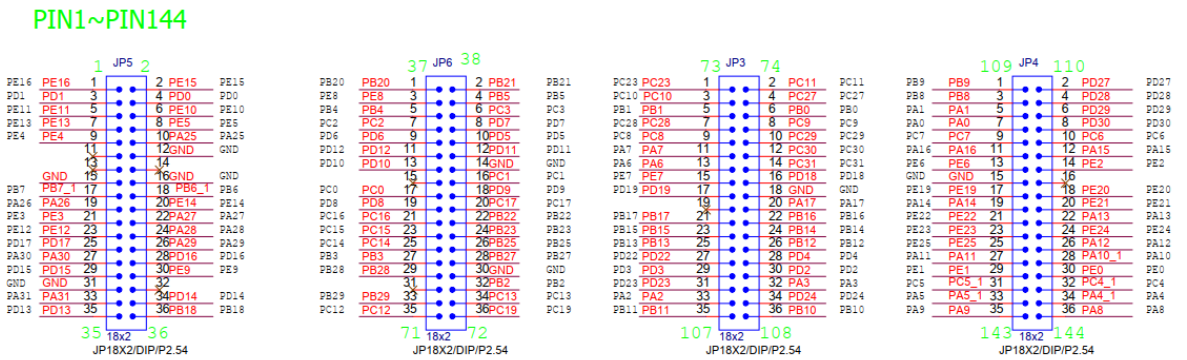

图 1-4 引出 GPIO 引脚示意图

图 1-5 接口及特殊功能引脚

3. 12V/5V/3.3V 电源

板载外部 12V-DC 电源输入，标准直流电源插座，开发板板载 DC-DC 芯片 MPQ4420 和 MPQ2019GN，可以将 12V 电源转换为 3.3/5V-DC，用于给开发板提供稳定的电源。5V 电源输入有两路选择，USB（+5V_1）和 MPQ4420 LDO（+5V_0）的输出，可以通过跳线帽 JP13 选择供电电源，如果 JP13 未接到任意供电电源，则 MCU 不会被供电。在使用了 LIN 通讯的情况下，建议使用外部电源供电，以保证 LIN 通信正常工作。

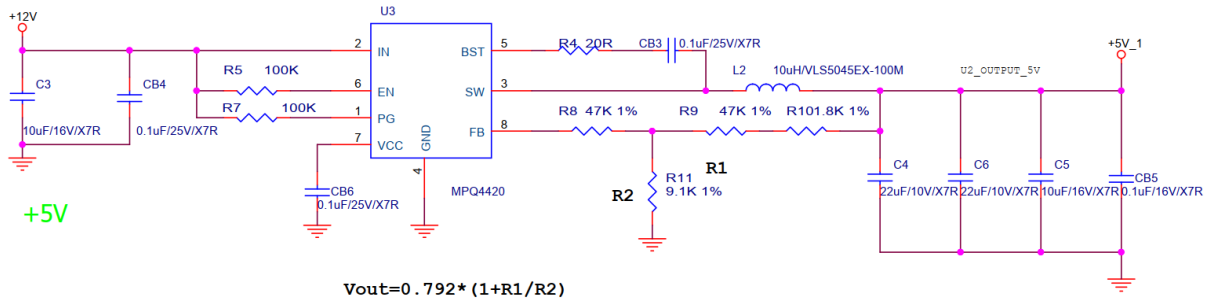

图 1-6 12V 转 5V 电路图

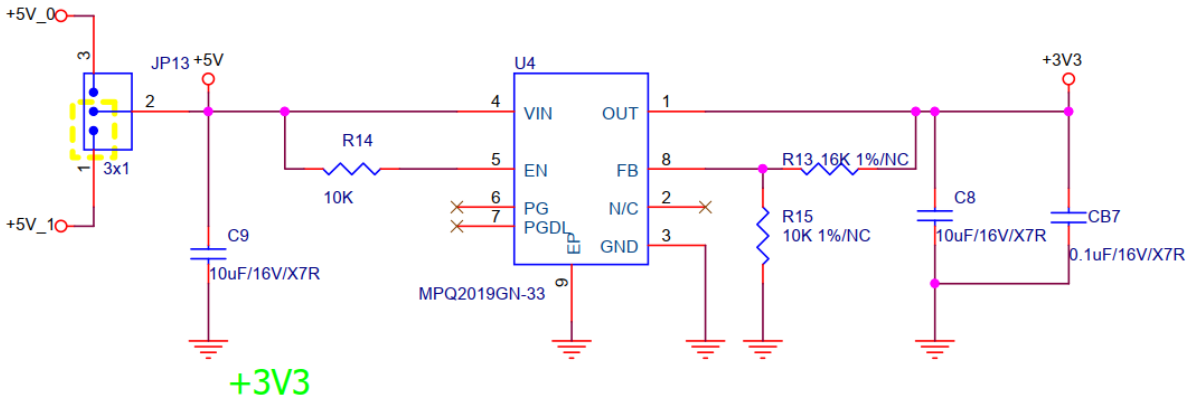

图 1-7 5V 转 3.3V 电路图

4. 5V/3.3V 选择口

AC7840x 支持 2.7~5.5V 宽电压输入，AC7840x_demo 板默认情况下是使用 3.3V 供电，若有要使用 5V 供电系统的，需要断开 L3 磁珠，使用跳线帽短接 JP2 的 1-2 引脚。若电压低于 LVR 寄存器设定值会触发 LVR 低电压复位，也可以通过 LVD 寄存器设置电压值，触发 LVD 中断。

表 1-1 LVD/LVR 设置

寄存器数值	LVD 阈值选择	LVR 阈值选择
0	低档 2.9v	高档 2.65V
1	高档 4.5v	低档 2.22V

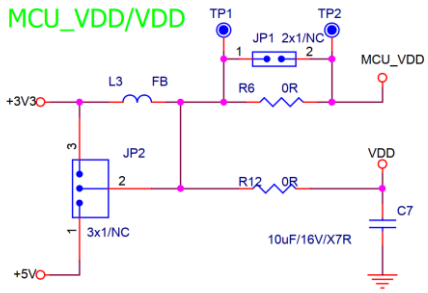

图 1-8 供电切换电路图

5. USB 转串口/UART1

USB 转串口座选用 Type-C 类型，AC7840x_demo 板选择用的 USB 转串口芯片为 CH340，win10 下可以直接识别到串口，win7 需要安装（CH340）驱动，用户也可以通过 USB 口对开发板提供 5V 供电。

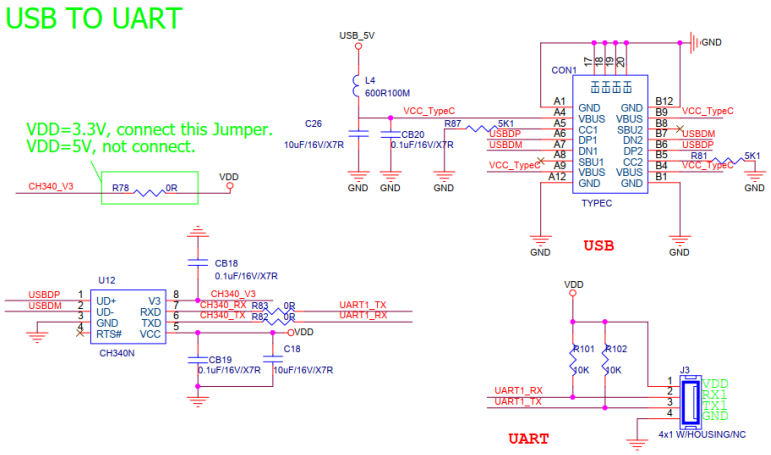

图 1-9 串口转 USB 电路图

6. 电源指示灯

电源指示灯使用板载 3.3V 供电，该指示灯 LED1 亮说明电源芯片都工作正常。

7. serial flash

预留 SPI 接口，通过 SPI1 通信，可用于扩展存储。

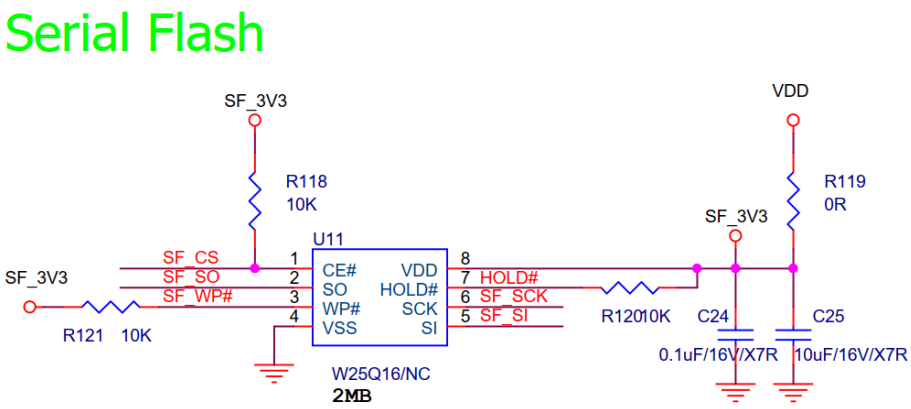

图 1-10 Nor Flash 原理图

8. 24C02 eeprom

预留 IIC 接口，通过 I2C0 通信，可用于扩展存储。

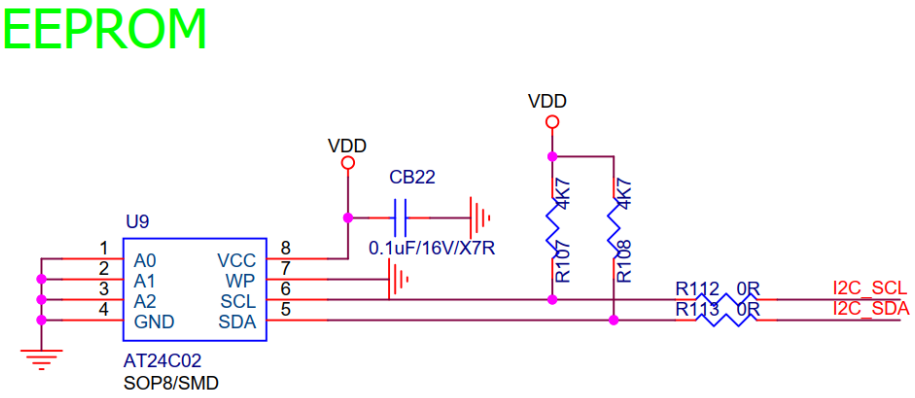

图 1-11 EEPROM 原理图

9. SWD 仿真口

预留 JTAG/SW 接口，分别有 5PIN 和 20PIN 调试底座，方便在线调试。

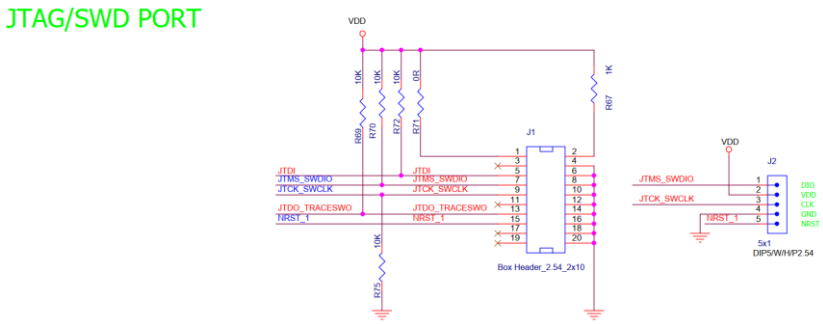

图 1-12 调试接口

10. RESET/NMI/KEY

单独引出芯片复位脚至按键 K1，按下可使芯片复位，低有效。

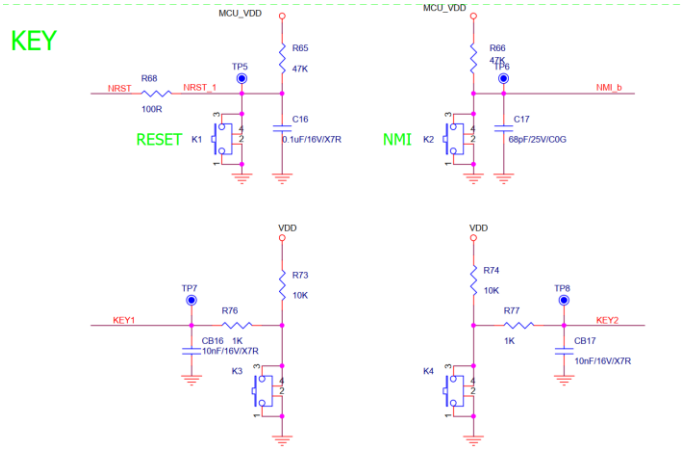

图 1-13 Reset/NMI/key 电路图

11. 2 个 LED/1 个 RGB LED

开发板提供了两个 LED（PC6/PC7）用于指示程序运行状态，两个灯都为绿色，两个 LED 分别通过一个三极管连接到 IO 口，IO 口通过输出高点亮 LED，同时 LED3/LED4 连接的 IO 口均具有 PWM 功能，可以通过 PWM 输出来调节 LED 亮度。

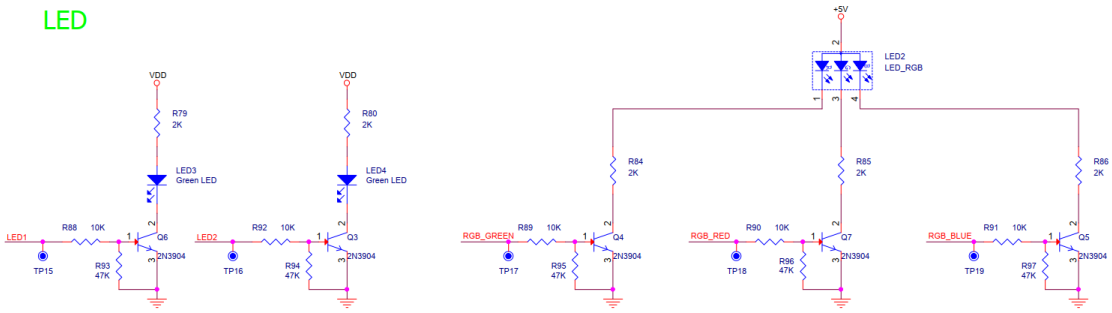

图 1-14 RGB LED/LED 电路图

12. 2 个按键

如图 1-13，单独引出芯片两个 GPIO 引脚至 K3(PA6)、K4(PE7)，两个按键可做输入检测用，也可用于 STOP1/ STOP2/ VLPS 模式下的唤醒。

13. NMI 按键

如图 1-13，单独引出芯片 NMI 引脚至 K2(PD3)，PD3 脚可以复用为 NMI 功能，为低电平时可触发 NMI 中断。

14. 启动模式配置

AC7840x 仅支持 eflash 启动。

15. 电位器

板载两路电位器，方便测试 ADC 模块和 ACMP 模块，两路电位器分别接到芯片 ADC 的两个不同通道 ADC0_IN10(PC2)和 ADC0_IN11(PC3)，通过旋钮电位器来改变对应引脚的输入电压，用于 ADC 检测和 ACMP 功能测试。

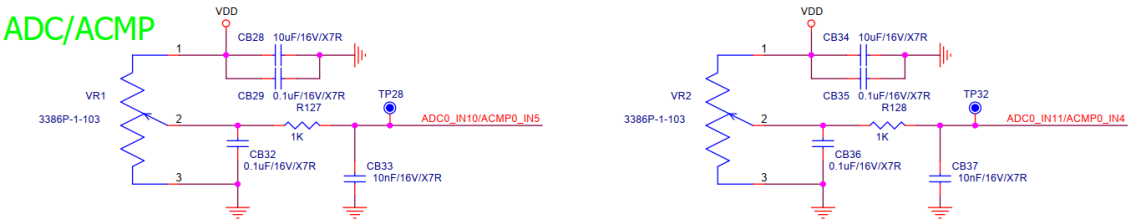

图 1-15 电位器原理图

1.3 开发板使用注意事项

为了便于用户更好的使用开发板，本小节总结了一些在使用过程中常见的一些需要注意的问题：

1. 如需测试 LIN 模块，必须在连接 12V 电源环境下测试，因为 LIN 控制器工作在 12V，仅使用 USB 5V 供电无法测试 LIN 模块。
2. 在测试 CAN/LIN 模块时，需要使能 CAN/LIN 收发器，给收发器休眠引脚正确的电平（CAN 收发器为低电平，LIN 收发器为高电平）。
3. AC7840x 的 demo 板在 12V 直流电源和 USB 5V 同时存在的时候，通过跳线帽 JP13 选择电源，如果 JP13 未接到任意供电电源，则 MCU 不会被供电。

2 芯片开发资料介绍

AC7840x 的 demo 板的软硬件资料，包括开发板硬件资料、芯片选型手册、参考手册、数据手册、CMSIS 驱动包，及基于 demo 板的应用例程，后续都会释放到官网上，链接如下：

https://www.autochips.com/developersr.html

基于开发 AC7840x 芯片的相关开发工具也会释放到官网，链接如下：

https://www.autochips.com/developersr.html?ty=73

AutoChips 技术支持论坛（21ic），链接如下：

https://bbs.21ic.com/iclist-864-1.html

3 MDK5 软件入门

3.1 新建基于固件库的 MDK5 工程

在搭建基于固件库的 MDK5 工程之前，需要先准备如下资料：

- MDK5 开发环境安装，建议使用 5.23 版本（除 5.30）及以上版本。MDK5 安装包可以从 keil 官网下载：https://www2.keil.com/mdk5/install
- AutoChips.AC7840x_DFP.x.x.x.pack 驱动包安装（版本会基于 pack 版本有所变动，待量产后会同步至官网）下载链接如下：

  https://www.autochips.com/developersr.html

新建工程之前，先建立一个新的文件夹，用于存放工程文件，如图 3-1 所示。

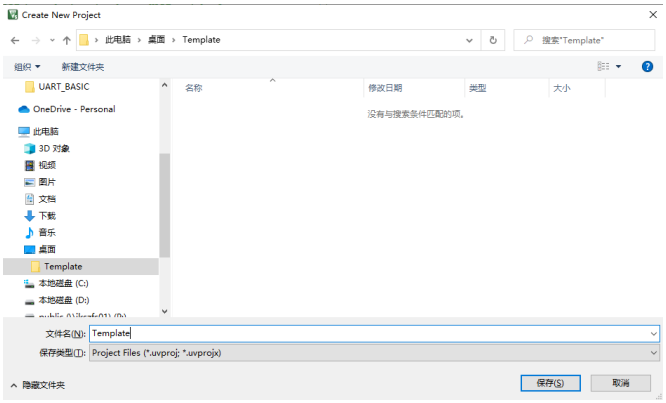

图 3-1 设置工程保存路径

打开 MDK 软件，按照以下步骤 Project->New uVision Project 新建工程，在弹出的界面定位到之前建立的 Template 文件加下，然后在该文件夹下新建 Project 文件。

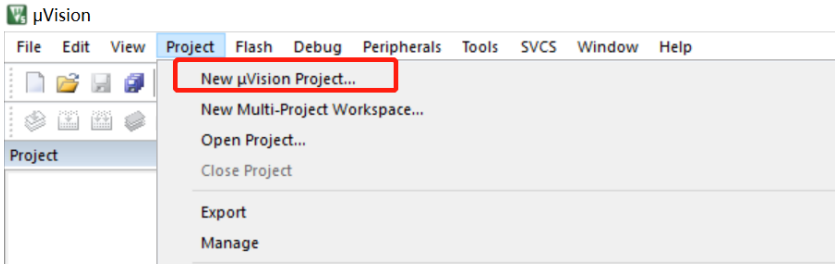

图 3-2 新建工程

选择所使用的芯片型号：在完成上述步骤后，接下来会弹出一个芯片型号选择界面，如图 3-3 所示，按照 AutoChips->AC7840x Series-> AC7840x -> AC78406YGLA 步骤选择 demo 板对应的芯片型号（如果使用的其他型号芯片，选择对应的型号即可）。

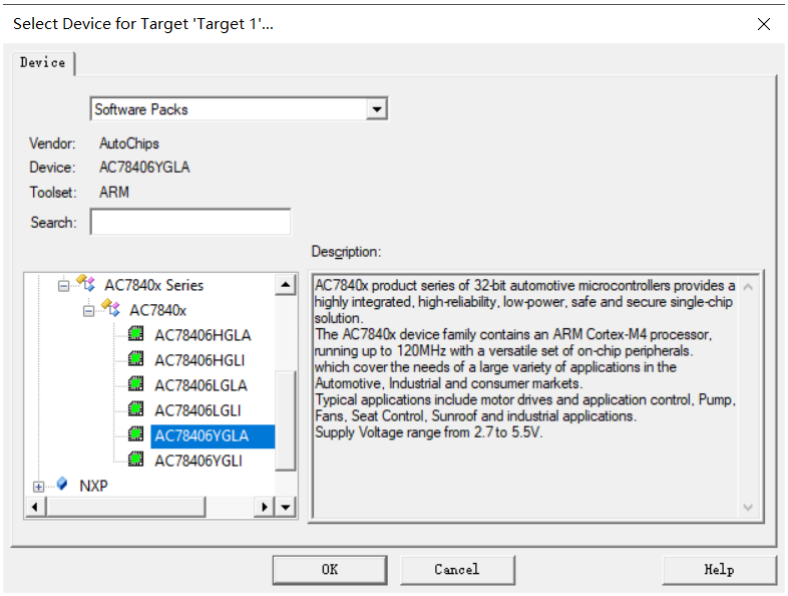

图 3-3 选择芯片型号

注意：只有正确安装了上面的 AutoChips.AC7840x_DFP.x.x.x.pack（版本会基于 pack 版本有所变动）驱动包，才能找的到 AC7840x 的芯片型号，如果没有该选项，请重新安装改驱动包。

完成芯片选型后，会弹出 Manage Run-Time Environment 窗口，如图 3-4 所示，可以在该对话框中添加启动文件、内核文件及所需要的外设驱动。如果 driver 没有正确勾选，会在 Validation Output 窗口提示有冲突的驱动，按提示勾选就可以。

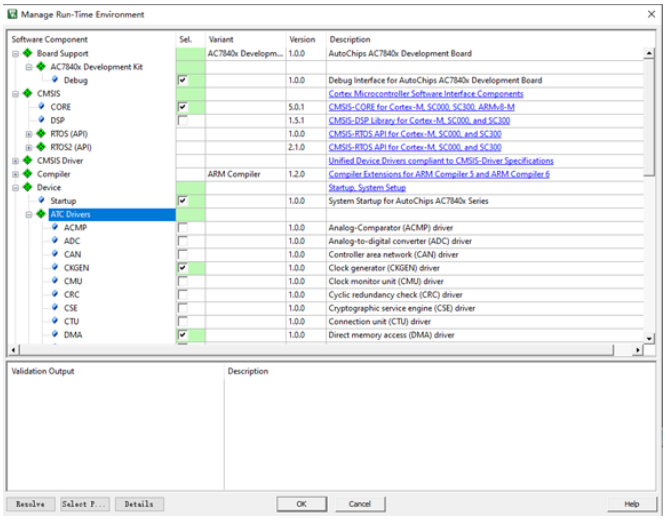

图 3-4 Manage Run-Time Environment 界面

说明：这是 MDK5 新增的功能，方便用户添加自己需要的组件，如果想要自己添加驱动等文件，可以直接点 Cancel

此次开发入门例程仅示范简单的串口打印功能，需添加 Debug 及 UART 等.c 文件，添加完成后可在 Project 窗口显示已添加的 driver。

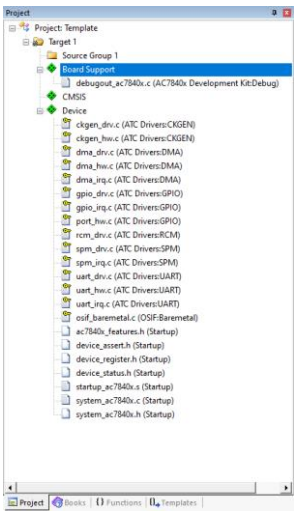

图 3-5 工程目录

注意：图中部分文件有钥匙样式图案，提示该文件为只读属性文件，这是因为在 RTE 环境下新建工程添加的驱动是从安装的 pack 目录下引用的，在该环境下新建的工程共用一份 driver，如果修改的话，会影响到其他工程。如需修改，需将相应的文件 copy 到工程本地路径下，修改只读属性，然后通过手动添加的方式加入到工程中。并去掉 RTE 环境中的文件，避免重复定义。

完成上述步骤后，开发环境基本已经搭好，接下来添加项目的开发代码就可以了。在 Template 文件夹下在新建一个 User 文件夹，新建一个 main.c 文件，然后将其添加到工程中的 User 文件夹下。

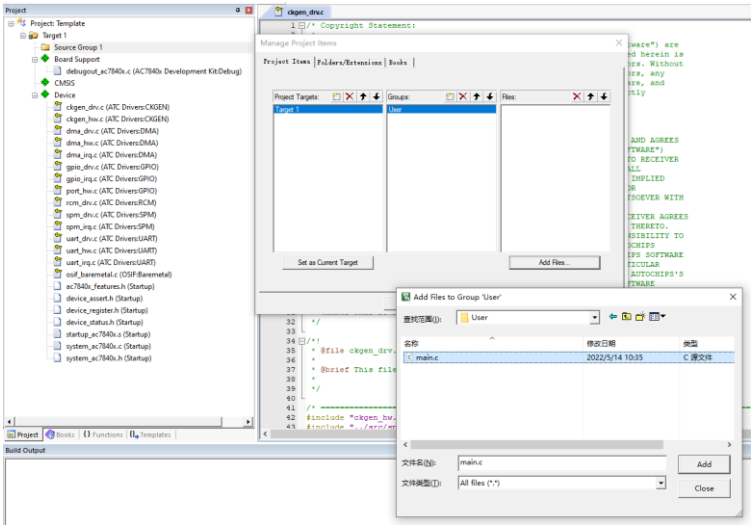

图 3-6 添加 main 文件

在 main.c 文件中编写 Uart 打印的代码，首先根据外设需求设置时钟（clock_config.c 文件作用为时钟使能，会随 demo 一起提供，用户也可以根据需求在各个模块的初始化过程中使能），调用 Initdebug 接口，编译可以使用 printf 函数，代码如下。

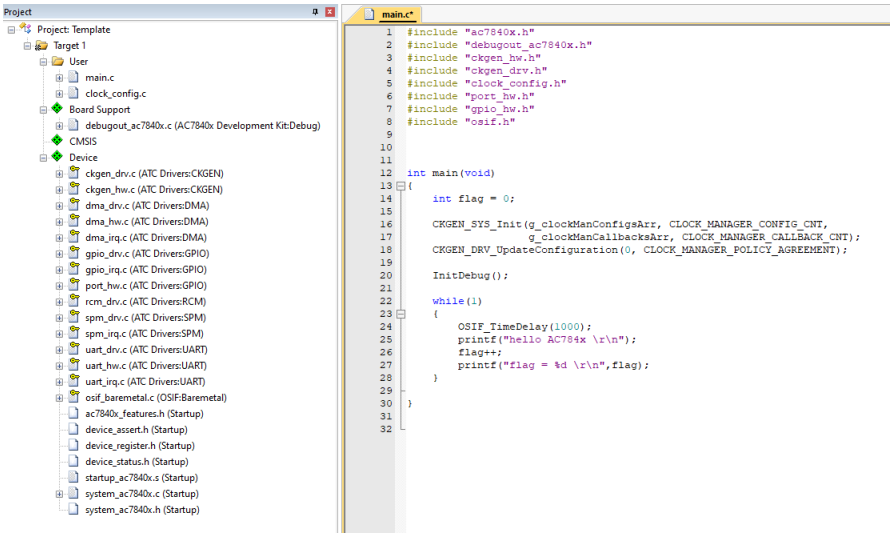

图 3-7 输出代码

附代码：

```
#include "ac7840x.h"
#include "debugout_ac7840x.h"
#include "ckgen_hw.h"
#include "ckgen_drv.h"
#include "clock_config.h"
#include "port_hw.h"
#include "gpio_hw.h"
#include "osif.h"

int main(void)
{
    int flag = 0;
    CKGEN_SYS_Init(g_clockManConfigsArr, CLOCK_MANAGER_CONFIG_CNT,
                   g_clockManCallbacksArr, CLOCK_MANAGER_CALLBACK_CNT);
    CKGEN_DRV_UpdateConfiguration(0, CLOCK_MANAGER_POLICY_AGREEMENT);

    InitDebug();

    while(1)
    {
        OSIF_TimeDelay(1000);
        printf("\r\n hello AC7840x \r\n");
        flag++;
        printf(" flag = %d \r\n",flag);
    }

}
```

3.2 程序的下载与调试

完成新工程的创建后，本小节介绍如何将编译成功的 demo 程序下载到开发板上并进行调试。需要准备以下环境：

- AC7840x_demo 板
- 12V 直流电源
- 调试器（ARM 仿真器都可以，包括但不仅限于：ATC-LINK、JLINK、ULINK、ST-LINK）
- 串口转接板
- PC

按照图 3-8 的方式连接好硬件便可以开始调试（如使用板载的 CH340 驱动芯片，可去掉串口转接板的连接），此次调试选用的调试器为原厂自主开发的 ATC-LINK。

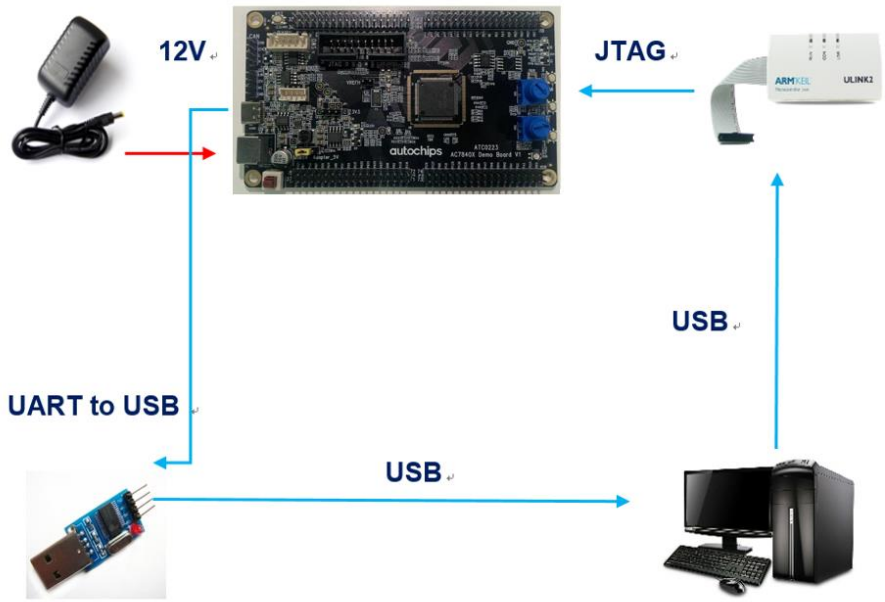

图 3-8 调试硬件连接图

注意： win10 环境下 ATC-LINK 免驱，win7 需要安装驱动，可以在 AutoChips 官网上下载 ATC-LINK 的驱动包。下载链接如下:https://www.autochips.com/developersr.html

连接好仿真器后，点击工程界面上面的魔术棒，然后在 Debug 界面，选择所使用的仿真器型号，ATC-LINK 对应的仿真器型号选择 CMSIS-DAP Debugger。

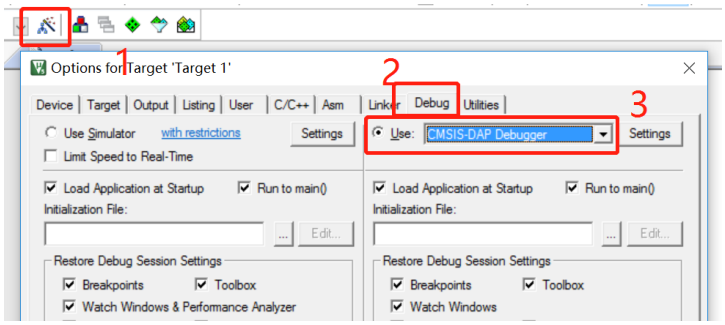

图 3-9 debug 选项卡设置

点击 Setting 按钮，设置一些仿真参数，Port 可以选择 SW 或 JTAG 模式。

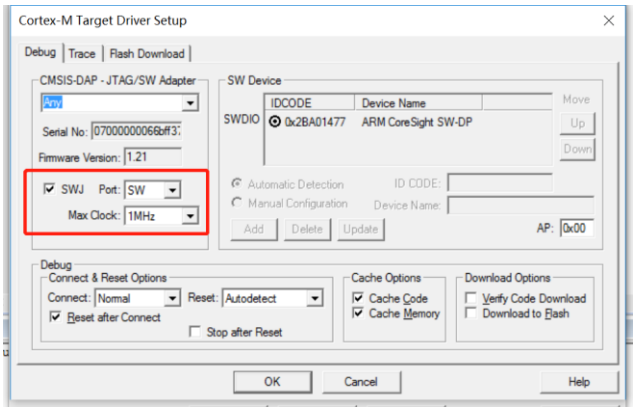

图 3-10 debug 参数配置

然后进入 Flash Download 选项卡，配置芯片下载算法和动作，一般选择默认配置，建议勾选 Reset and Run，这样可以在下载程序后，软件自动复位运行。不勾选则下载后程序不会运行，需要手动按下复位按钮程序才会运行。

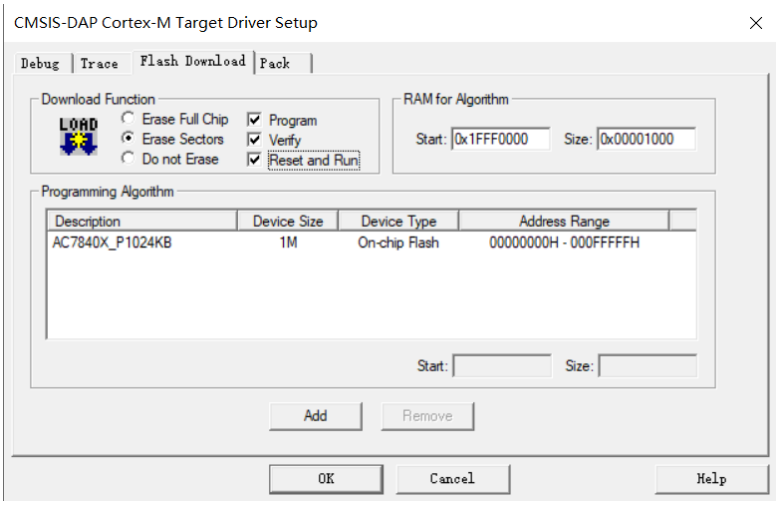

图 3-11 Flash Download 选项卡设置

配置完成后点击 OK，然后点击 Load 按钮，将程序下载到开发板中：

图 3-12 下载程序

程序下载成功后，就已经开始运行了，接上串口线便可以在串口上位机看到输出结果。

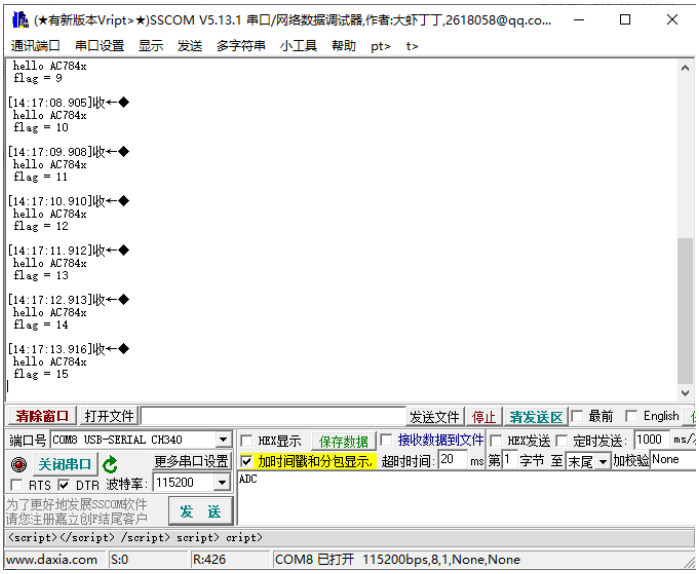

图 3-13 输出结果

点击 Debug 按钮进入仿真，然后通过左上角的按钮，可以控制芯片复位，全速运行，停止，单步运行、运行到断点处等操作。

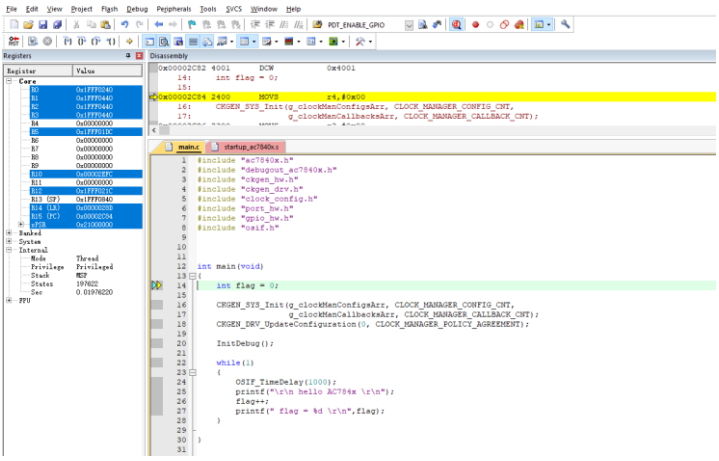

图 3-14 进入仿真

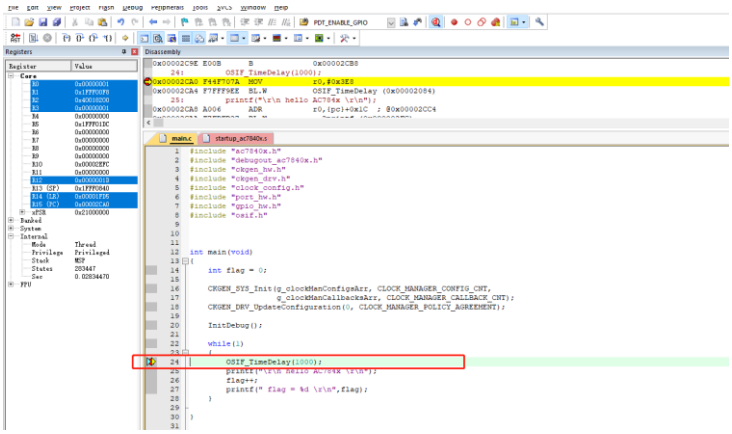

图 3-15 设置仿真断点

3.3 MDK5 仿真调试技巧

3.3.1 变量监控

通过 Watch 界面，可以对一些变量进行监控，我这里添加了 flag 变量的监控：

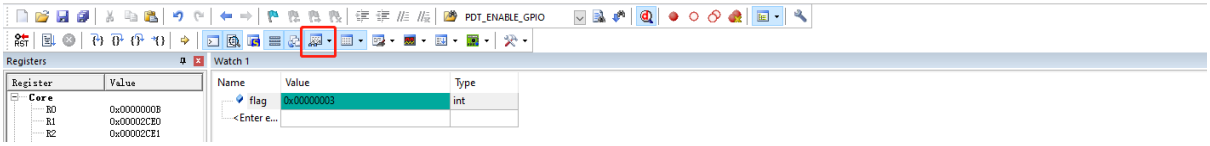

图 3-16 变量监控

> **注意：**变量监控最好对全局变量进行监控，局部变量或静态变量无法被全局监控。

3.3.2 查看 memory

通过 memory 窗口，可以查看任意有效的地址数据：

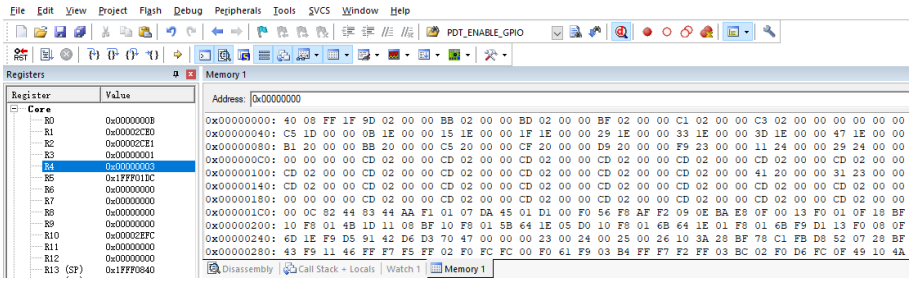

图 3-17 查看 memory

3.3.3 查看外设寄存器

通过 system viewer windows，可以查看外设寄存器的值：

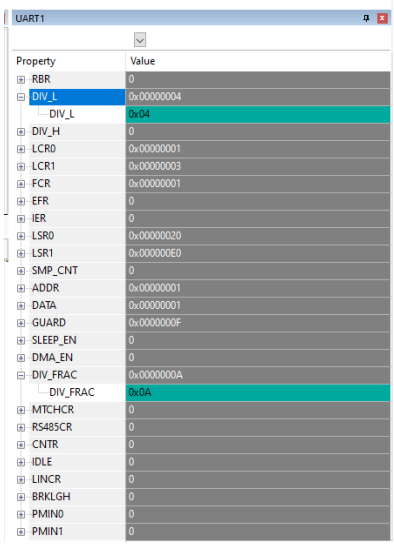

图 3-18 查看外设寄存器

3.3.4 查看 CPU 寄存器

通过 Register windows，可以查看 CPU 寄存器值

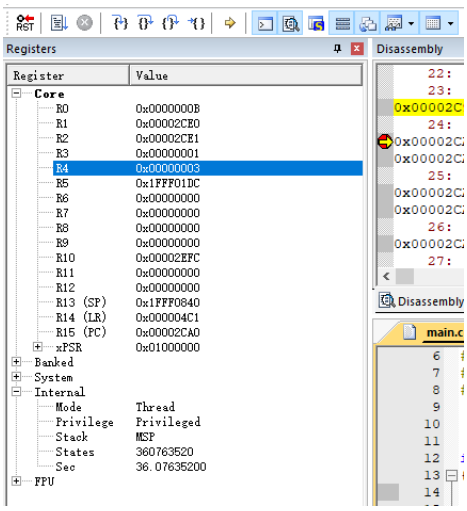

图 3-19 查看 CPU

3.3.5 查看函数嵌套

通过 call stack windows，可以看到函数调用的嵌套关系。

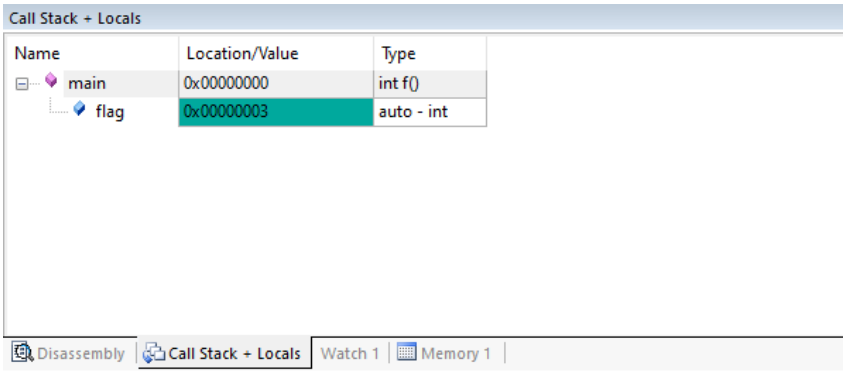

图 3-20 函数调用嵌套

通过此窗口，可以看到当前代码停在哪个函数，以及从哪个函数进入的这个函数，逐级往上。在调试程序的时候较为重要。例如在程序进入 hardfault handler 时可通过此窗口，查询是从那个函数跳转进入 hardfault handler 的。

4 AC7840x 开发基础知识介绍

4.1 系统架构

AC7840x 是根据 ISO 26262 开发的，具有针对 ISO26262 ASIL-B 完整性级别的综合安全概念。系统架构图如下所示。

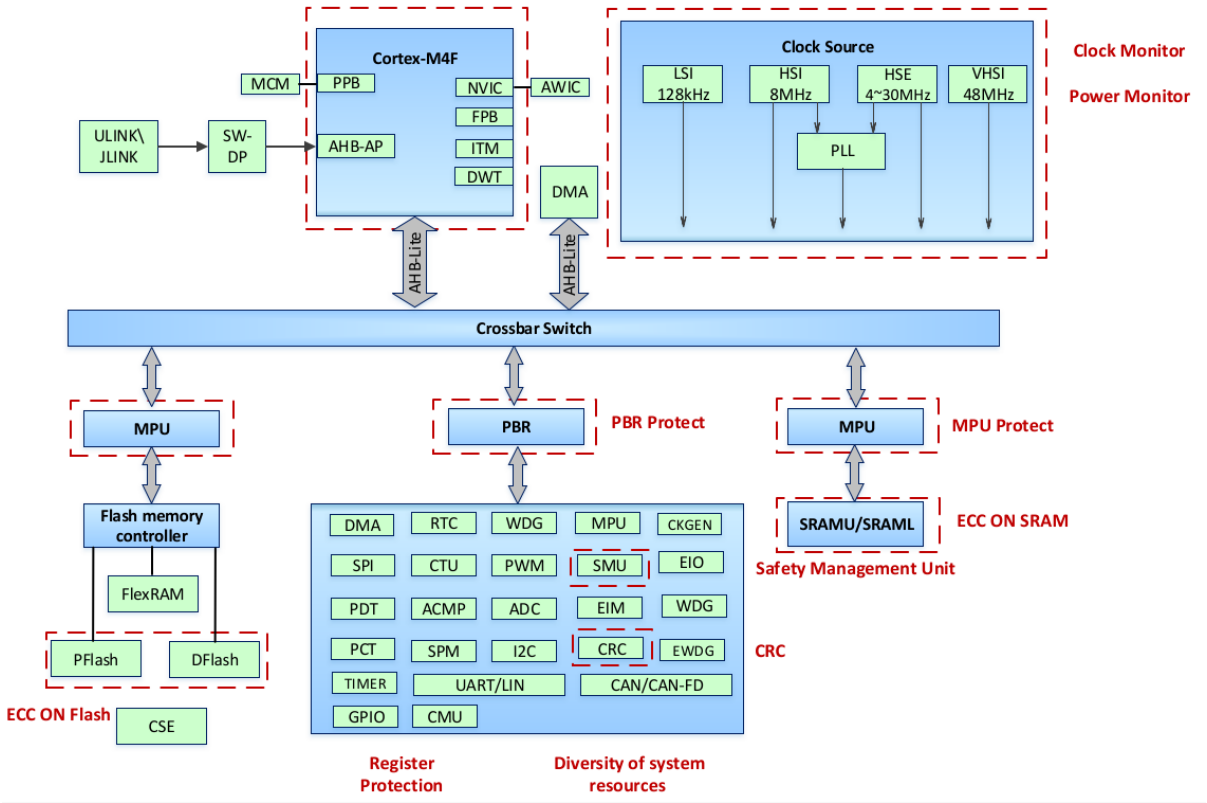

图 4-1 AC7840x MCU 系统架构

4.2 地址分配

AC7840x 采用的 cortex-M4 内核，仅支持 eFlash 启动，数据总线为 32 位宽度，程序存储器(P-Flash)、数据存储器(D-Flash)、寄存器和输入输出（I/O）接口统一编址在 4G 字节的线性地址空间里。详细分配请参见《ATC_AC7840x_ReferenceManual_CH》表 4-2 存储器映射表。

4.3 时钟

模块包含一个锁相环（SPLL）作为时钟源，SPLL 可由内部或外部基准时钟作为基准。该模块可以控制此 SPLL 时钟或内部/外部基准时钟之一作为 MCU 系统时钟源，生成所有模块的时钟和频率。

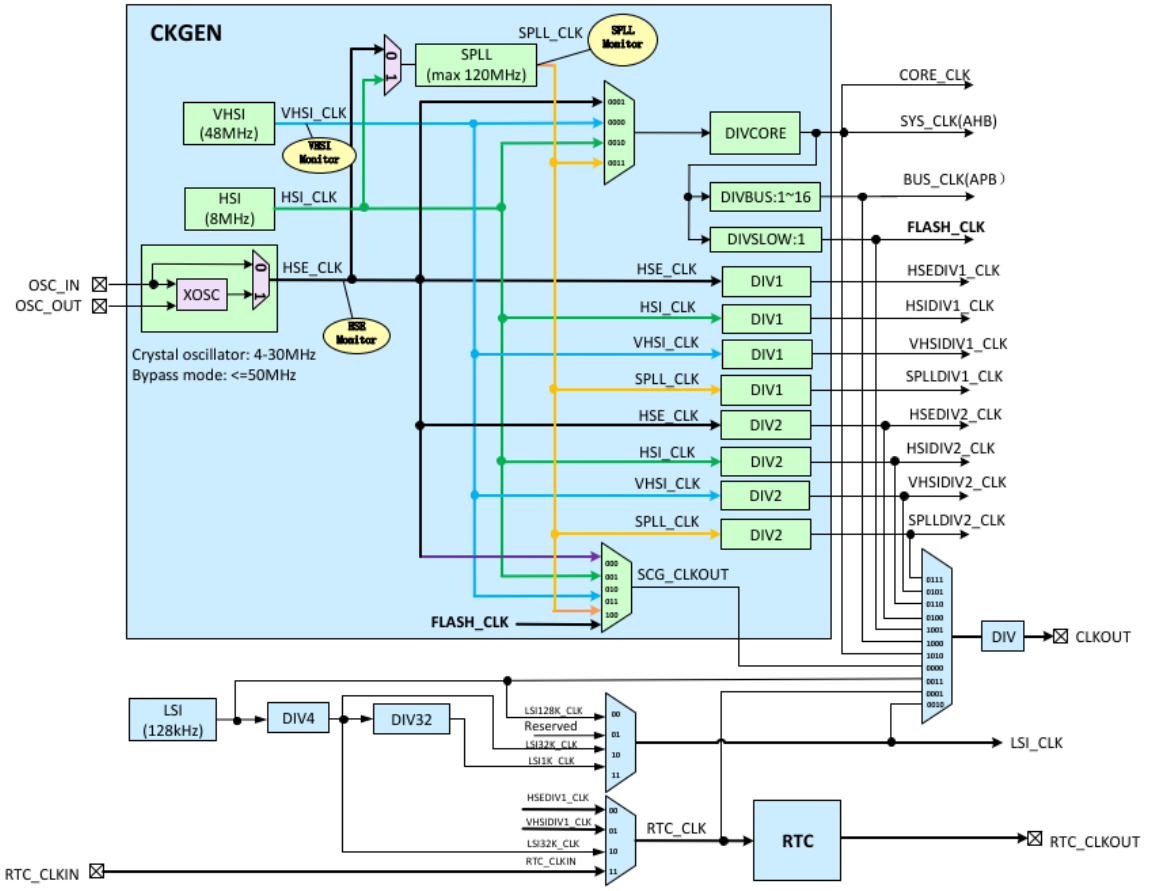

图 4-2 时钟树

AC7840x 共有 5 个时钟源：

- 高速内部时钟（HSI）：内部 RC 振荡器提供 8MHz 时钟源
- 超高速内部时钟（VHSI）：内部 RC 振荡器提供 48MHz 时钟源，上电或复位默认的系统时钟源
- 外部高速时钟（HSE）：外部 OSC 提供 4MHz ~30MHz 晶振
- 低速内部时钟（LSI）：内部低速 RC 振荡器提供 128KHz 时钟源
- 系统 PLL 时钟（SPLL）：提供高达 120MHz 的高速时钟

图 4-2 指出 AC7840x 的系统时钟和外设时钟是单独分开的，需要独立的使能。配置时可参考官方提供的 demo 工程中的 clock_config.c 和 clock_config.h 文件。

4.4 端口复用和重映射

AC7840x 有丰富的外设资源，为了方便用户的硬件设计，AC7840x 的 GPIO 引脚有 8 种复用功能，不同的外设可以被复用到不同的 GPIO 引脚上。具体 GPIO 支持的复用功能可以查阅《ATC_AC7840x_ReferenceManual_CH.pdf》表 19-1 GPIO 复用功能描述。

表 4-1 复用功能配置

64LQFP	100LQFP	144LQFP	引脚名称	功能 0	功能 1	功能 2	功能 3	功能 4	功能 5	功能 6	功能 7
35	55	80	PC9	HIGH_Z	GPIO	UART1_TX	PWM1_FLT1	PWM5_CH0	CAN0_STB(0)	UART0_RTS	HIGH_Z
36	56	81	PC8	HIGH_Z	GPIO	UART1_RX	PWM1_FLT0	PWM5_CH1	HIGH_Z	UART0_CTS	HIGH_Z

将 PC8，PC9 脚设置为 UART 功能示例代码如下：

```
GPIO_DRV_SetMuxModeSel(PORTC,9UL, PORT_MUX_ALT2);
GPIO_DRV_SetMuxModeSel(PORTC,8UL, PORT_MUX_ALT2);
```

4.5 中断优先级管理

AC7840x 支持 0~16 级中断优先级配置，0 级最高。可以通过内核头文件 core_cm4.h 提供的 NVIC_SetPriority 函数来设置中断优先级，例如我想设置 UART1 中断优先级为 3：

```
NVIC_SetPriority(UART1_IRQn, 3);
```

5 AC7840x 驱动库简介

5.1 驱动库结构

AC7840x 的 CMSIS 包 driver 架构因支持 Autosar 的开发，在之前工作的基础上做了一些更进一步的优化，同时也方便用户在不同编译器上的适配，AC7840x 的驱动架构图如下图所示。

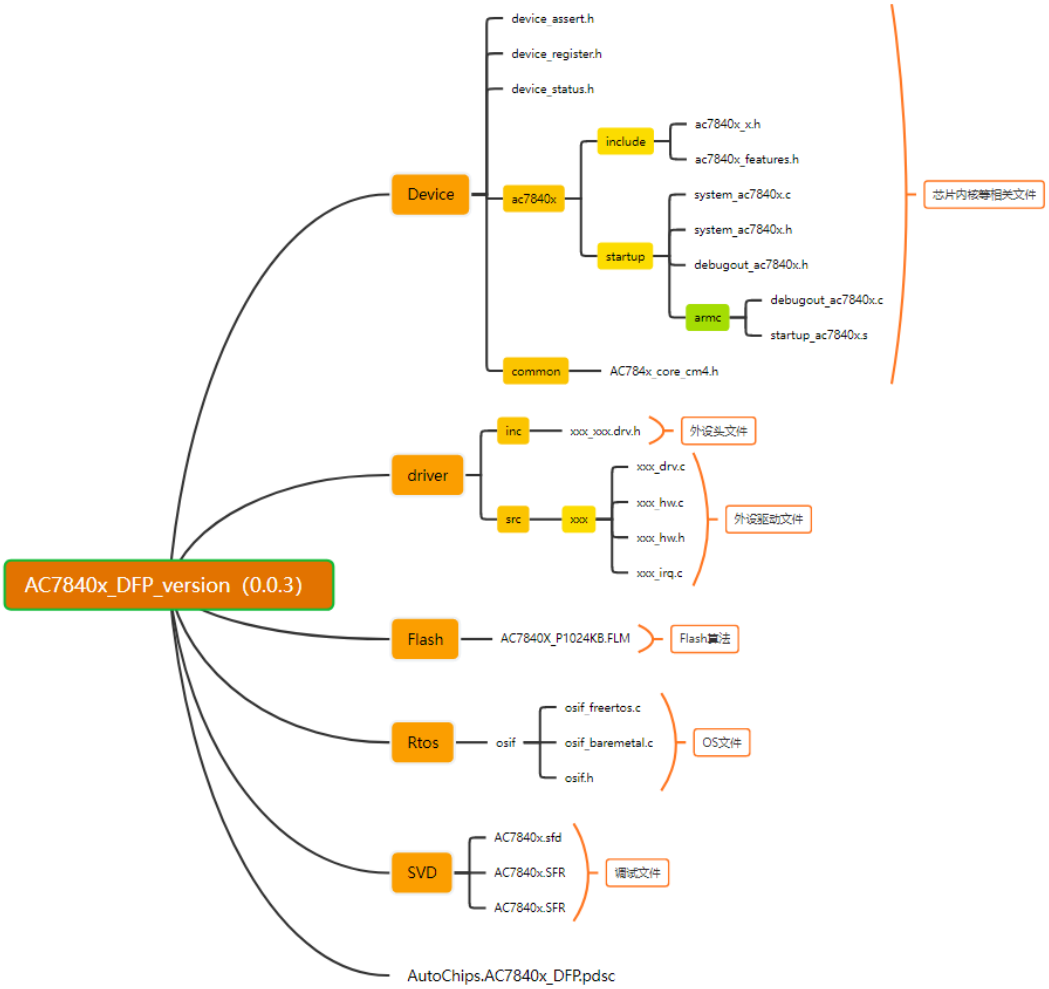

图 5-1 AC7840x 驱动架构图

5.2 Device 文件夹介绍

5.2.1 启动文件

启动文件 startup_ac7840x.s 存放在 Device->ac7840x->startup->armc 下，在启动文件中定义了栈空间的大小，还有中断向量表等相关相关定义。在芯片上电后，会首先执行 Reset_Handler 函数，在

该函数中会调用 SystemInit 函数对 MCU 进行初始化，然后跳转到_main 库，最后在_main 库中会跳转到 main 函数中运行。

5.2.2 debugout_ ac7840x.c 文件

在 Device->ac7840x->startup->armc 路径下，存放了 debugout_ac7840x.c 文件，可用于实现格式化打印输出的重定向，通过调用其初始化 InitDebug()函数，将 printf 输出重定向到芯片的 UART1 输出（PC9，PC8）。

⚠**注意：**如果代码中使用到了 printf 相关的函数，必须要包含此文件，或者自己实现重定向功能，否则会导致程序运行不正常。

5.2.3 系统文件

在 Device->ac7840x->startup 文件夹下存放有 system_ac7840x.c 文件，该文件实现了对芯片系统寄存器相关的操作， 例如读取芯片的专属 UUID。

5.2.4 寄存器定义

芯片所有的寄存器都定义在 ac7840x.h 这个文件中，如果用户不想使用官方提供的驱动库，而想自己通过寄存器操作的方式来实现功能，只需要引入此文件即可。

5.3 外设驱动

如图 5-1 介绍，外设基本包含三个.c 文件，具体.c 文件功能如下：

xxx_drv.c: provides xxx integration functions.

xxx_hw.c: provides xxx hardware access functions.

xxx_irq.c: provides xxx interrupts handler functions.

5.3.1 中断回调机制

为了便于对中断的管理，每个驱动文件中已经实现了对应外设的中断 IRQ handler，因此，在编写应用代码的时候，不需要自己去实现中断入口函数。下面是 UART 驱动代码中的中断入口函数：

```
/*!
 * @brief UART1 interrupts handler
 *
 * @param[in] none
 * @return none
 */
void UART1_IRQHandler(void)
{
    if (g_uartIsr[1U])  /* UART or LIN IRQHandler */
    {
        g_uartIsr[1U](1U);
    }
```

```
    else
    {
        /* Do nothing */
    }
}
```

考虑到应用上会需要在中断中执行一些操作以及判断一些事件标志。因此提供了以回调函数的方式实现应用层执行中断代码。中断回调函数的原型统一如下：

```
uart_callback_t UART_DRV_InstallRxCallback(uint32_t instance,
uart_callback_t function, void *callbackParam)
```

5.4 UART 初始化示例

UART 初始化流程如图 5-2 所示。

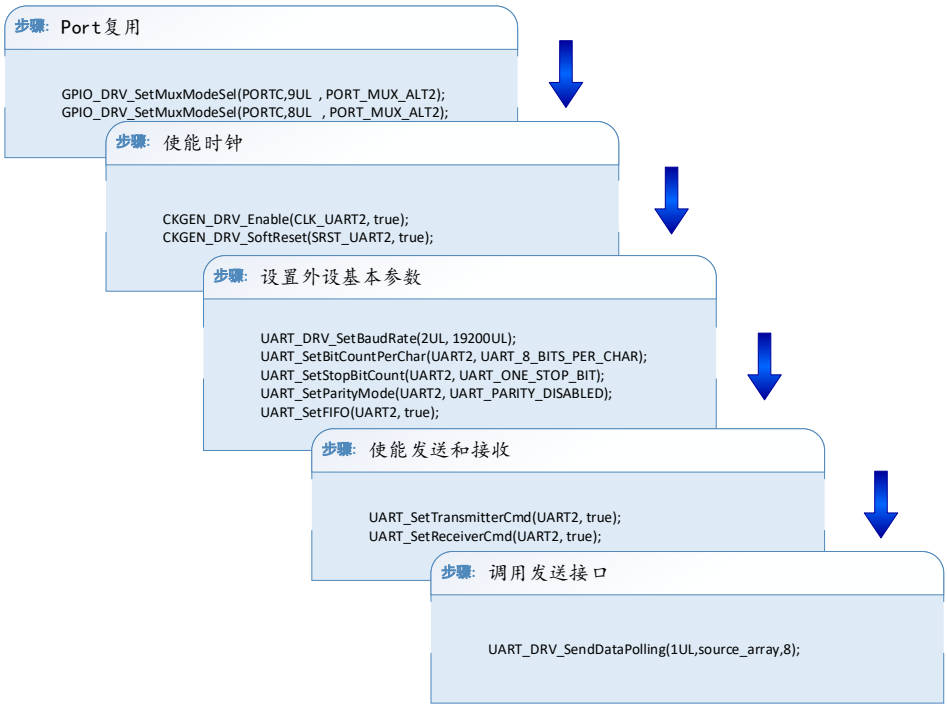

图 5-2 UART 初始化流程

示例代码：

```
#define UART1_INDEX (1U)

uart_user_config_t s_uartUserConfig = {   //uart 初始化结构体

    .baudRate = 115200U,
    .transferType = UART_USING_INTERRUPTS,
    .parityMode = UART_PARITY_DISABLED,
    .stopBitCount = UART_ONE_STOP_BIT,
    .bitCountPerChar = UART_8_BITS_PER_CHAR,
```

```
    .rxDMAChannel = 0U,
    .txDMAChannel = 0U,
};

UART_DRV_Init(UART1_INDEX,&s_uartState,&s_uartUserConfig); // 调用初始化接口

UART_DRV_SendDataPolling(UART1_INDEX,source_array,8); //  调用发送接口
```