芯海科技
CHIPSEA

EVB-32F03X-START

开发板用户手册

REV 1.1

芯海科技（深圳）股份有限公司
地　址：深圳市南山区蛇口南海大道1079号花园城数码大厦A座9楼
电　话：+(86 755)86169257　　传　真：+(86 755)86169057
网　站：www.chipsea.com　　邮　编：518067
微信号：芯海科技

版本历史

历史版本	修改内容	版本日期
REV 1.0	初版生成	2019-03-12
REV 1.1	基于新版 SDK 生成新的示例代码	2019-11-14

目录

1 使用入门

1.1 系统要求

EVB-32F03X-START 是芯海科技（深圳）股份有限公司开发一款基于 Cortex-M0 的开发板，使用 EVB-32F03X-START 开发板须至少达到以下环境要求：

- Windows PC (Windows 7/10)
- 1 个电源线（或者 1 条 USB A 口转 Micro-B 口数据线），这个线用于给开发板供电
- 1 个 J-LINK（或者 DAP-Link）调试器，用于程序的下载和调试
- 1 个 USB 转串口下载器

使用该开发板，请参照下图所示，连接 EVB-32F03X-START V1.1 开发板与 PC 机。该开发板后续文档简称 EVB-32F03X-START 开发板。

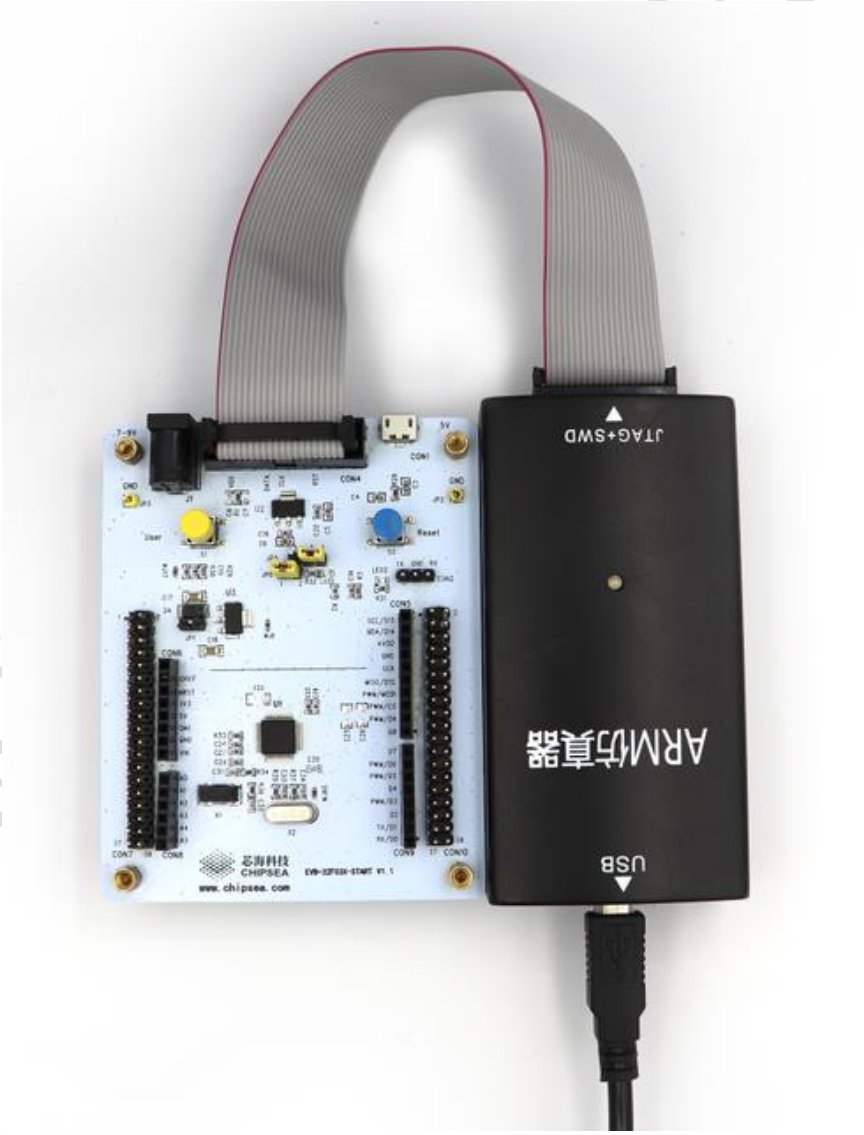

图 1.1 硬件连接图

2 软件开发工具安装

2.1 MDK 软件获取

按照授权的不同，MDK-ARM 分为 MDK-Lite、MDK-Basic、MDK-Standard 和 MDK-Professional 四个不同的版本。其中 MDK-Lite 可以免费使用，剩下的三个版本需要序列号或授权许可密钥。本文以 MDK-ARM Version 5.26 为例 。

MDK 软件的下载步骤如下：

1. 访问 Keil 官网 www.keil.com，单击完 “Product Downloads” 项的窗口中列出 Keil 公司的四种软件开发工具，单击“MDK-ARM”项。
2. 完成联系方式的填写，并单击“MDK526.EXE”项（如下图所示）启动下载任务，即可完成对 MDK 软件下载。

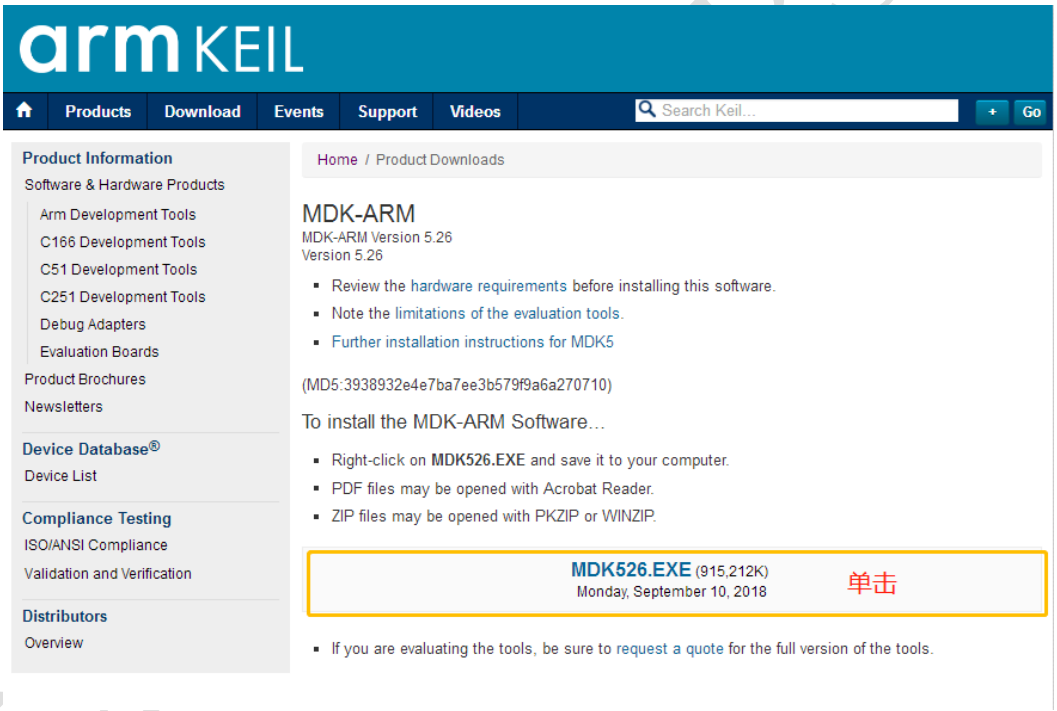

图 2.1 下载 MDK 软件

3. 在安装完 MDK-ARM 软件后，还需下载微控制器的软件安装包(software pack)，访问 http://218.17.109.235/FolderPublish.aspx?code=B5f5a0f39fefb4af78785189452d2effb，即可进入软件安装包(software pack)下载页面。
4. 下载目标器件所支持的软件安装包(software pack)。以 EVB-32F03X-START 开发板为例，则需下载 “Chipsea.CS32F0xx_DFP.1.0.0.pack” F0 系列软件安装包(software pack)。

2.2 MDK 软件安装

MDK-ARM V5.26 **软件具体的安装步骤如下：**

1. **安装之前，如果本地电脑存在** Keil C51、ADS **软件建议先卸载，否则易导致** MDK-ARM **软件无法正常使用**。
2. **以管理员运行安装包（如** mdk526.exe），**弹出许可证界面，勾选同意**。
3. **弹出软件安装时的安装路径，保持默认的设置并单击**“Next”**按钮（注意** Keil V5 **版本的** MDK-ARM **软件的安装路径分为**“Core”**和**“Pack”**两个部分）**。
4. **完成软件使用者信息填写，点击**“Next”**按钮**。
5. **软件拷贝结束后会提示安装设备驱动程序，勾选始终信任的方框，完成安装**。

软件安装包(software pack) **具体的安装步骤如下**:

1. **在线安装**
   - MDK-ARM **软件会自动连接互联网并开始下载软件安装包**(software pack)**（注意使用此方法下载和安装软件安装包**(software pack)**需要较长时间）**。
2. **离线安装：**
   - MDK-ARM **软件运行后。单击工具栏上的**“Pack Installer”**按钮，启动安装包管理器（如图** 2.2）。
   - **通过**“File”**菜单中选择**“Import”**项，导入** “Chipsea.CS32F0xx_DFP.1.0.0.pack” 。

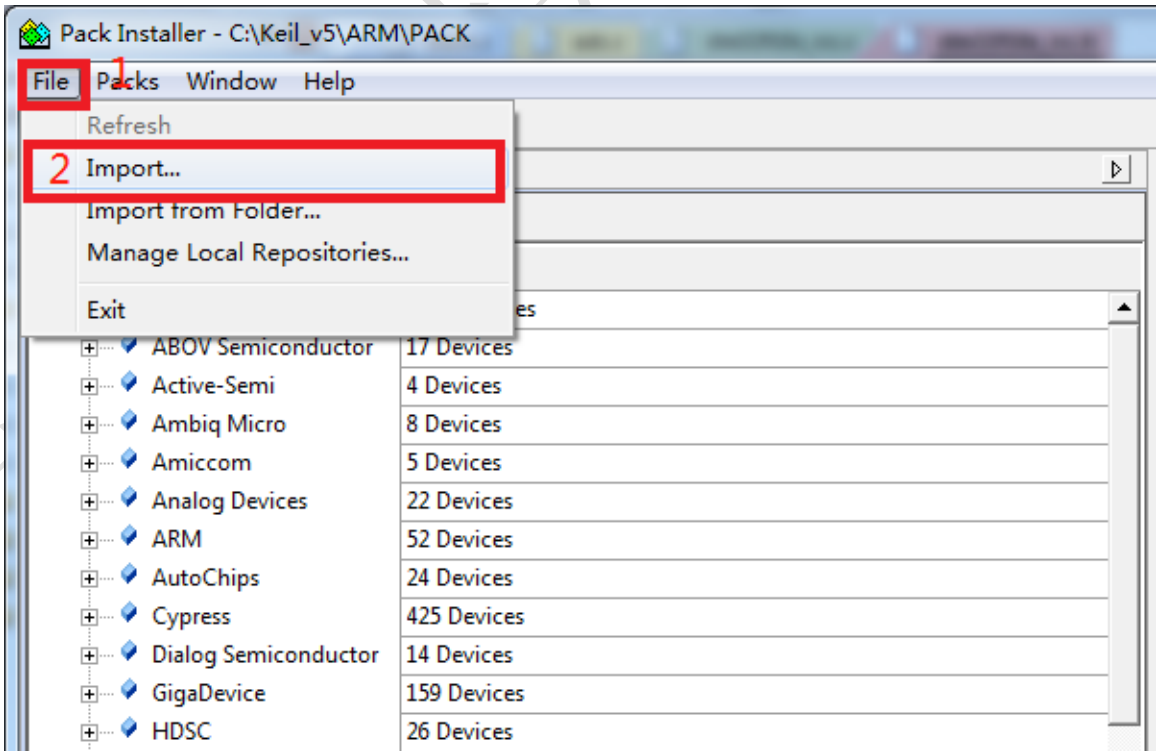

图 2.2Pack Installer

3. **安装完成后，在**“Pack Installer”**窗口的器件列表中会出现**“Chipsea”**项，展开**“CS32F0 Series”，**选择具体型号的** MCU(**如下图** 2.3 **所示**)。

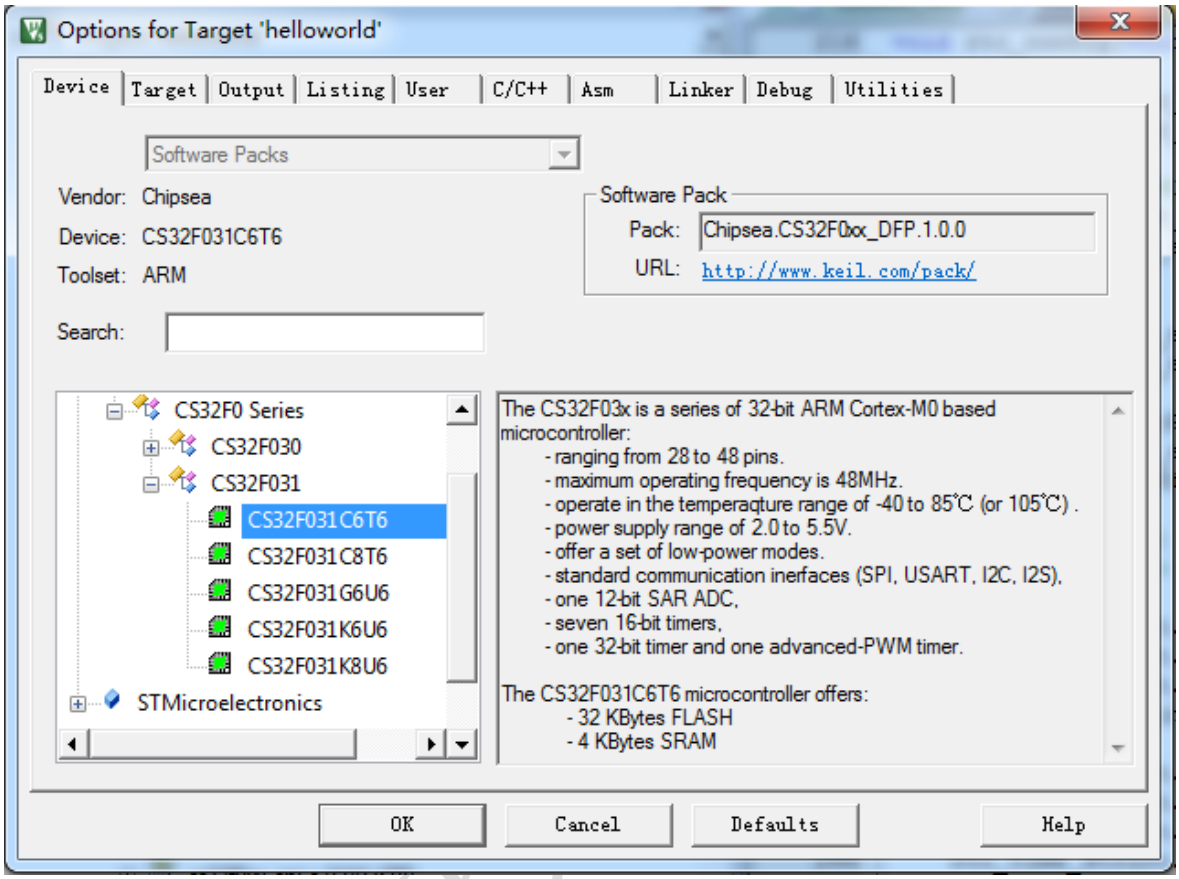

图 2.3 **软件包图**

2.3 IAR 软件获取

IAR 是一家公司的名称，也是一种集成开发环境的名称，我们平时所说的 IAR 主要是指集成开发环境。IAR 下载步骤如下：

- 访问网址：www.iar.com/iar-embedded-workbench/#!?currentTab=free-trials，单击 Download Software，启动下载任务。

2.4 IAR 软件安装

IAR 安装步骤如下：

1. 以管理员身份运行 EWARM-CD-8322-19423.exe 文件，启动安装过程。
2. 选择 Install IAR Embedded Workbench for Arm，后续按照默认配置，完成安装。
3. 注意：USB 驱动安装时，选择对应仿真器的驱动（例如 J-LINK）。
4. 经过上述的步骤，IAR 软件安装已经完成。

3 EVB-32F03X-START 硬件资源介绍

CS32F03x 是芯海科技（深圳）股份有限公司开发的一款基于 Cortex-M0 MCU。

EVB-32F03X-START 开发板是为了方便用户熟悉这款 MCU，并快速应用到自己的设计中，获得时间和成本优势。

本节分为以下 3 部分进行介绍：

- EVB-32F03X-START 开发板硬件资源说明
- EVB-32F03X-START 开发板原理图说明
- EVB-32F03X-START 开发板注意事项

3.1 EVB-32F03X-START 开发板硬件资源说明

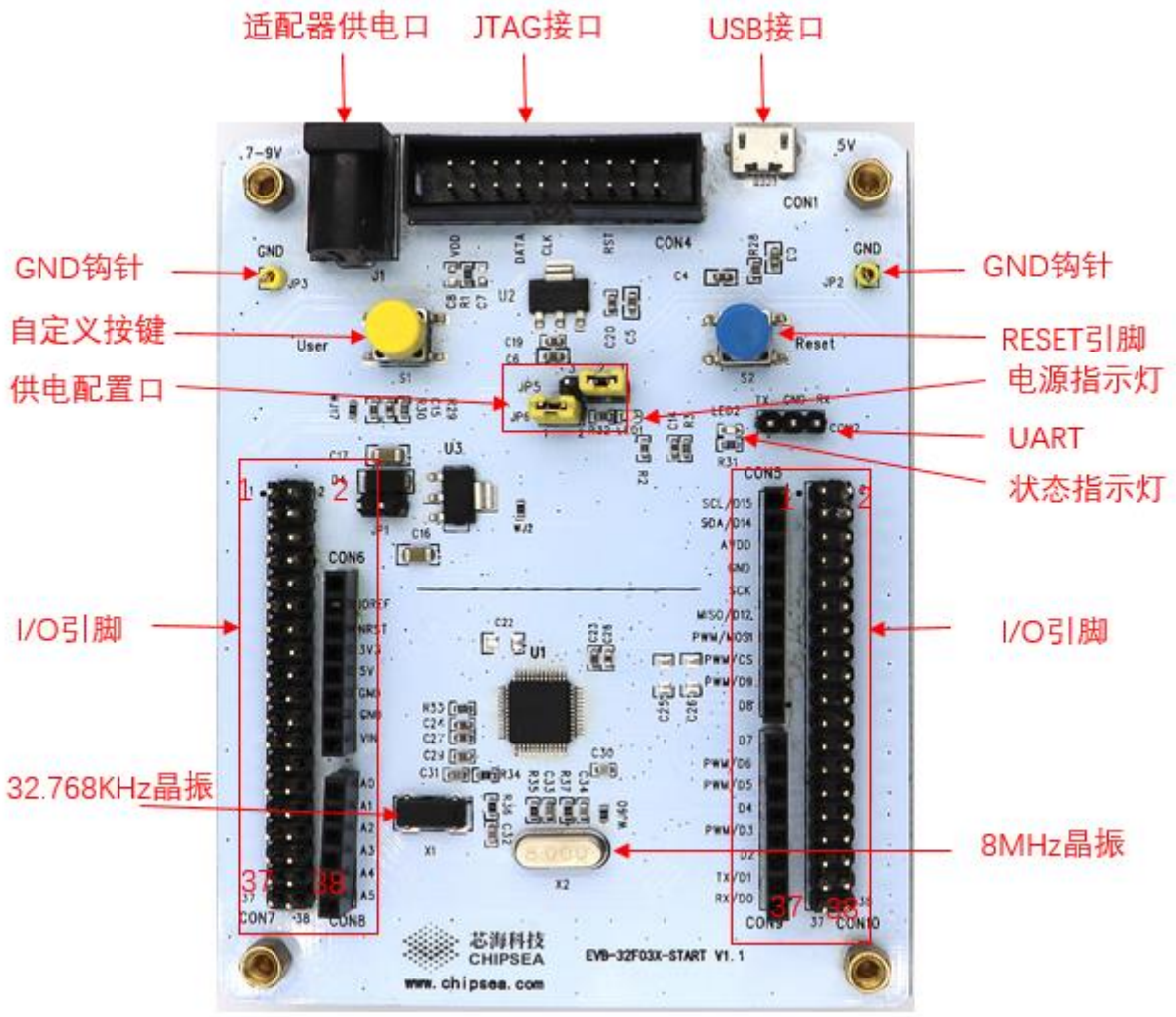

图 3.1 板上资源接口图

3.1.1 EVB-32F03X-START 开发板资源

EVB-32F03X-START 开发板资源如下所示：

- MCU：CS32F03x
- 1 个电源指示灯 LED1（红色）
- 1 个状态指示灯 LED2（默认 PA5 ,可通过跳线设置为 PB13）
- 1 个自定义按键（默认 PC13，可选用）
- 1 个复位按键
- 1 个 UART 接口（默认 PA2，PA3，CS32F031C8x 系列不可用）
- 1 个 USB 供电口
- 1 个电源适配器供电口 (9-15V)
- 3 个供电配置口 JP1、JP5、JP6 （JP1 默认不短接；若 JP5 短接 23，选择适配器供电；若 JP5 短接 12，选择 USB 供电；JP6 短接，为 MCU 提供 3.3V 电源）
- 1 个 JTAG 接口（仅支持 SWD）
- 除晶振占用的 IO 口外，所有 IO 口全部引出。

3.1.2 EVB-32F03X-START 开发板特性

EVB-32F03X-START 开发板特性如下所示：

- 多种供电方式选择，可使用 usb 电源，电源适配器，调试器供电
- 多种供电电压选择， 3.3V 、5V，其它符合 2.0-5.5V 要求的值

3.1.3 硬件资源接口

EVB-32F03X-START 开发板硬件资源接口如下所示：

1. 适配器供电口
   针对功耗大的应用，需要消耗较大的电能，必须使用电源适配器，以满足应用需求。推荐采用输出电压为 9V 的适配器，使用其它规格的适配器，输出电压不超过 15V。
2. JTAG 接口
   板载 20 针标准 JTAG 调试口，该 JTAG 口可以和 ULINK、JLINK 直接连接。这个调试接口仅支持 SWD。
3. USB 供电口
   Micro USB 接口，只使用了电源线和地线。
4. GND 插针
   地电位。
5. RESET 引脚

复位按键（RESET），用于复位 MCU。

6. UART 口

   此 UART 接口，可以配置为 USART1 或 UART2，Pin1 引脚连接 PA2（Tx），Pin3 引脚连接 PA3（Rx）。还可以通过电阻配置将引脚连接到 CON10，用作其它功能。

7. 状态指示灯 LED2

   绿色的状态指示灯，默认连接 PA5 。在调试程序的时候，可以指示程序的状态。

8. 电源指示灯 LED1

   红色的 LED 灯，用于指示电源状态。在电源接通的时候，该灯会亮，否则不亮。通过这个 LED，可以判断开发板的上电情况。

9. I/O 引出脚

   开发板 IO 引出端口： CON5、CON6、CON7、CON8、CON9、CON10。其中，CON7 和 CON10 采用 2*38 的排针， CON6 和 CON9 采用 1*8 的排针，CON8 采用 1*6 的排针，CON5 采用 1*10 的排针。针对具体的应用场合，可以通过电阻配置特定的 I/O 口输出。

10. 自定义按键

    自定义功能按键，连接 MCU 的 PC3 引脚，通过软件设计，可以实现相应的功能。

11. CS32F03x

    开发板的 MCU。以 CS32F031C8T6 为例，该芯片具有 64K FLASH，8K SRAM，39 个快速 I/O 口，5 通道 DMA 控制器，2 路 I2C， 5 路 UART， 2 路 SPI，1 路 12bit ADC，7 个 16bit 定时器，1 个 32bit 定时器，1 个增强控制型 PWM 定时器等资源。

3.2 EVB-32F03X-START 开发板原理图说明

3.2.1 MCU 部分原理图

以 CS32F031C8T6 为例，这款 MCU 拥有 2 路 I2C，2 路 SPI，5 路 UART，供电电压范围为 2.0-5.5V，工作温度范围为 -40~105 度。

MCU 部分原理图如下所示：

- 32.768KHz 外部晶振：芯片有内部晶振，但是精度不高，对精度要求高的场合，需要配置使用外部晶振，才能达到设计要求。
- 8MHz 外部晶振：使用精度要求高的场合。
- 用户按键：按键按下时，PC13 检测到低电平；按键松开时，PC13 检测到高电平。

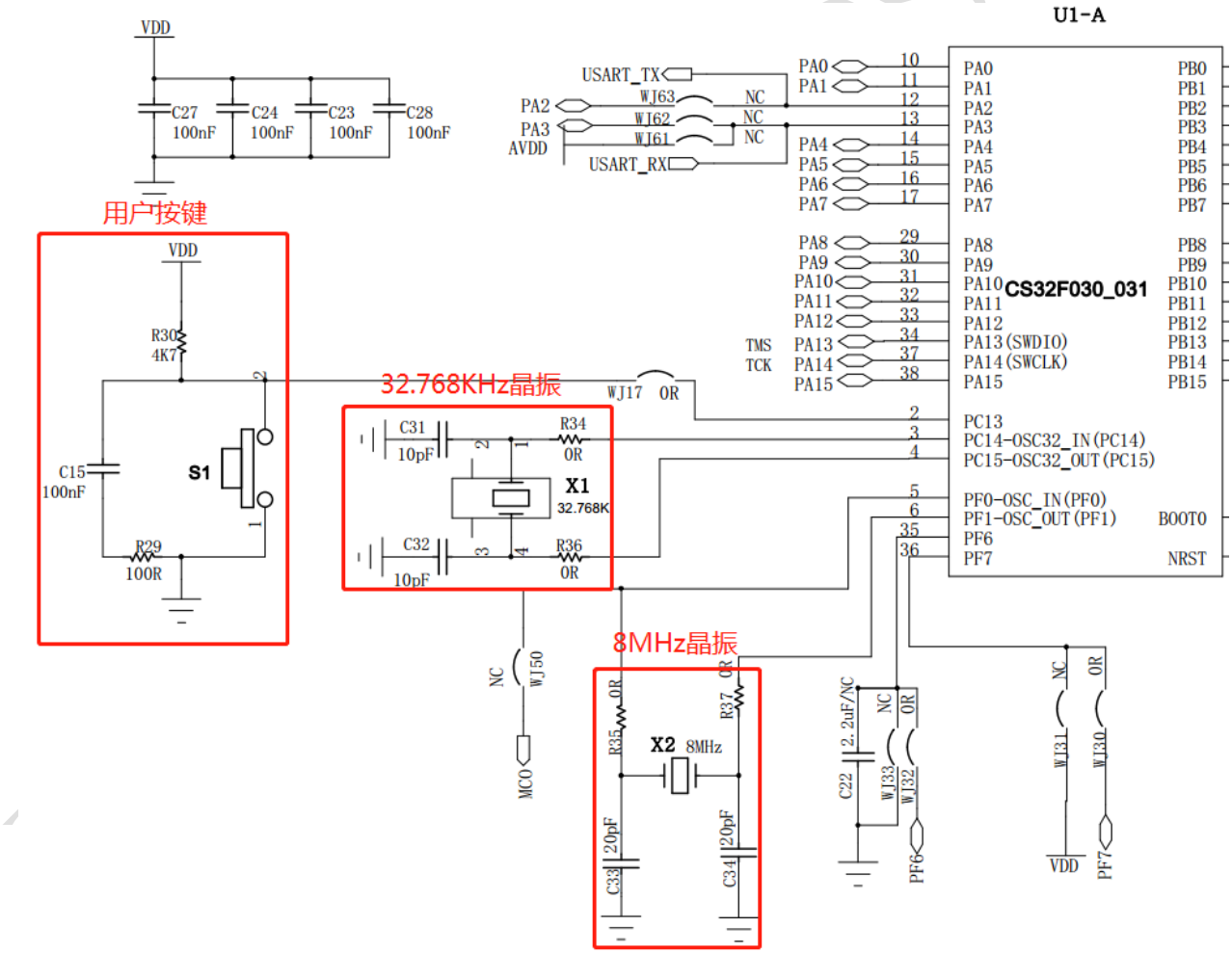

图 3.2 MCU 部分原理图

- **复位按键：芯片是低电平复位。正常状态下，引脚是高电平；按键按下时为低电平，执行复位操作。另外，** R2 、R3、C14 **构成上电复位电路，通过给开发板上电，也能执行复位操作。**

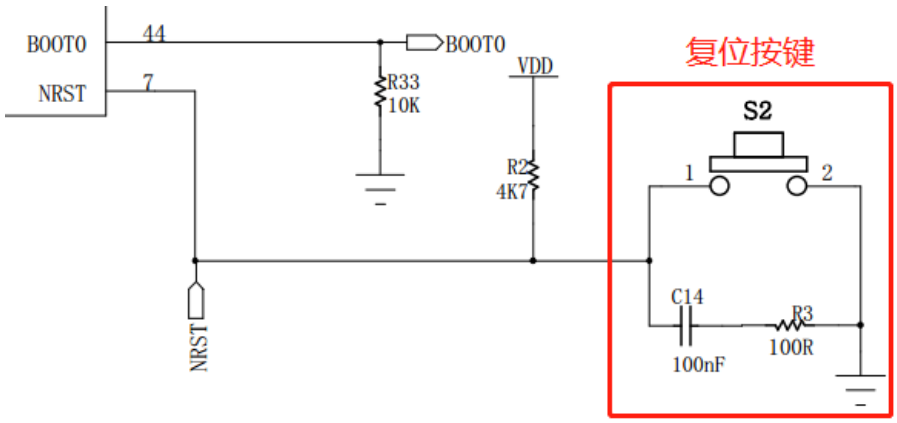

图 3.3 **复位按键原理图**

- **芯片电源部分：芯片电源分为数字部分和模拟部分，在供电设计方面，采用了数模隔离设计，这样设计优势是数模供电互不干扰，另外也方便对数模部分功耗进行单独测量。同时，也分模拟地和数字地，并进行了隔离。这样的设计，可以确保芯片系统更稳定的工作。**

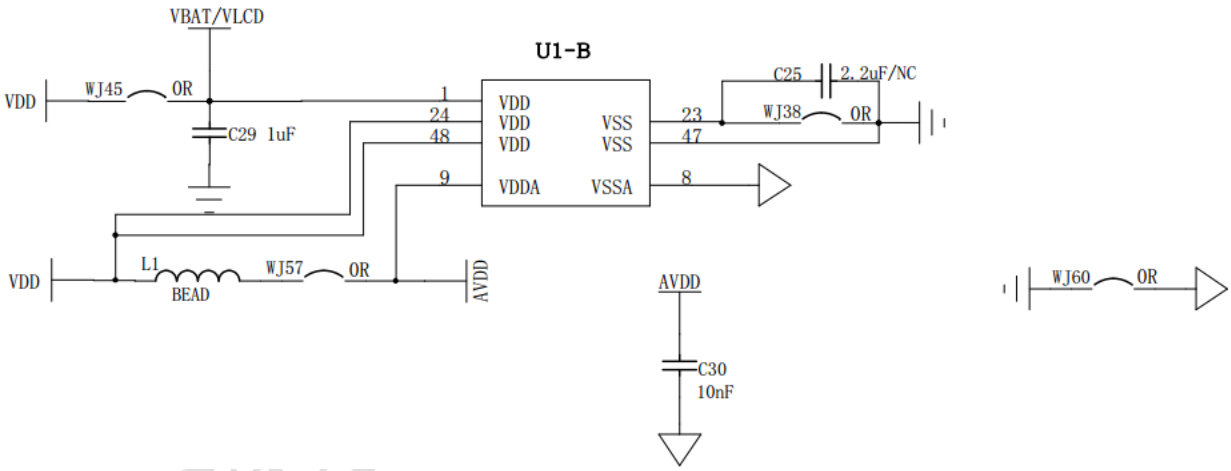

图 3.4 **芯片电源原理图**

3.2.2 I/O 接口部分

MCU 的 I/O 口除了晶振、按键之外，全部引到接口上，使用时注意以下几点：

1. CON7 接口：
   - Pin31，默认与 PF1 连接
   - Pin25，默认与 PC14 断开
   - Pin27，默认与 PC15 断开
   - Pin29，默认与 PF0 断开
   - Pin36，默认与 PB8 断开
   - 如果需要使用这些 I/O，需要使用 0 欧姆电阻进行短接

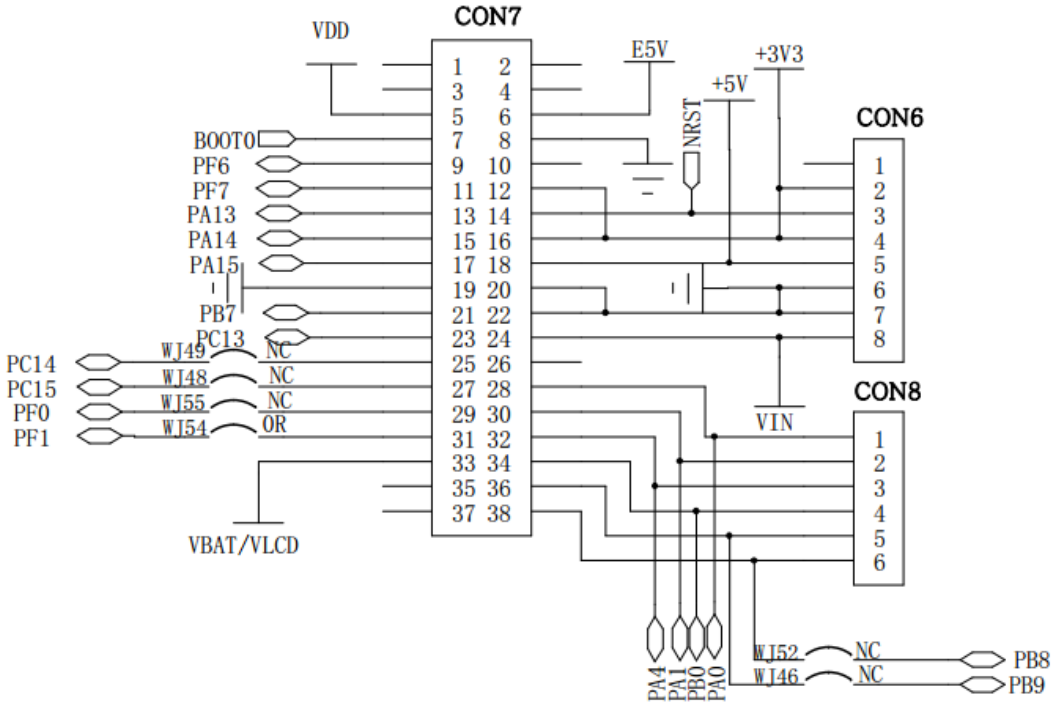

图 3.5 CON7 口原理图

2. CON10 接口：
   - Pin30，PB13、PA5 只能二选一进行使用，默认与 PB13 连接
   - Pin 26，PB15、PA7 只能二选一进行使用，默认与 PB15 连接
   - Pin 28，PB14、PA6 只能二选一进行使用，默认与 PB14 连接
   - Pin 11，PB13、PA5 只能二选一进行使用，默认与 PA5 连接
   - Pin 13，PB14、PA6 根据使用情况配置，默认断开
   - Pin 15，PB15、PA7 根据使用情况配置，默认断开
   - 配置具有关联性，请仔细核对原理图进行操作

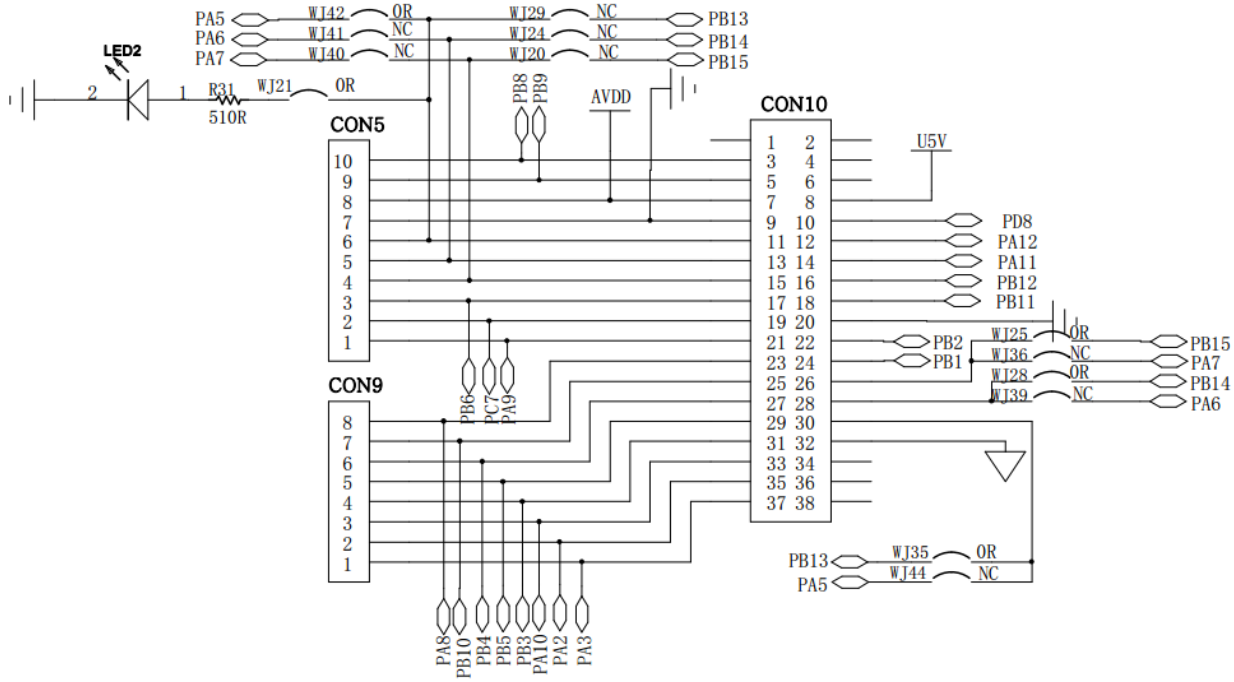

图 3.6 CON10 **口原理图**

3.2.3 串口接口部分

MCU 串口功能引脚，通过引线连接到 CON2，需要注意的是 CON2 的 pin2 是 GND。UART 可以是 USART1 或 UART2 。USART_TX 连接 PA2，USART_RX 连接 PA3 。

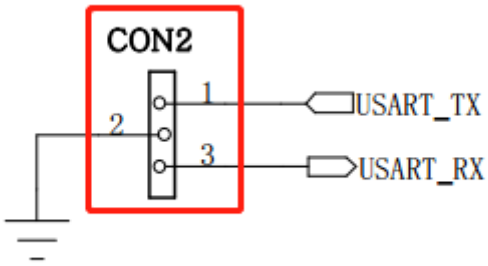

图 3.7 串口原理图

3.2.4 JTAG/SWD 接口部分

接口使用的是标准的 JTAG 口，方便调试器的连接。CS32F03x MCU 仅支持 SWD 连接方式。

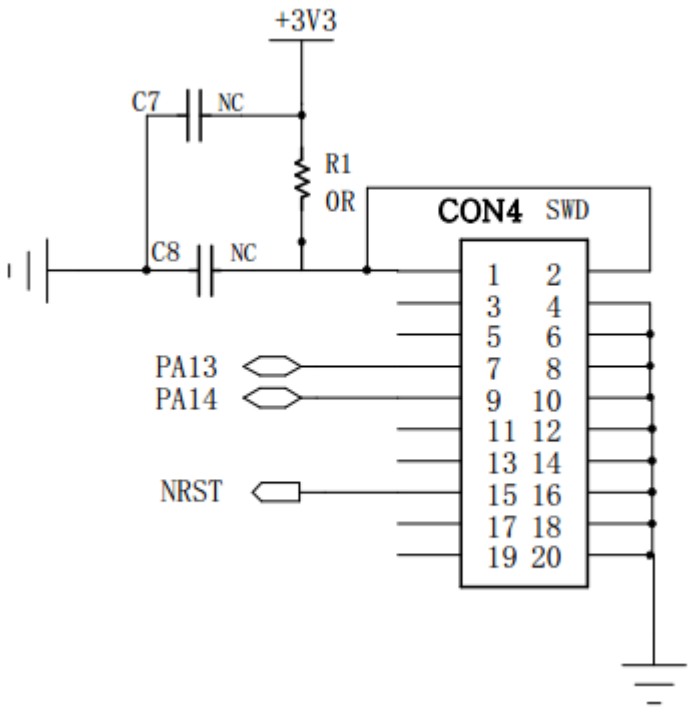

图 3.8 下载接口原理图

3.2.5 电源接口部分

电源接口部分配置方式：

- **供电方式上，有** USB **电源和电源适配器两种，通过** JP5 **配置**
- **供电电压上，**MCU **的供电电压选择上也有两种，分别是** 3.3V **和** 5V **，通过** JP5 **和** JP6 **配置**

JP5 JP6 **接法举例：**

- **如果选择** 3.3V **芯片供电电压，短接** JP6 **的两个** Pin **脚**
  - **使用** USB **电源时，短接** JP5 Pin1 **与** Pin2
  - **使用适配器供电时，短接** JP5 Pin2 **与** Pin3
- **如果选择** 5V **芯片供电电压，**JP6 **的右** Pin **脚与** JP5 **的** Pin **脚短接，具体如下**
  - **使用** USB **电源时，短接** JP6 **的右** Pin **脚和** JP5 **的** Pin1
  - **使用适配器供电时，短接** JP6 **的右** Pin **脚和** JP5 **的** Pin3

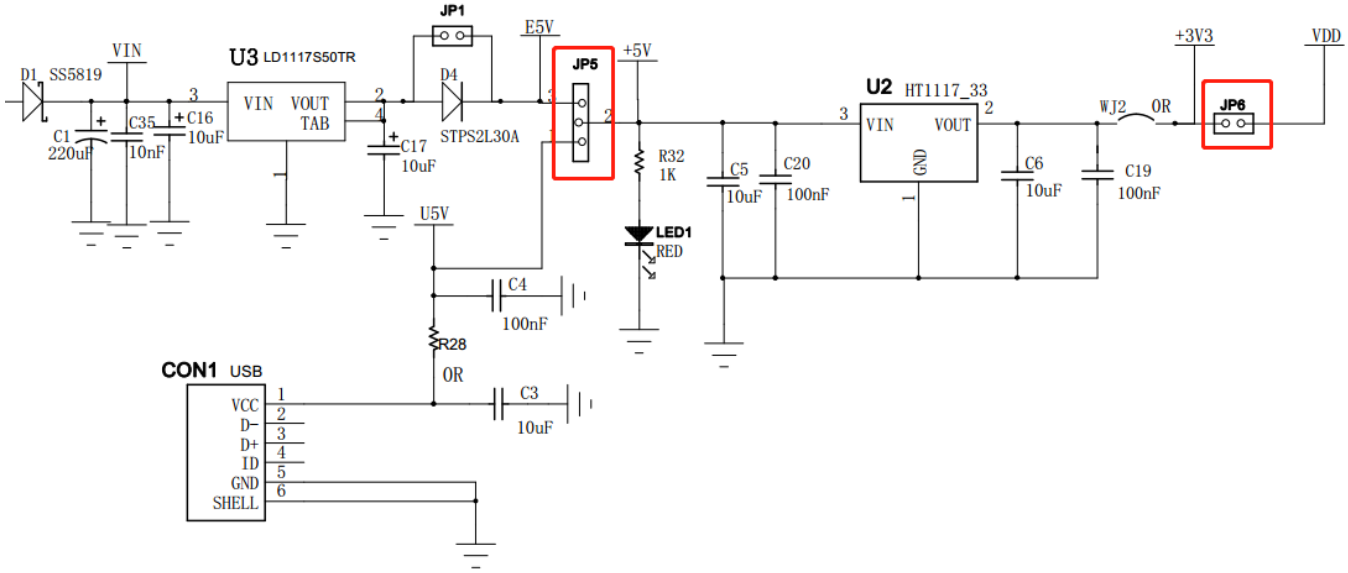

图 3.9 **电源接口部分原理图**

3.2.6 接口 Pin 脚速查表

开发板 I/O 引出口，对应引脚默认定义，请参考如下表格，如需配置 IC 的 Pin，请参考原理图。

CON7					CON10			
编号	**Pin 脚**	**编号**	**Pin 脚**		**编号**	**Pin 脚**	**编号**	**Pin 脚**
1	NC	2	NC		1	NC	2	NC
3	NC	4	NC		3	PB8	4	NC
5	VDD	6	E5V		5	PB9	6	NC
7	BOOT0	8	GND		7	AVDD	8	U5V
9	PF6	10	NC		9	GND	10	PD8
11	PF7	12	+3V3		11	PA5	12	PA12
13	PA13	14	NRST		13	NC	14	PA11
15	PA14	16	+3V3		15	NC	16	PB12
17	PA15	18	+5V		17	PB6	18	PB11
19	GND	20	GND		19	PC7	20	GND
21	PB7	22	GND		21	PA9	22	PB2
23	PC13	24	VIN		23	PA8	24	NC
25	NC	26	NC		25	PB10	26	PB15
27	NC	28	PA0		27	PB4	28	PB14
29	NC	30	PA1		29	PB5	30	PB13
31	PF1	32	PA4		31	PB3	32	GND
33	VBAT/VLCD	34	PB0		33	PA10	34	NC
35	NC	36	NC		35	NC	36	NC
37	NC	38	NC		37	NC	38	NC

CON5			CON6			CON8			CON9	
编号	**Pin 脚**		**编号**	**Pin 脚**		**编号**	**Pin 脚**		**编号**	**Pin 脚**
1	PA9		1	NC		1	PA0		1	PA3
2	PC7		2	+3V3		2	PA1		2	PA2
3	PB6		3	NRST		3	PA4		3	PA10
4	NC		4	+3V3		4	PB0		4	PB3
5	NC		5	+5V		5	NC		5	PB5
6	PA5		6	GND		6	NC		6	PB4
7	GND		7	GND					7	PB10
8	AVDD		8	VIN					8	PA8
9	PB9									
10	PB8									

表 3.1 Pin 脚速查表

3.2.7 开发板完整原理图

开发板原理图下载地址：访问

http://218.17.109.235/FolderPublish.aspx?code=B5f5a0f39fefb4af78785189452d2effb。

3.3 EVB-32F03X-START 开发板使用注意事项

在使用 EVB-32F03X-START 开发板过程中需要注意以下问题：

- USB 最大的供电电流为 500mA，如果存在大的负载，需要选用适配器供电。
- MCU 的调试接口仅支持 SWD，所以，在仿真器使用过程中出现 JTAG 不能使用，请及时切换到 SWD 模式。
- 当使用某个 IO 口的时候，请查看开发板的对应接口原理图，确保 I/O 口短接电阻已正确配置。
- 对于板上使用跳线帽的地方，要仔细确认连接的正确性。
- 不建议使用仿真器对开发板供电。
- 开发板连接扩展板时，使用之前，需要仔细确认原理图的 I/O 已正确配置，再去进行调试。

4 CS32F0xx Keil Software Pack 描述

双击 Chipsea.CS32F0xx_DFP.1.0.0.pack 安装以后，在路径 C:\Keil_v5\ARM\PACK\Chipsea 下可以看到下列子文件夹，如图 4.1 所示。

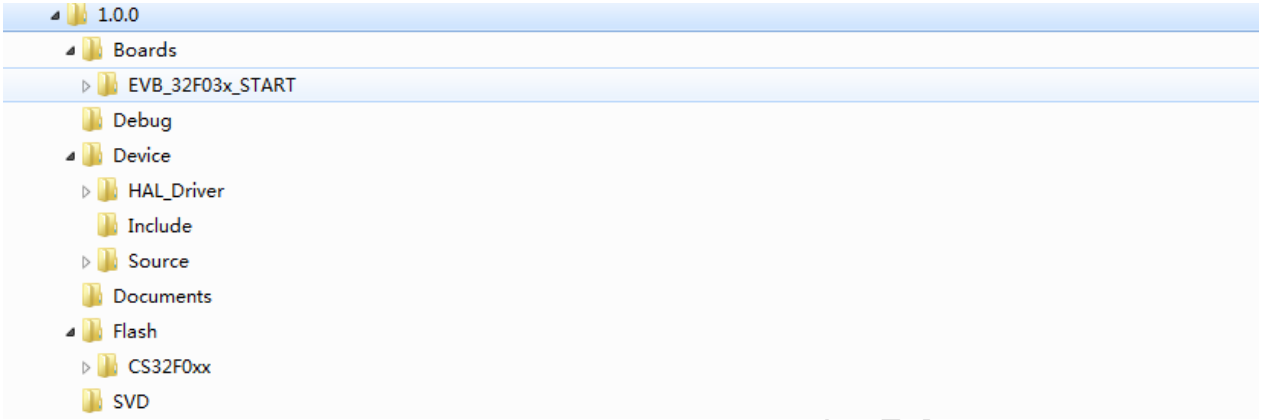

图 4.1 软件环境

4.1 Boards 文件夹

此文件夹包含 EVB-32F03X-START 开发板支持的相关文件。

- Common 子文件夹
  此文件夹包含 Buttons.c、LED.c 文件。
- Documents 子文件夹
  此文件夹包含 EVB-32F03X-START 开发板。
- HAL_Examples 子文件夹
  此文件夹下包含 EVB-32F03X-START 外设库使用示例。
- RTX_Blinky 子文件夹
  此文件夹包含“RTX_Blinky”工程，该工程是一个基于 EVB-32F03X-START 开发板的符合 CMSIS 标准的跑马灯程序。
- Utilities 子文件夹
  此文件夹下包含 EVB-32F03X-START 开发板的公共示例。主要包含开发板按键、IIC EEPROM 以及 SPI FLASH 公用示例。

4.2 Device 文件夹

此文件夹包含 CS32F03x 开发所必须的基本代码以及外设驱动程序文件。

- HAL_Driver 子文件夹
  a) inc 文件夹
     此文件夹下包含外设的 inc 头文件，例如 cs32f0xx_adc.h、cs32f0xx_crc.h 等。
  b) src 文件夹

此文件包含了实现各个模块不同功能的源文件，例如 cs32f0xx_adc.c、cs32f0xx_crc.c **等**。

c) templates **文件夹**

用户使用的工程模板 DEMO。**用户可以快速在该工程上面替换** HAL_Examples **文件夹中的外设应用程序，加快工程开发周期**。

- Include **子文件夹**

  此文件夹下包含了 cs32f0xx.h、cs32f0xx_conf.h、system_cs32f0xx.h **头文件**。

- Source **子文件夹**

  此文件夹下包含了：

  a) ARM **文件夹**

  此文件夹下包含操作代码选项区的启动文件 CS32F0xx_OTP.s、cs32f030 **启动文件** startup_cs32f030.s、cs32f031 **启动文件** startup_cs32f031.s.

  b) system_cs32f0xx.c **源文件**

  此文件主要是 CMSIS Cortex-M0 **设备外设访问层系统源文件，主要用来配置系统时钟**。

4.3 Document 文件夹

此文件夹下包含用户开发所使用的文档，主要包含如下文档

- CS32F03X 用户手册
- CS32F03X 数据手册
- 应用笔记
- CS32F03X FAQ
- CS32F03X SDK API 参考手册
- EVB-32F03X-START 用户手册
- EVB-32F03X-START 开发板示例使用说明

4.4 Flash 文件夹

此文件夹下包含了 DFP 预定义的 Flash Program 算法，用来支持 MCU 的编程。其中包含 16k、32k、64k Flash 烧录算法以及 opt 烧录算法。

4.5 SVD 文件夹

此文件夹是包含完整微控制器系统（包括外设）的程序员视图的系统视图描述 XML 文件。

5 使用 MDK 软件新建工程

5.1 新建“hello world”工程

5.1.1 EVB-32F03X-START 开发板硬件连接

将 PC 串口的 RX 和 TX 引脚分别与开发板相对应的 D8(PA9)和 D2(PA10)引脚相连接。

5.1.2 软件配置

1. 在桌面新建 Helloworld 文件夹，在此目录下面建立 User 子文件夹。
2. 在 User 子文件夹下面建立空文件 main.c，并把后文附录 1 的代码复制到 main.c 中。
3. 单击 MDK 的菜单： Project–>New Uvision Project 。然后将目录定位到刚才建立的文件夹 Template 之下，将工程文件保存到 Template 文件夹中。工程命名为 helloworld，点击保存。
4. 接下来会出现一个选择 MCU 的界面，根据 EVB-32F03X-START 所使用的 MCU 型号，以 CS32F031C8T6 为例：依次选择 Chipsea→CS32F0 Serise→CS32F031→CS32F031C8T6 。点击 OK，MDK 会弹出 Manage Run-Time Environment 对话框， 在这个界面，添加自己需要的组件，从而构建开发环境。
5. 依次勾选 CORE、Startup 以及所有的外设文件，然后单击 OK 按钮。如图 5.1。

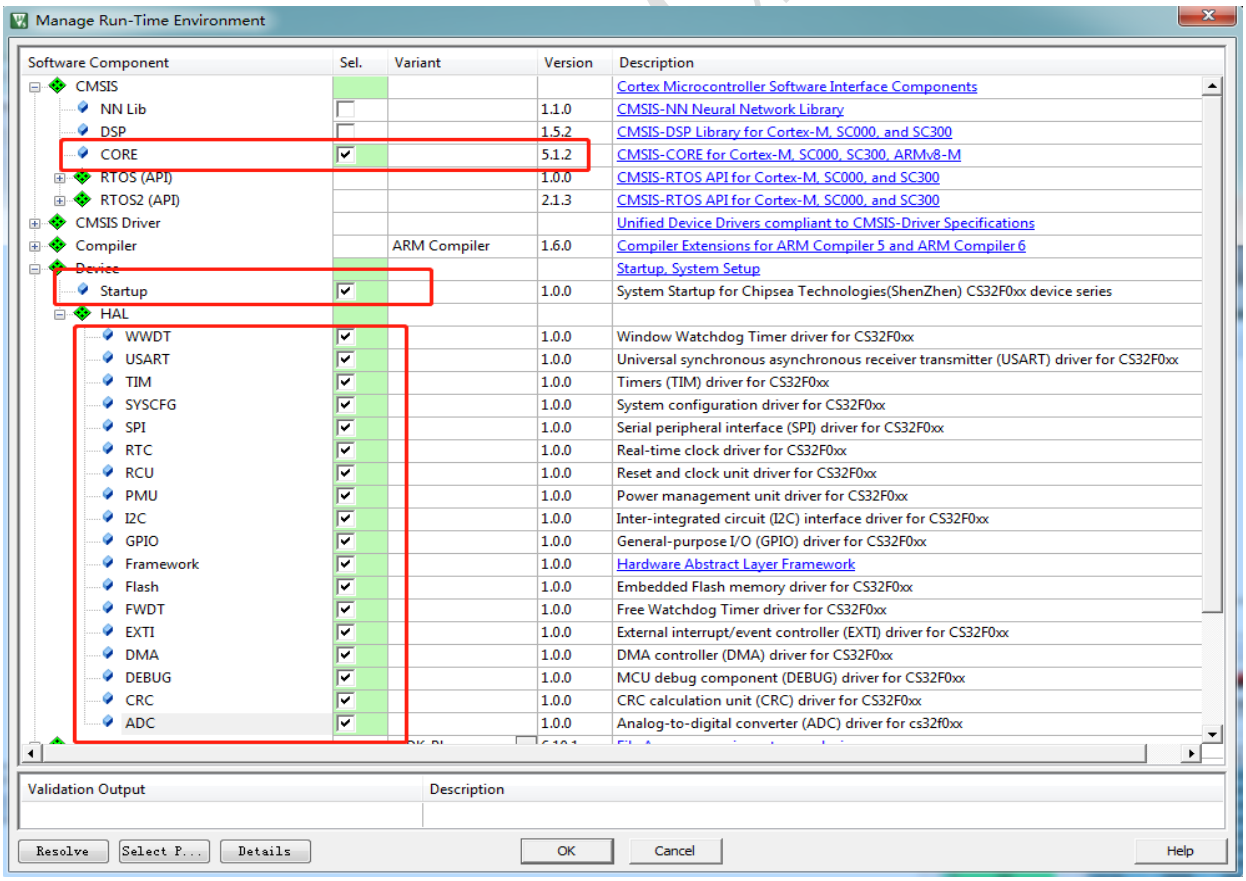

图 5.1 开发环境搭建

6. 单击分组 Manage Project Items 按钮。双击 Target 修改名字为 helloworld，双击 SourceGroup1

修改名字为 Uesr。并为 User 添加 main.c 文件，然后点击 OK。可以看到 Target 名字以及 Groups 情况，如图 5.2 。

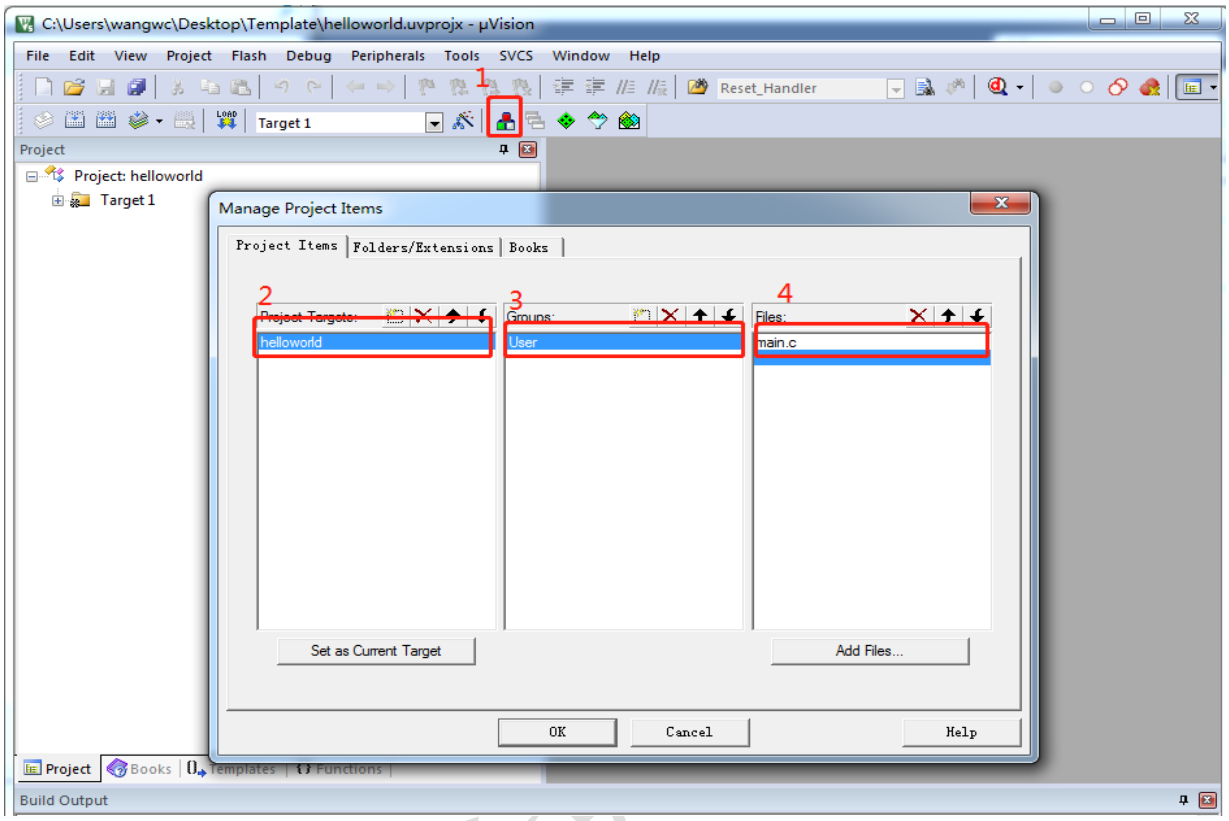

图 5.2 添加文件到 User 分组

7. 修改 Options for Target
   - 修改 c/c++选项卡
     - 修改 Include Paths，将 User 目录加入编译器头文件搜索路径（单击右边的按钮，弹出一个添加 path 的对话框，然后将 User 目录添加进去，如图 5.3）
     - 增加预定义宏 CS32F031（具体操作是在 Define 选项框中粘贴 CS32F031 如图 5.4，具体情况请根据相应的系列定义相应的宏）
   - 修改 Target 选项卡，勾选 Use MicroLib
   - 修改 Output 选项卡，勾选 Create HEX File，如图 5.5
   - 修改 Debug 选项卡
     - 配置仿真器，以 J-LINK 为例，在 Debug 选项卡选择 Use 复选框，在下拉列表中选择仿真器 J-LINK/J-TRACE Cortex，并勾选 Run to main 单选框，如图 5.6
     - 设置 JLINK 的参数，单击 Settings 按钮，弹出 Cortex JLink/JTrace Target Driver Setup 在 Debug 选项卡下选择 SW 模式，在 Flash Download 下面勾选 Reset and Run，如图 5.7，依次单击确定与 OK 关闭配置框。

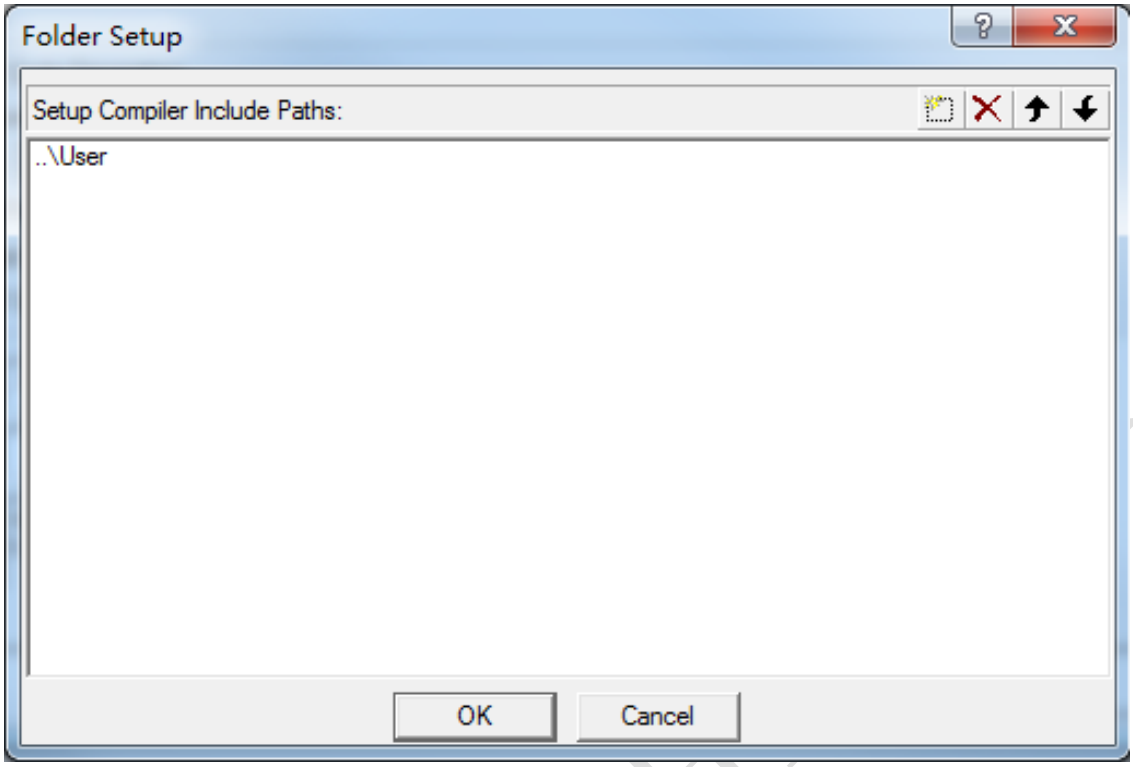

图 5.3 添加头文件路径

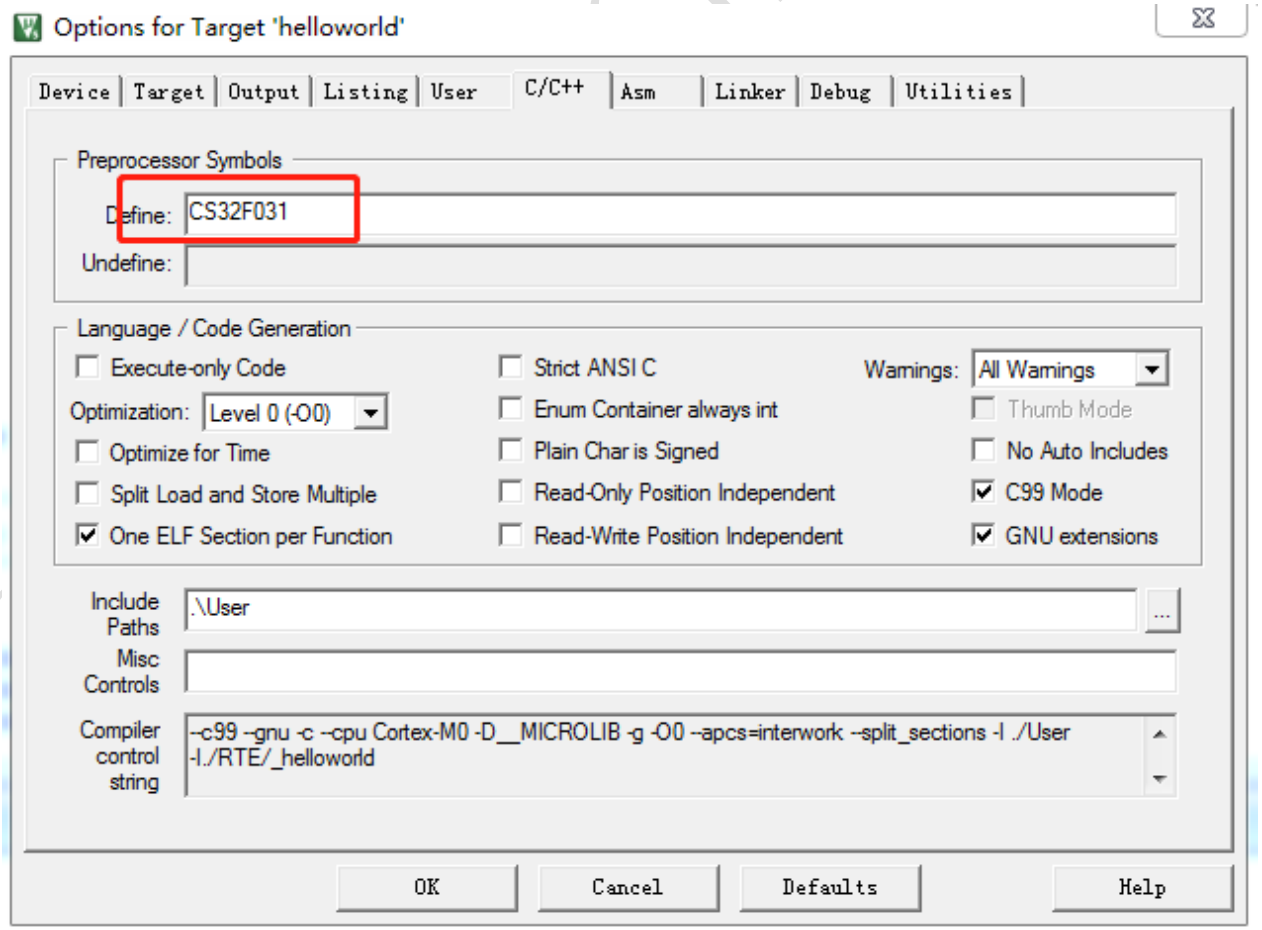

图 5.4 定义宏

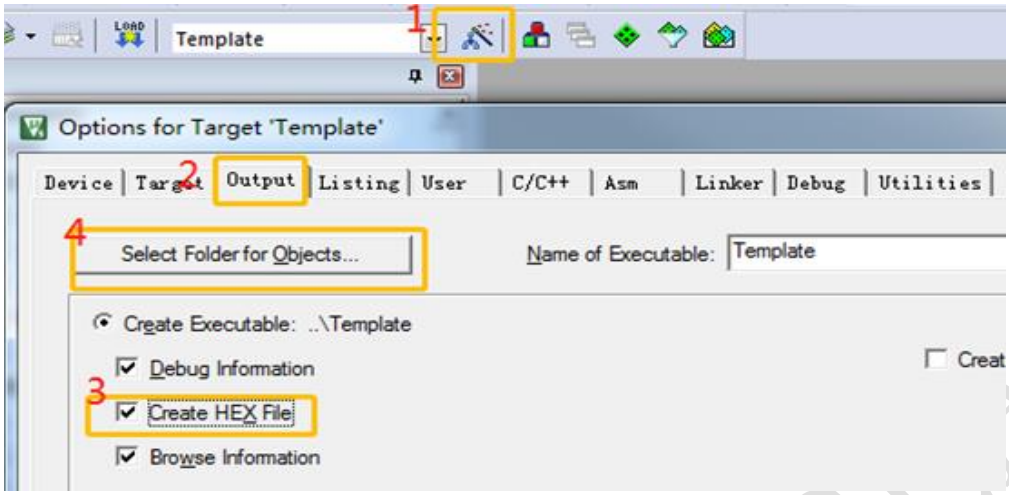

图 5.5 配置选项卡

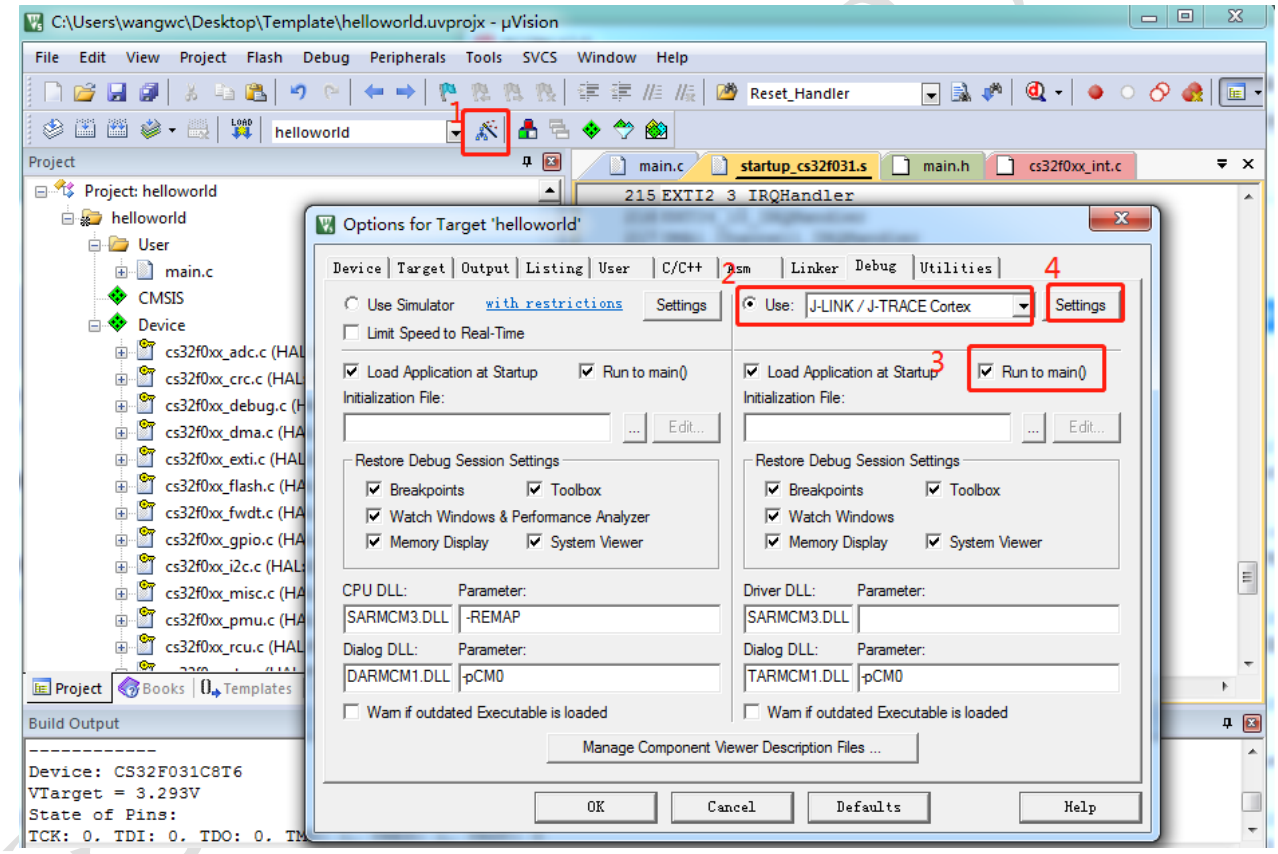

图 5.6 Debugger 选项卡设置

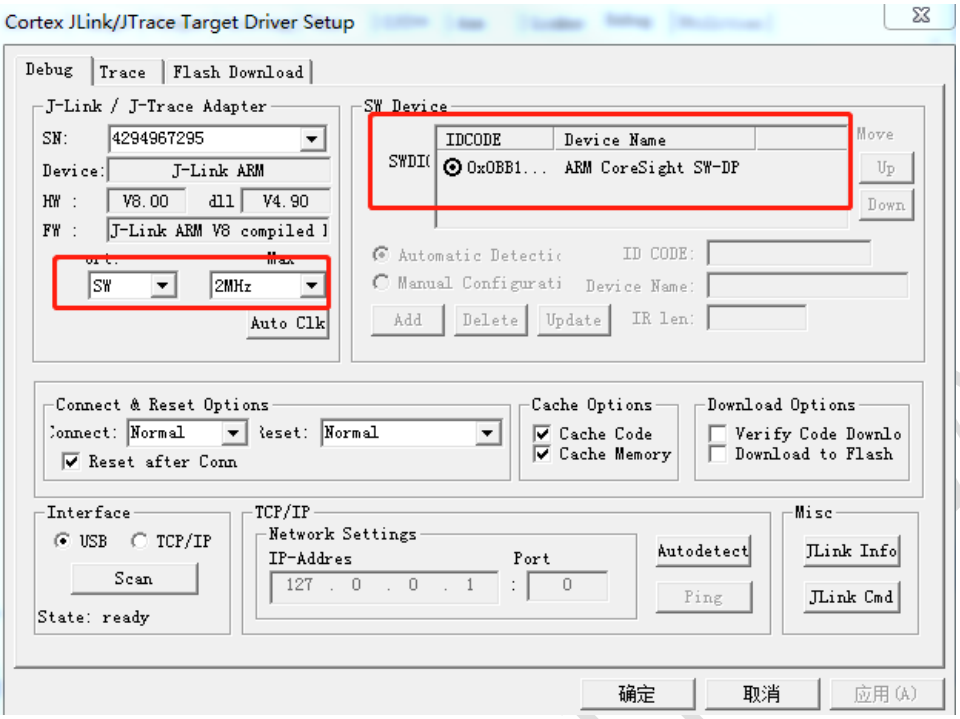

图 5.7 JLINK 模式设置

8. **单击** Rebulid **按钮，编译工程，编译成功之后，单击** Download **按钮，将该程序下载到** EVB-32F03X-START **开发板**。
9. **打开串口工具，将波特率设置为** 115200，**按下开发板的复位按钮** S2，**可以看到在串口工具中输出** “hello world”，**如图** 5.8 **所示**。

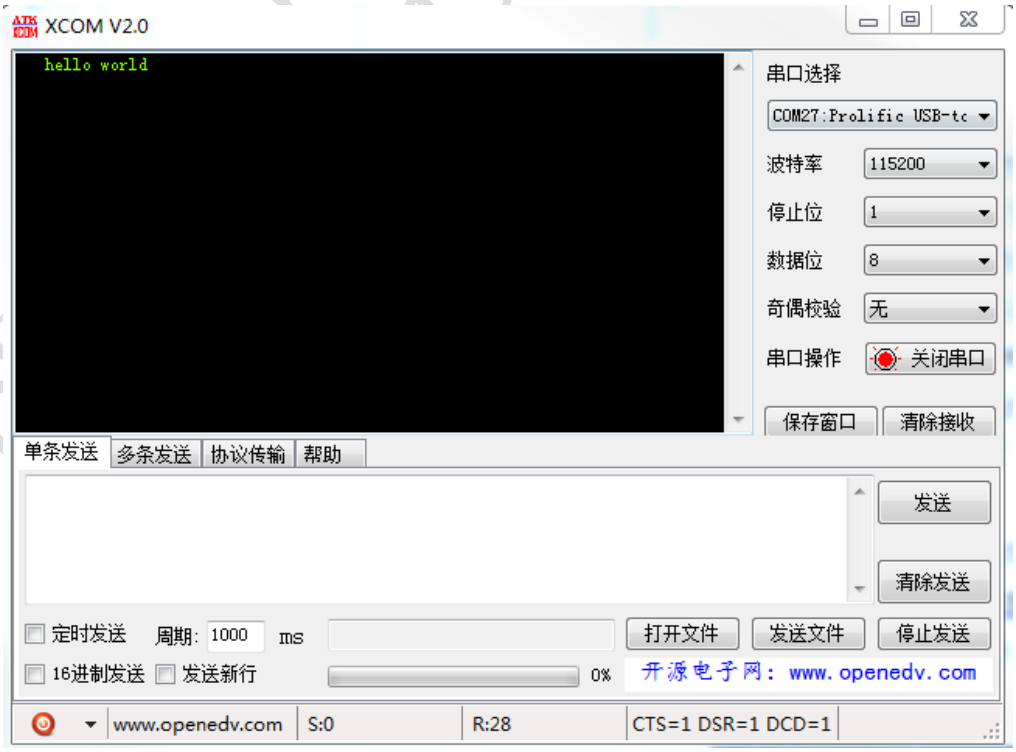

图 5.8 **输出** hello world

5.2 编译现有的 MDK 项目

编译 “helloworld**” 工程步骤如下**:

1. **进入** helloworld **工程目录**。
2. **双击** helloworld.uvprojx。
3. **在** Project **菜单中，选择** Rebuild all target files。
4. **如果项目编译成功，屏幕上将会显示** Build Output **窗口，如图** 5.9。

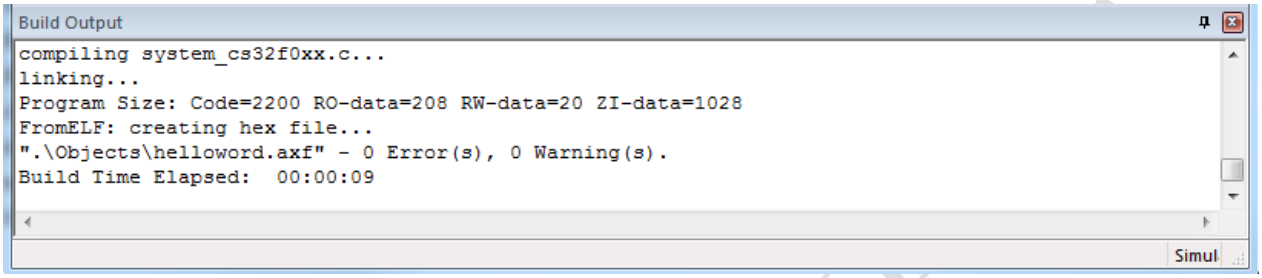

图 5.9 **项目编译**

5.3 在线仿真调试 CS32F03x 程序

以 JLINK **为例，在线仿真调试有以下方式：**

1. **设置断点，**Ctrl+F5 **启动在线仿真，对** CS32F03x MCU **进行调试，如在配置项中已勾选** Run to main()**选项，程序则直接运行到** main **函数入口，在主函数** 54 **行号前单击左键设置断点，如图** 5.10 **所示**。

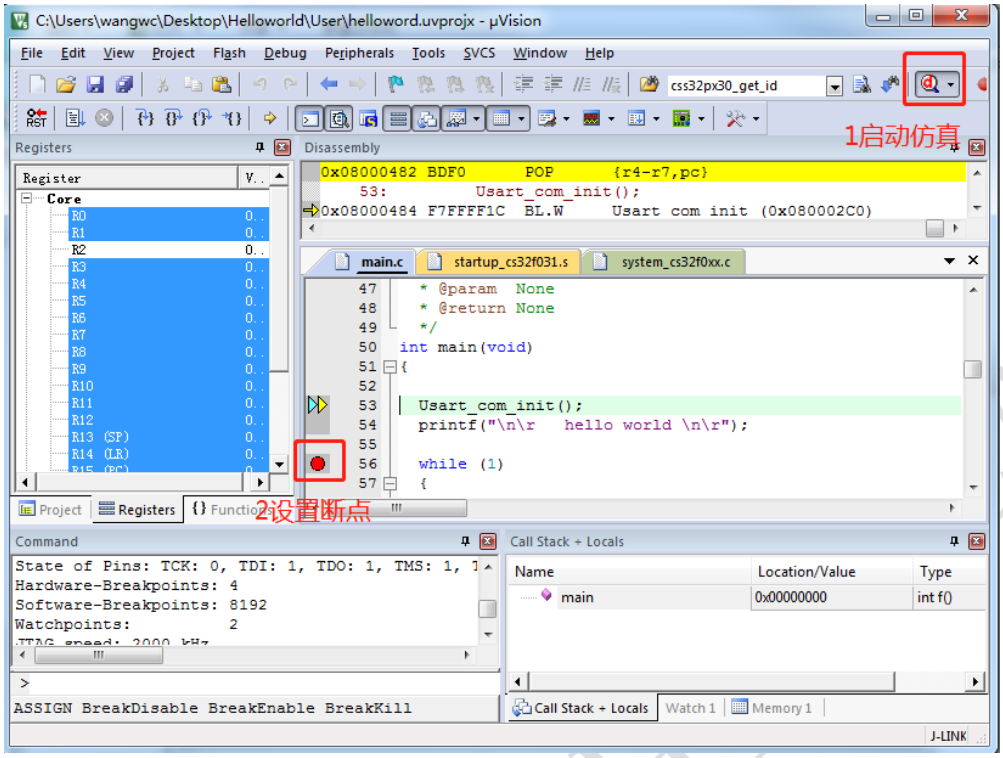

图 5.10 开始仿真

2. **全速运行，单击** Run **按钮，程序将会快速执行到断点处。如图** 11 **所示，实现在硬件上面的真实仿真。**
3. Call Stack +Locals **窗口：用于显示** main **函数内的变量及所调用的子函数地址值。**
4. Watch **窗口：用来查看特殊变量的窗口。用户可自行输入某个变量名，根据程序运行的进度查看变量值。**
5. Memory **窗口：通过在** Address **后的编辑框内输入地址的值即可显示相应内存的值。**
6. Register **窗口：用于显示各寄存器的值，它们用于保存和处理内核里暂时使用的数据。**
7. **单击**"View->SystemVier"，**可以查看相应外设所包含所有寄存器的值。**

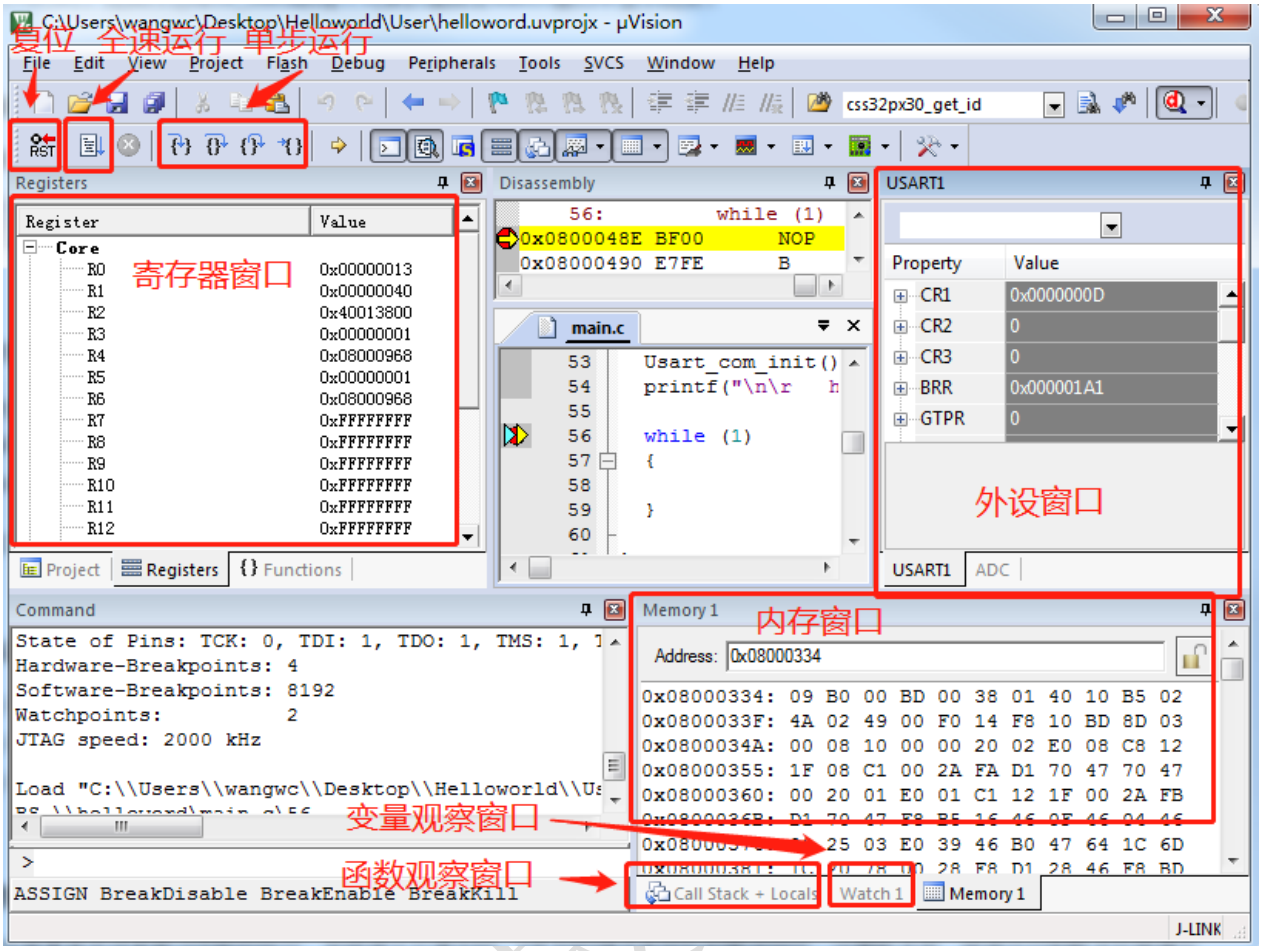

图 5.11 仿真运行断点处

附录 1 hello world

```
/**
  * @file    USART/USART_Printf/main.c
  * @brief   Main program body
  * @author  ChipSea MCU Group
  * @version V1.0.0
  * @date    2018.11.01
  * @copyright CHIPSEA TECHNOLOGIES (SHENZHEN) CORP.
  * @note
  * <h2><center>&copy; COPYRIGHT 2018 ChipSea</center></h2>
  *
  */
#include <stdio.h>
#include <string.h>
#include "cs32f0xx.h"
#include "cs32f0xx_conf.h"

/** @addtogroup CS32F0xx_HAL_Examples
  * @{
  */

/** @addtogroup USART_Printf
  * @{
  */
static void usart_com_init(void);

/**
  * @fn int main(void)
  * @brief  Main program.   MCU clock setting is  configured  through SystemInit()
  *         in startup file (startup_cs32f0xx.s) before to enter to application main.
  * @param  None
  * @return None
  */
int main(void)
{
  usart_com_init ();
  printf("\n\r   hello world \n\r");

    while (1)
    {

    }
}

/**
 * @fn void usart_com_init(void)
 * @brief  Initializes USART.
 * @param  None
 * @return None
 */
static void usart_com_init(void)
{
    usart_config_t usart_configStruct;
    gpio_config_t  gpio_configStruct;

   // Clock Config
```

```
    rcu_ahb_periph_clock_enable_ctrl(RCU_AHB_PERI_PORTA, ENABLE);
    rcu_apb2_periph_clock_enable_ctrl(RCU_APB2_PERI_USART1, ENABLE);

    // GPIO MF Config
    gpio_mf_config(GPIOA, GPIO_PIN_NUM9, GPIO_MF_SEL1);
    gpio_mf_config(GPIOA, GPIO_PIN_NUM10, GPIO_MF_SEL1);

    gpio_configStruct.gpio_pin = GPIO_PIN_9 | GPIO_PIN_10; //PA9,PA10 USART
    gpio_configStruct.gpio_mode = GPIO_MODE_MF;
    gpio_configStruct.gpio_speed = GPIO_SPEED_MEDIUM;
    gpio_configStruct.gpio_out_type = GPIO_OUTPUT_PP;
    gpio_configStruct.gpio_pull = GPIO_PULL_NO_PULL;
    gpio_init(GPIOA, &gpio_configStruct);

    // USART Config
    usart_def_init(USART1);
    usart_configStruct.usart_rate = 115200;
    usart_configStruct.data_width = USART_DATA_WIDTH_8;
    usart_configStruct.stop_bits = USART_STOP_BIT_1;
    usart_configStruct.usart_parity = USART_PARITY_NO;
    usart_configStruct.flow_control = USART_FLOW_CONTROL_NONE;
    usart_configStruct.usart_mode = USART_MODE_RX | USART_MODE_TX;
    usart_init(USART1, &usart_configStruct);
    usart_enable_ctrl(USART1,ENABLE);
}


/**
  * @brief  Retargets the C library printf function to the USART.
  * @param  None
  * @return None
  */
int fputc(int ch, FILE *f)
{
    usart_data_send(USART1, (uint8_t) ch);
    while(RESET == usart_flag_status_get(USART1, USART_FLAG_TCF));
    return ch;
}

/**
  * @}
  */
```