版本	V1.1
日期	202211

Quick Start

APT32F171x 系列

Revision History

版本	日期	描述	作者
V1.0	2022-4-15	新建使用说明	lq
V1.1	2022-11-16	1、修改”例程获取方式”介绍 2、完善“串口配置”说明 3、更新部分示意图	wangch

目录

1. 文档说明 4

2. 资源列表 4

3. 硬件环境 4

3.1 调试模块（蓝色部分） 5

3.2 芯片（红色部分） 5

4. 软件环境 6

4.1 CDK 开发环境 6

4.2 代码结构 6

5. 例程运行 7

5.1 例程获取方式 7

5.2 导入 flash 算法 13

5.3 适配代码 13

5.4 GPIO 可视化配置 17

5.5 外设使用 22

5.6 编译工程 22

5.7 下载调试 23

1. 文档说明

该文档基于爱普特CSI驱动代码架构，适用于APT32F171x系列芯片

2. 资源列表

为了使用APT32F171x系列芯片，您将可能需要下列资源

- **系列开发板**。详见 硬件环境 说明。
- **miniUSB线**。通常随开发板发出。
- **CDK**。详见 软件环境 说明。
- **相关文档**
    - 本文档。
    - 系列使用手册。面向应用开发人员，旨在给予171x系列内核、存储及全部外围的完整信息。
    - 系列内产品的数据手册。包含芯片管脚分布、封装信息、电器参数等型号专属信息。
    - APT32F171x系列CSI_API说明手册.pdf。
    - APTCHIP_FAQ.chm。集合了一些使用爱普特芯片时的常见问题和解决方法。
- **驱动和demo工程**。详见例程运行。 内含
    - 芯片CSI驱动库及模块示例代码。

3. 硬件环境

您收到的开发板大致如下图所示，分左右两部分。蓝色部分是调试模块；红色部分是 APT32F171x 系列模块。左右部分通过 SWD 接口和供电相互连接，所以在某些时候这两部分可以分开使用。

开发板实际布局不同时期可能会有一点差别，下图仅为参考。

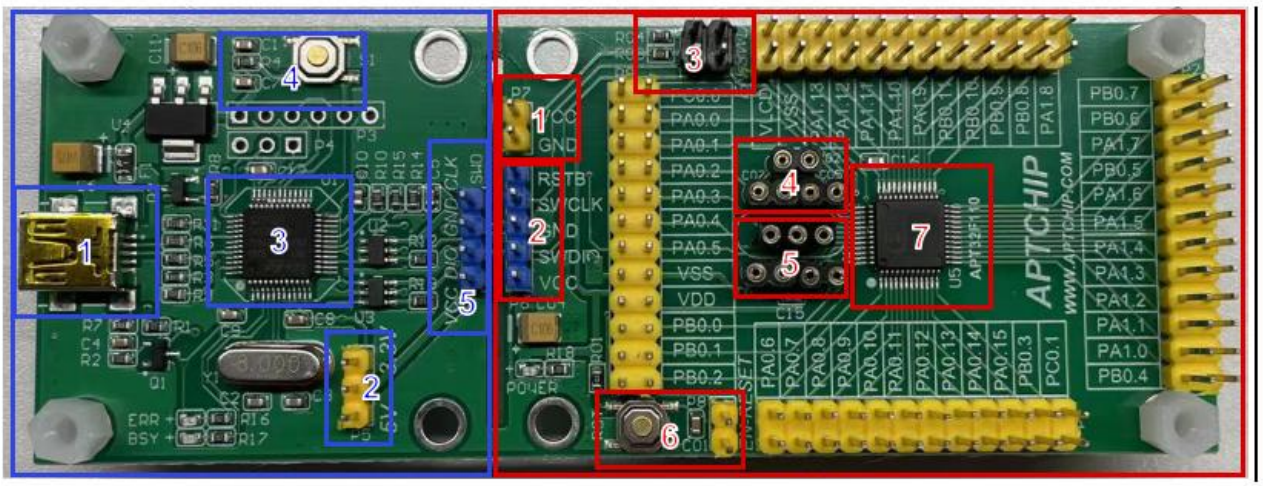


Figure 1 开发板实物图

3.1 调试模块（蓝色部分）

1、 miniUSB 口，用以连接 PC。

2、 选择系列模块的供电电压。系列模块的工作电压来自调试模块，5V/3V3 可选。

3、 调试主芯片。

4、 调试主芯片复位按键。

5、 SWD 调试接口。

3.2 芯片（红色部分）

1、 电源。可用于调试时飞线，电压取决于调试模块 2 的选择

2、 SWD 调试接口。与调试模块连接的接口。

3、 SWD 跳线。APT32F171x 系列只有一组 SWD 接口。SWD 口为 PA0.4（SWCLK）和 PA0.3（SWDIO）。这组 SWD 跳线用来控制来自调试模块的 SWD 信号要不要连接芯片的 PA0.3 和 PA0.4。使用时需用跳线连接

4、 晶振电路和跳线。APT32F171x 系列支持外部晶振。但因为内部也有时钟，所以外部晶振不是必须的。需要外部晶振时，这里预留了晶振和电容接插口，可以直接插入对应晶振和电容，这里是副晶振。

5、 晶振电路和跳线。APT32F171x 系列支持外部晶振。但因为内部也有时钟，所以外部晶振不是必须的。需要外部晶振时，这里预留了晶振和电容接插口，可以直接插入对应晶振和电容，这里是主晶振。

6、 复位电路和跳线。APT32F171x 系列支持外部复位脚复位。但因为同时支持 POR，所以外部复位不是必须的。当复位脚对应的管脚用作它途时，需要将这个跳线断开。

7、 芯片。APT32F171x 系列芯片。

4. 软件环境

4.1 CDK开发环境

APT32F171x系列使用平头哥的剑池CDK作为软件开发平台。

CDK下载地址：https://occ.t-head.cn/community/download?id=575997419775328256

按照提示安装即可。有几个注意事项：

- CDK不支持中文路径。
- 如果您的电脑使用了如360之类的杀毒软件，除了在安装过程中允许CDK的操作之外，安装之后，必须将整个CDK安装目录加入到杀毒软件的白名单区。
- 如果需要增减工程文件，必须在CDK工程视图下完成。如果在windows文件目录中操作（复制粘贴）后编译会出现错误。

4.2 代码结构

下图为工程视图。这里面包含了6个组件。

apt32f171x是工程组件，客户的业务逻辑代码一般放在Packages这个组件下面。

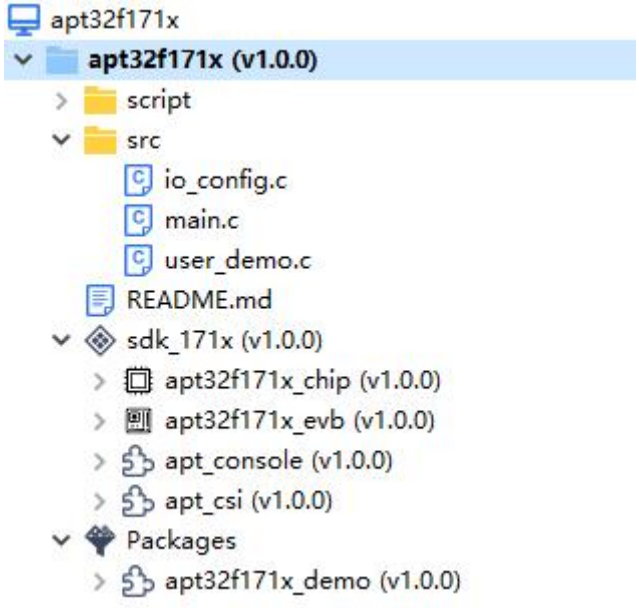


Figure 2 代码结构

5. 例程运行

5.1 例程获取方式

5.1.1 SDK包获取

APT32F171x的SDK可以在CDK中安装插件获得（CDK的版本请确保是最新版本，以下步骤中所使用的CDK版本为V2.16.2），具体步骤如下：

1、打开CDK，在如下路径找到APT32F171x的SDK插件。

2、默认的插件状态显示”Not Install”，点击一下，插件就会自动安装，安装完毕后，插件状态显示”Installed”。

3、点击OK结束插件安装。

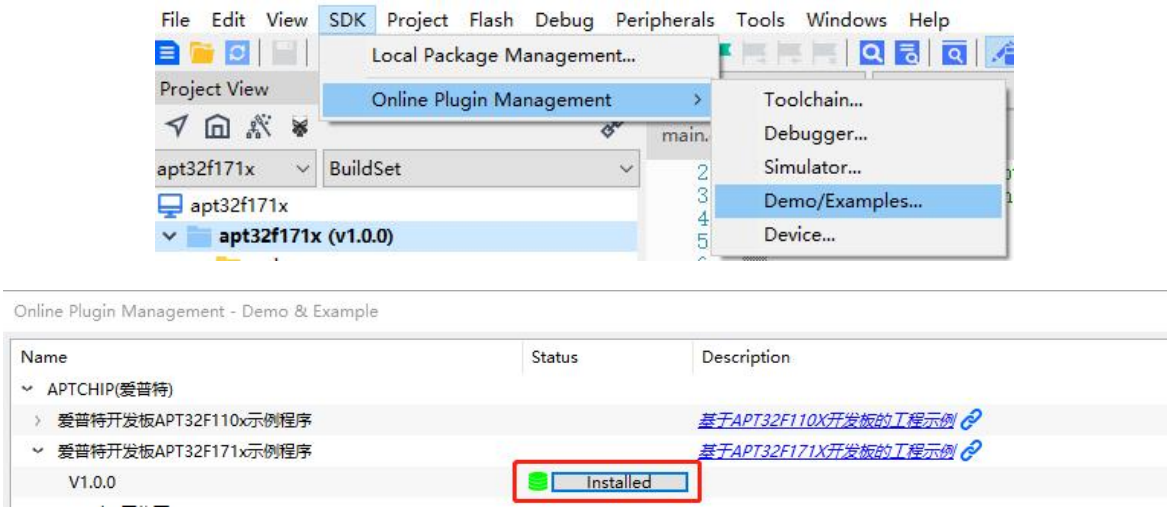
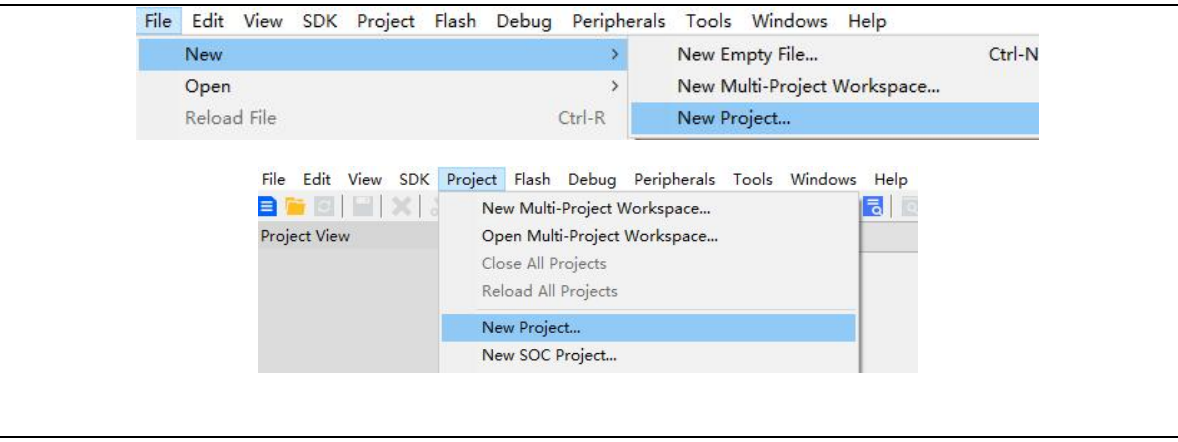


Figure 3 创建新的 workspace

5.1.2 工程创建方式

1、 打开CDK，创建一个新的Project到一个目录，可以通过”File->New->New Project”或者”Project->New Project”两种方式创建一个新的Project。



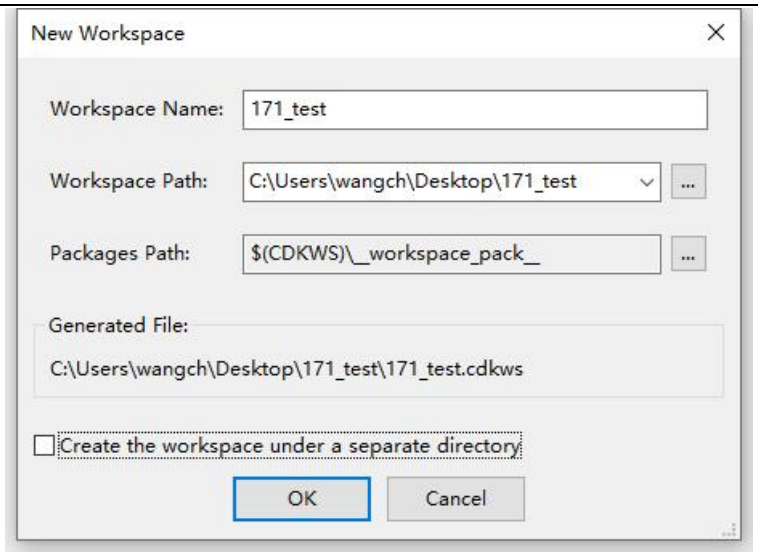


Figure 4 创建新的 Project

2、 在接下来弹出的界面中配置好工程参数，参考下图进行配置，点击OK结束配置界面。

注意：在输入工程名之前，要先选中下方的”apt32f171x_demo”。

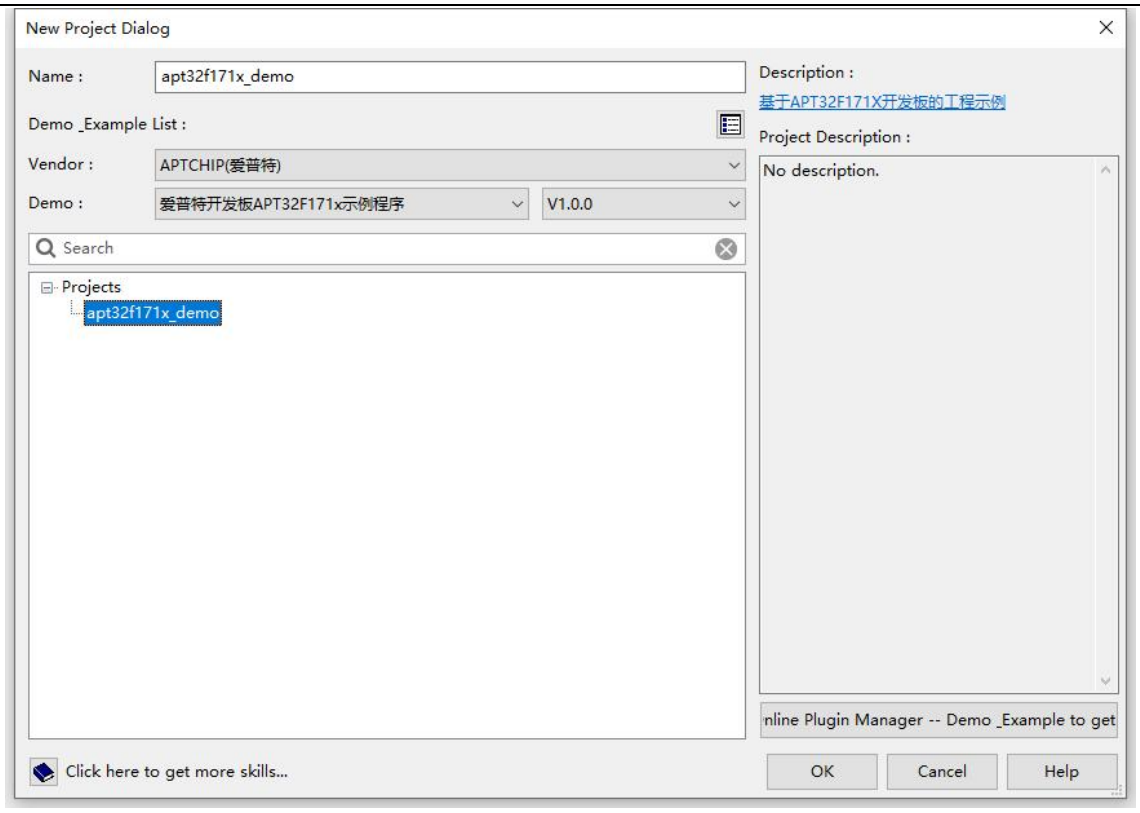


Figure 5 配置工程参数

3、 到这里，我们就完成了APT32F171x工程的创建，此时在CDK左边栏我们就可以看到工程的目录树。

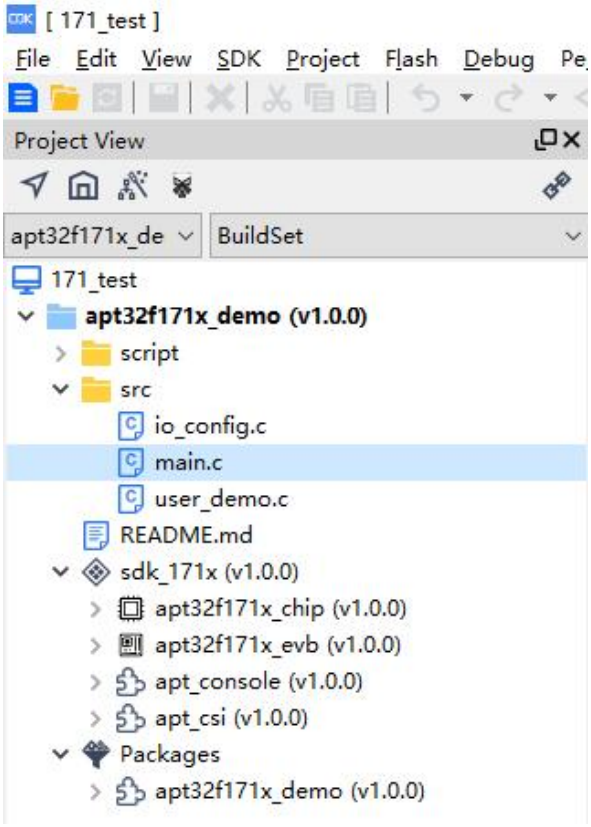


Figure 6 创建好的工程目录树

5.1.3 工程配置

如果用户是按照上述步骤创建工程，则工程中已经完成了所有必要的工程配置，可忽略本节内容，直接编译下载即可。

注意：因CDK版本不同，部分配置界面可能不太一样。

1. 配置选项Compiler

- 点击图中标注1，弹出配置对话框
- 标注2、3按照图中配置所选择
- 标志4中的配置选项在SDK包中的board文件夹下global config文件中，将里面的配置信息copy到4中的配置选项里。标注5建议勾选。

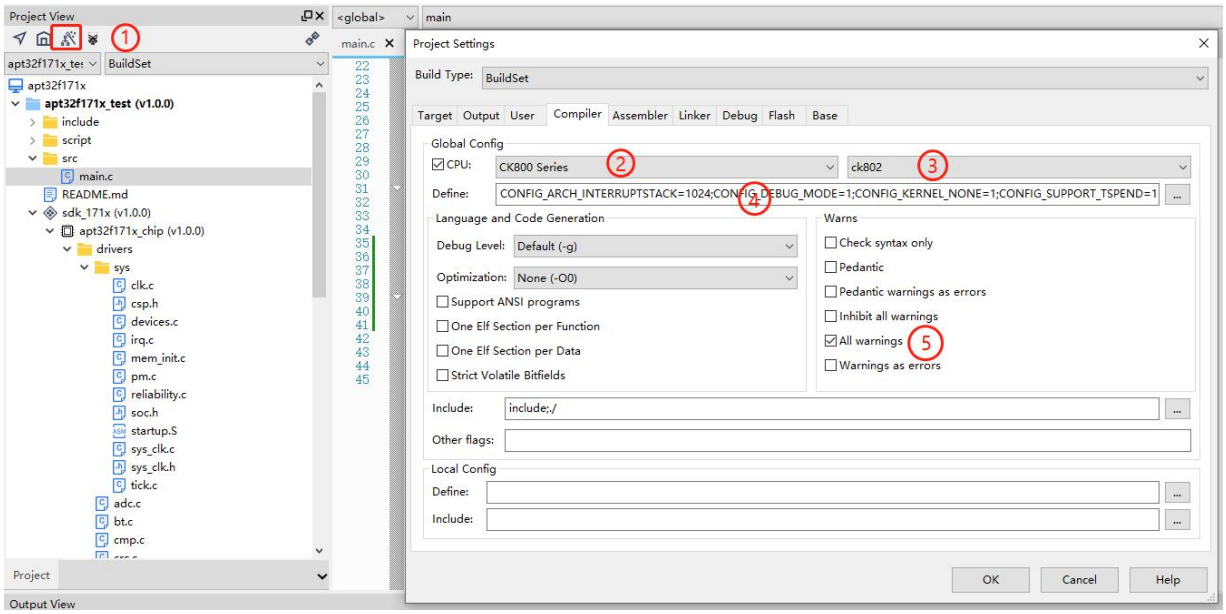


Figure 7 工程配置

2. 配置选项Output

- 标注6，7选择需要的输出文件

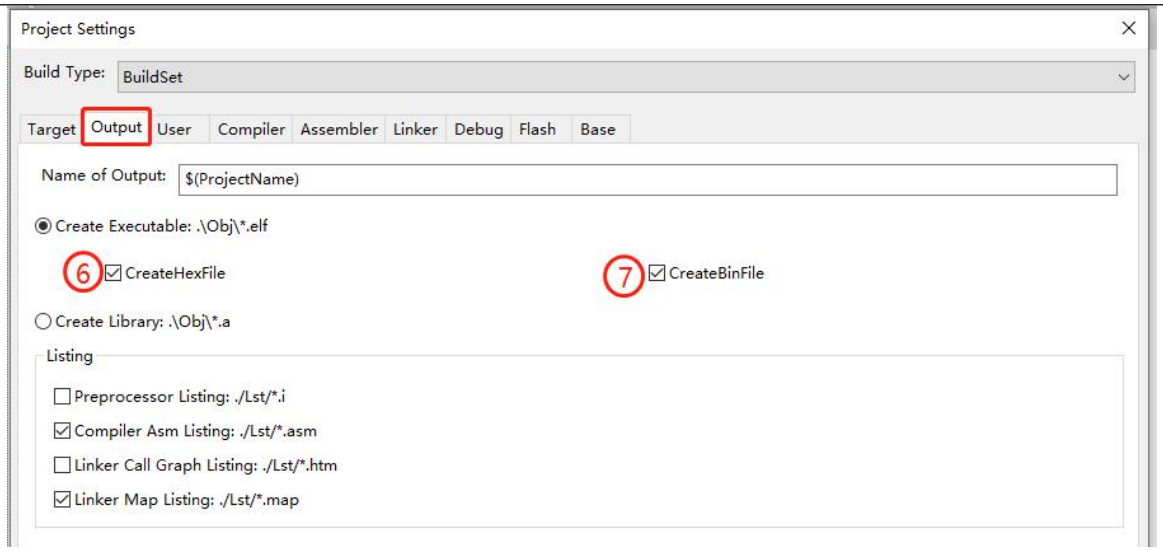


Figure 8 工程配置

3. 配置选项Linker

- 标注8选择Linker文件,一般在工程路径的board目录下，名称叫gcc_flash.ld。结合自己的工程存放位置选择

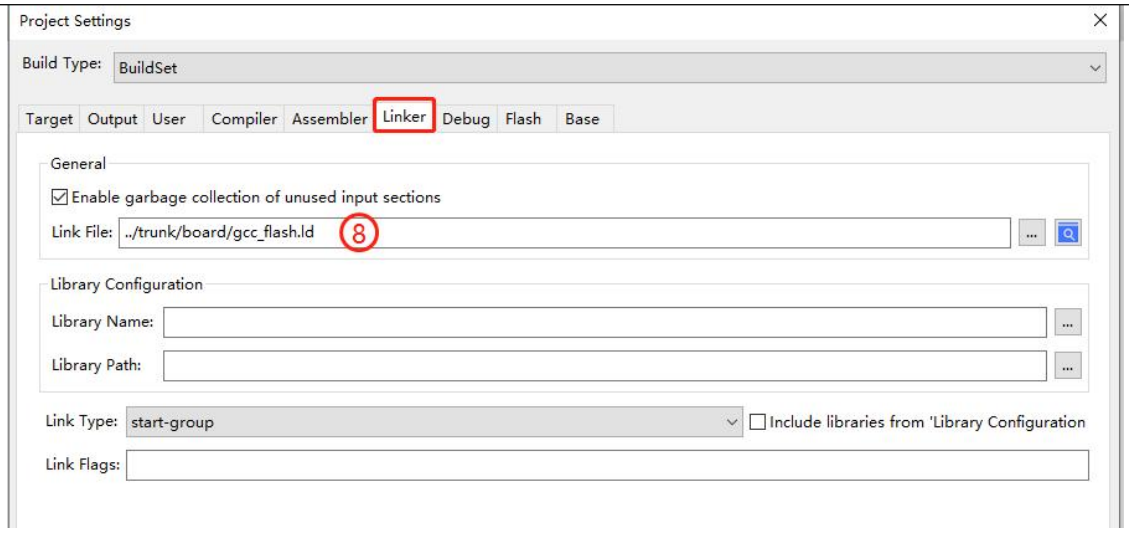
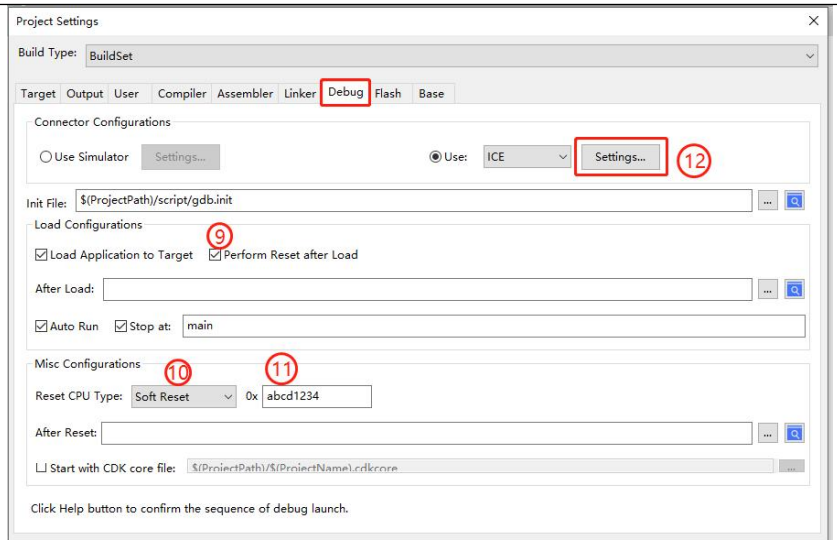


Figure 9 工程配置

4. 配置选项Debug

- 标注9,10,11按图所示配置
- 点击标注12，标注13，14按图所示配置



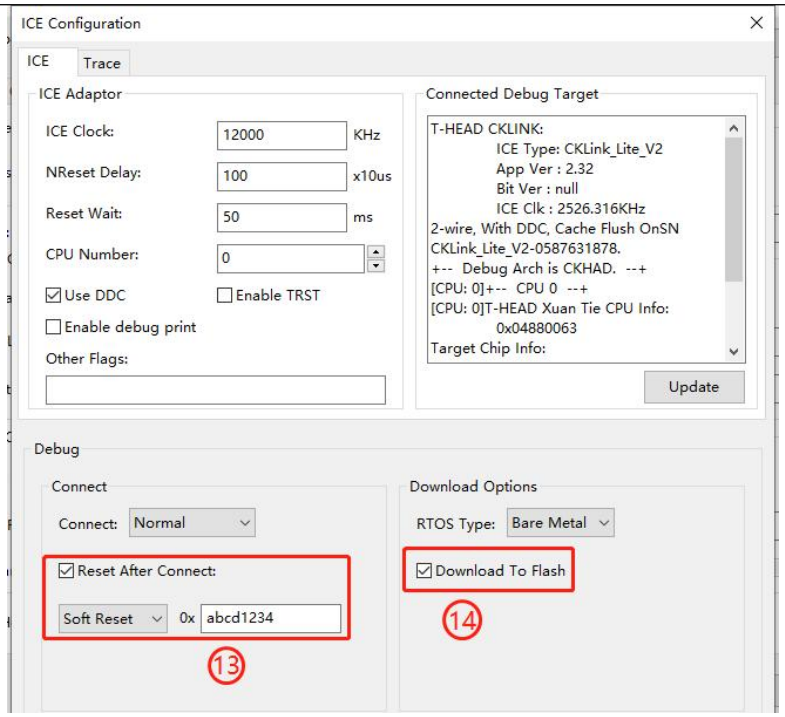


Figure 10 工程配置

5. 配置选项Flash

- 标注15主要是和下载相关的配置，按图配置即可

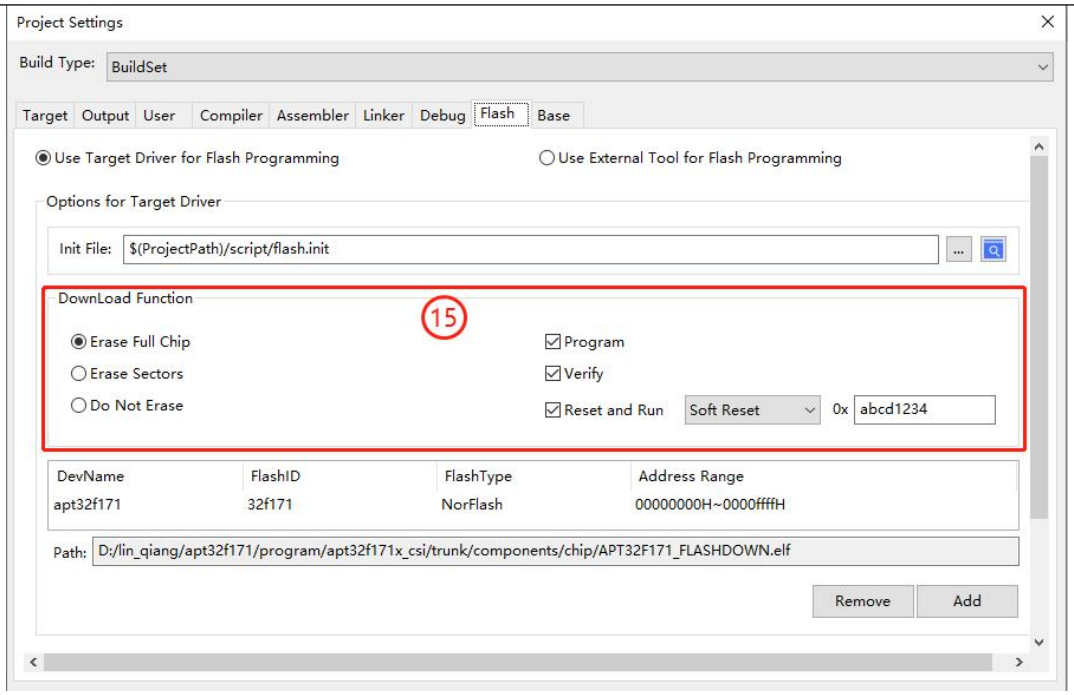


Figure 11 工程配置

5.2 导入flash算法

CSI驱动代码使用了SOC工程结构，SDK中已包含工程所需要的算法文件。所以无需额外导入。

5.3 适配代码

这部分说明涉及最基础的嵌入式编程相关的内容，包括系统初始化和中断实现。

Main()函数需要调用两个函数，system_init()和board_init()，如下图所示。分别完成时钟配置和串口配置的工作。

```
#include "iostring.h"
#include "csi_drv.h"
#include "board_config.h"
#include "demo.h"

//------------------------------------------------------------
extern void system_init(void);
extern void board_init(void);
extern void user_demo(void);
extern void __ChipInitHandler(void);
//------------------------------------------------------------


int main()
{
    system_init();
    __ChipInitHandler();                    //Graphical interface initialization
    board_init();

    csi_pin_set_mux(PC01, PC01_OUTPUT);     //PC01 output
    csi_pin_set_high(PC01);                 //PC01 output high;

    my_printf("hello apt32f171!\n");

    user_demo();                            //demo


    while(1)
    {
        mdelay(100);                        //delay 100ms
        csi_pin_toggle(PC01);               //PC01 toggle
    }

    return 0;
}
```

Figure 12 main 函数

5.3.1 时钟配置

示例工程中提供了时钟配置，可以根据应用增减，但下图中黄色背景的为必须有的步骤。

函数	说明	位置

system_init()	__attribute__((weak)) void system_init(void)
	{
	CK_CPU_DISALLNORMALIRQ; //关闭全局中断
	CSI_iwdt_close(); //关看门狗 iwdt（调试时用）
	CSI_sysclk_config(); //配置系统时钟（必须有）
	CSI_get_sclk_freq(); //更新系统时钟 SCLK 的值（必须有）
		system.c

```
    CSI_get_pclk_freq();   //更新外设时钟 PCLK 的值（必须有）
    CSI_tick_init();       //初始化 systick
    CK_CPU_ENALLNORMALIRQ; //开启全局中断
}
```

其中csi_sysclk_config()会去读取一个时钟配置结构体变量tClkConfig（位于board_config.c）。可以通过修改tClkConfig来实现系统和外设主时钟的配置。下图是默认配置。

```
/// system clock configuration parameters to define source, source freq(if selectable), sdiv and pdiv
csi_clk_config_t tClkConfig =
    {SRC_HFOSC, HFOSC_48M_VALUE, SCLK_DIV1, PCLK_DIV1, 48000000, 48000000};
    //{SRC_EMOSC, EMOSC_VALUE, SCLK_DIV1, PCLK_DIV2, 5556000, 5556000};
    //{SRC_IMOSC, IMOSC_5M_VALUE, SCLK_DIV1, PCLK_DIV1,5556000, 5556000};
    //{SRC_IMOSC, IMOSC_4M_VALUE, SCLK_DIV1, PCLK_DIV1,4194000, 4194000};
```

Figure 13 时钟配置结构体

5.3.2 串口配置

仿真环境下，支持两种串口信息的输出。

1. 虚拟串口，即debug print，不需要连接外部的硬件串口。使用时需要勾选semi host配置，并且打印的时候使用printf，例如printf("hello world!\n")，调试时打开debugger pane界面，在output页面查看调试信息

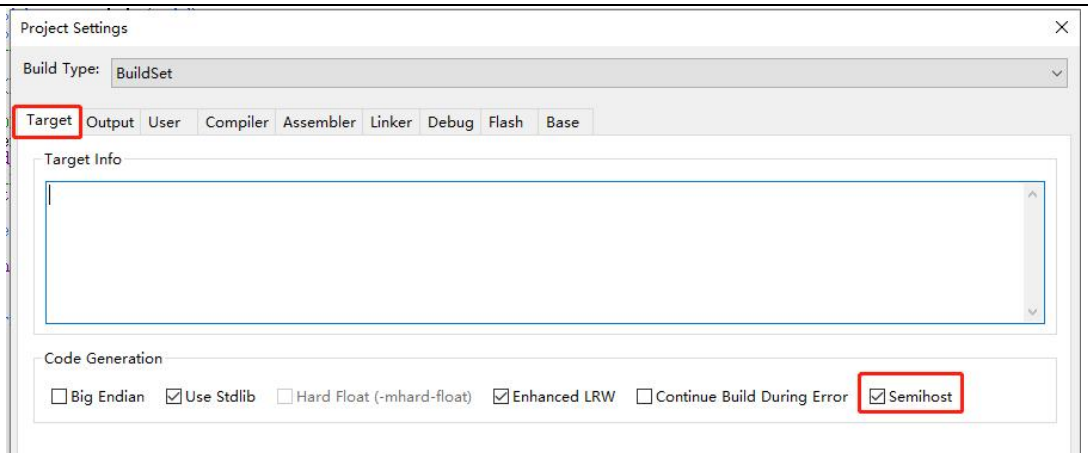


Figure 14 勾选 Semihost

2. 真实的串口输出，使用真实的硬件串口。使用时需要确保全局宏DBG_PRINT2PC=1（见Figure7）存在。调试串口数据格式固定为数据位8位，停止位1位，不校验。默认使用UART0，PA11（TXD）和PA12（RXD）。注意查看MCU 型号是否有UART0外设。UART口及对应的管脚资源可改。如果要调整串口资源，需要在board_config.h中更改相应的宏，并调整硬件连接。

   打印的时候使用my_printf，例如my_printf("hello world!!\n");

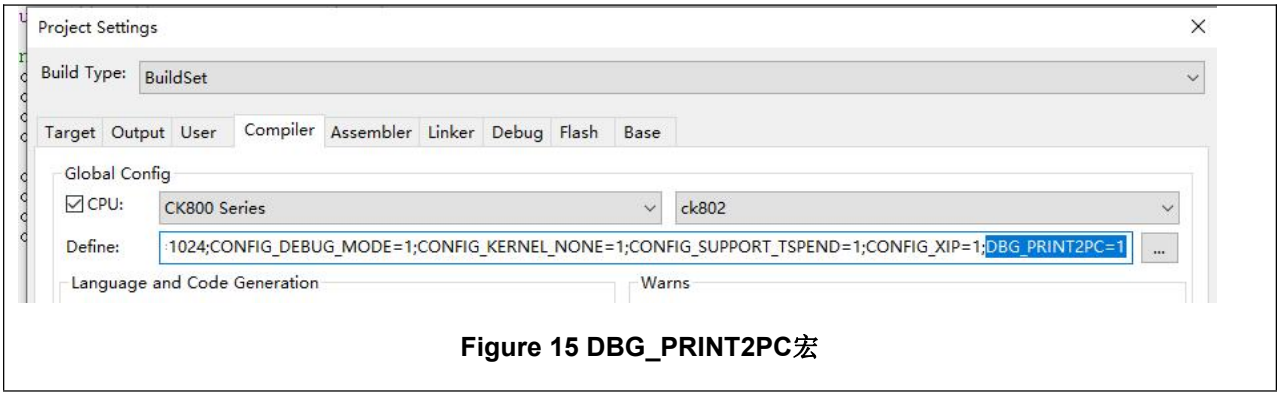


Figure 15 DBG_PRINT2PC宏

函数	说明	位置

board_init()	实现 UART 硬件配置，包括 UART id、波特率、管脚等。
	所有的宏在 board_config.h 中定义。
	__attribute__((weak)) void board_init(void)
	{
	//console config for print
	console.uart_id = (uint32_t)CONSOLE_IDX;
	console.baudrate = 115200U;
	console.tx.pin = CONSOLE_TXD;
	console.tx.func = CONSOLE_TXD_FUNC;
	console.rx.pin = CONSOLE_RXD;
	console.rx.func = CONSOLE_RXD_FUNC;
	console.uart = (csp_uart_t *)(APB_UART0_BASE + CONSOLE_IDX * 0x1000);
	console_init(&console);
	}
		board_config.c

5.3.3 中断函数

中断处理函数位于board/src/interrupt.c中。这个文件搭建了中断处理函数的框架，特别是某些功能实现和中断强相关的外设，如BT。

```
void BT0IntHandler(void)
{
    // ISR content ...
    bt_irqhandler(BT0);
}
```

Figure 16 中断处理函数举例

在上述例子中，bt_irqhandler()函数在驱动代码（bt.c）中定义，但具有weak属性。我们推荐用户使用驱动中已经定义的中断处理函数。但在一些特殊的场合，用户仍然可以以同样的名字对这个函数重新定义，而不需要修改BT0IntHandler内部的代码。此时，编译会忽略weak属性的预定义函数，将用户新写的同名函数加入编译。

另外需要注意的是，中断函数的进出会比普通的函数调用有更多的步骤，会消耗更多的代码资源。所以,如果

- 应用对代码大小敏感，可以删掉无用的中断处理函数。以ifc_irqhandler ()为例，如果应用中没有用到IFC相关的中断，那么删除这部分代码的步骤如下：

1、 interrupt.c中删除void IFCIntHandler(void)

2、 srartup.S中将原来IFCIntHandler替换为DummyHandler

- 应用对运行时间敏感，除了尽量减少中断函数内部的处理外，还可以考虑减少函数调用层级。

5.4 GPIO可视化配置

5.4.1 简介

171x支持GPIO初始化的可视化配置，可视化配置在工程evb组件的svc文件夹下，即下图指示，里面包含171x系列所有封装信息；若用户用SDK包新建工程，工程建立后，需要将svc目录下chip_config_dll.dll文件拷贝到工程目录下。

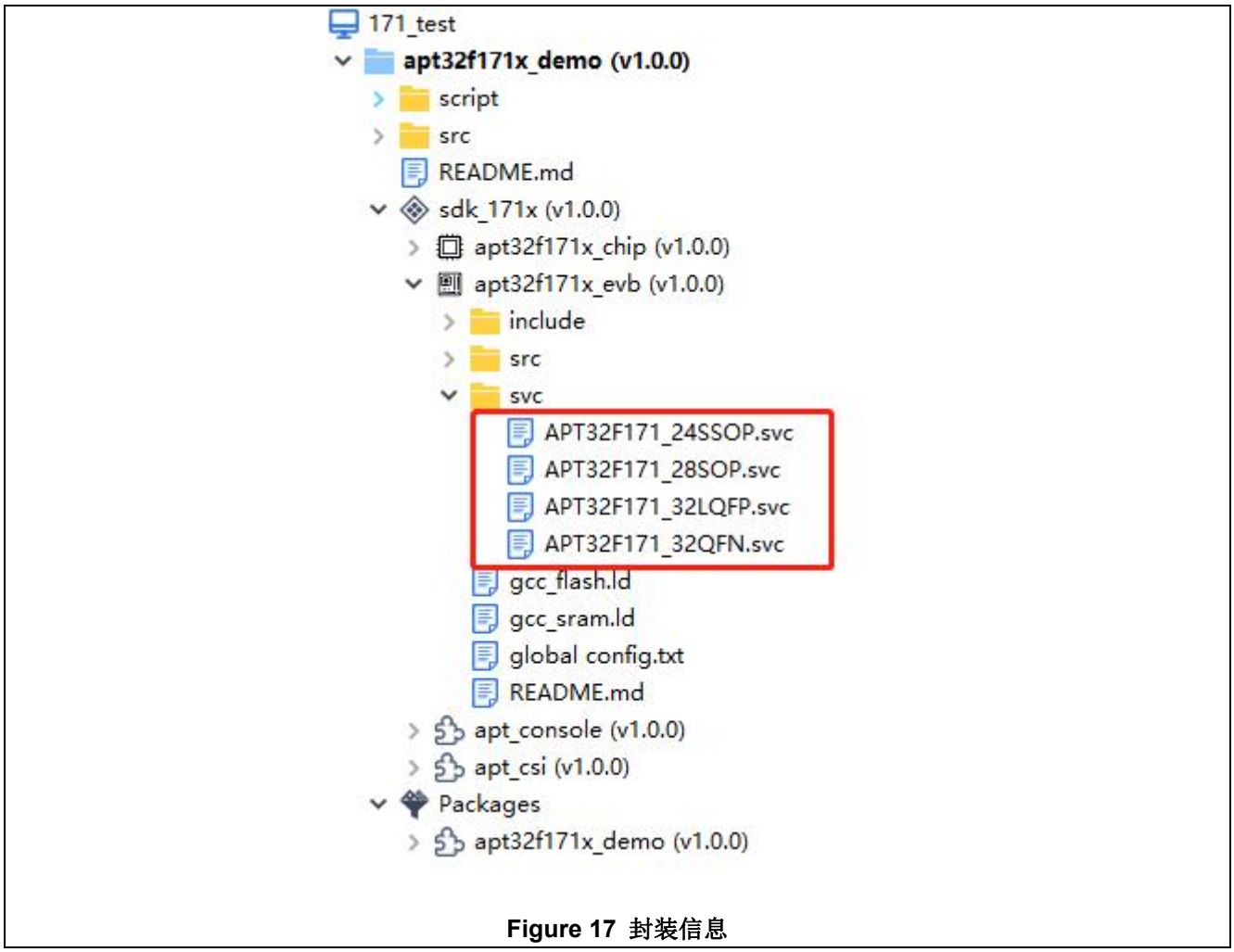


Figure 17 封装信息

5.4.2 GPIO配置成复用功能

以下配置以APT32F171的QFN32封装为例（最大管脚资源），配置12脚（PA0.8）PinFunctions:CLO，

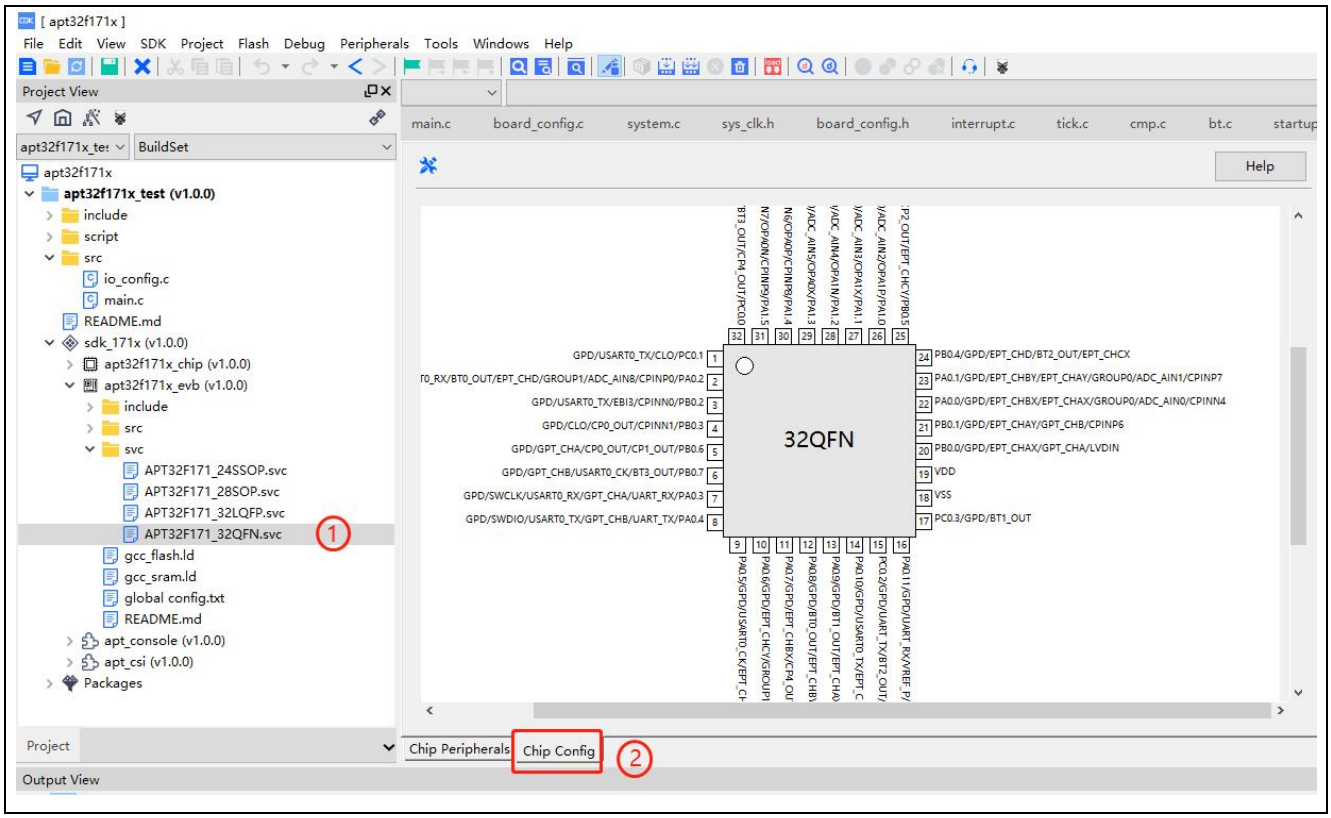
Pullup : pull none， Direction : output。

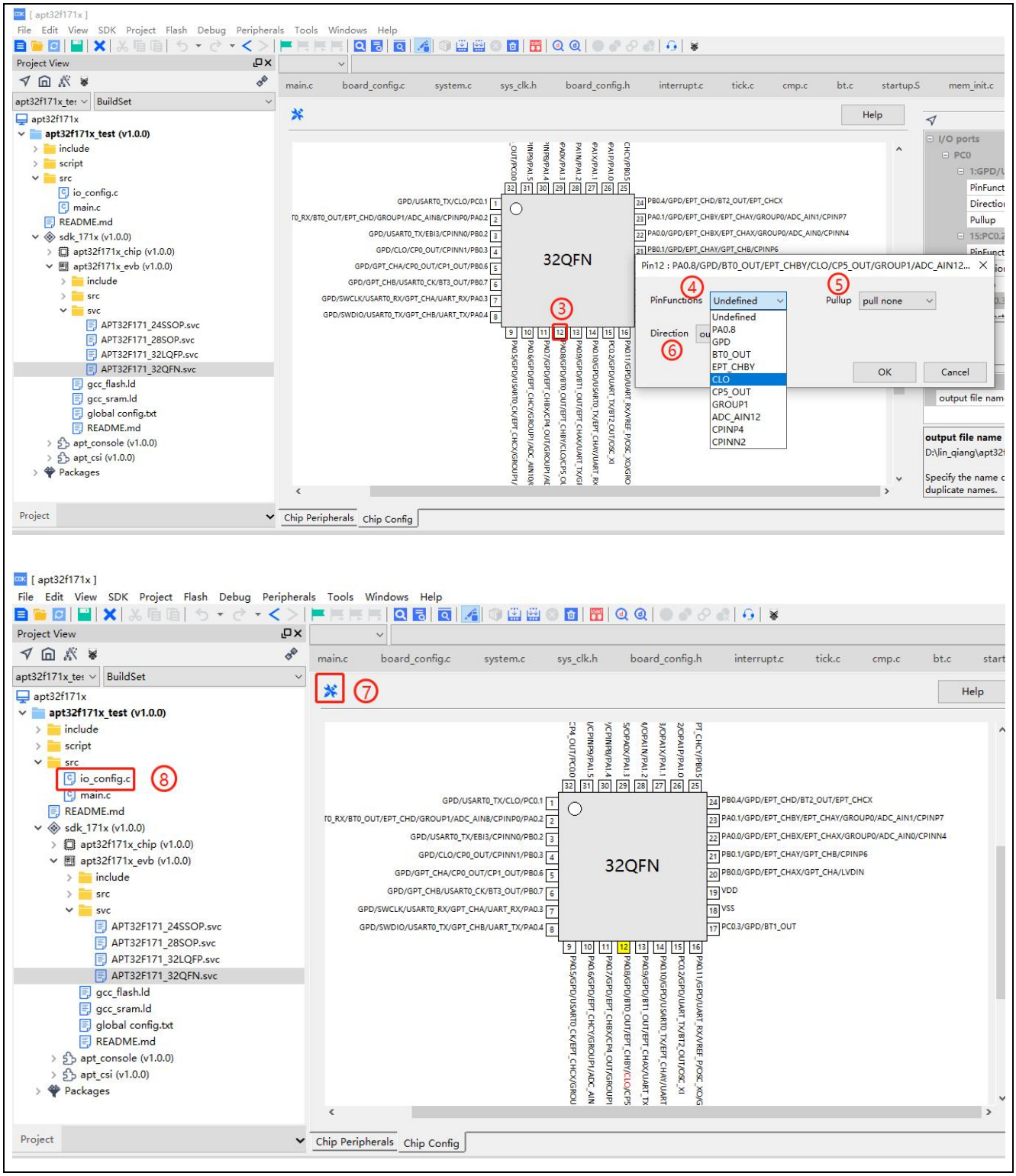
1. 点击图中红色1标注地方：apt32f171x_evb/svc下的APT32F171_QFN32；再点击红色2标注的地方Chip Config。
2. 双击图中红色3标注的管脚12，在弹出的对话框中红色标注4选择需要配置的功能，在红色标注5选择管脚的上拉/下拉/禁止上下拉功能
3. 点击红色标注6 设置输入还是输出，最后点击OK，完成配置。

4. 再点击图中红色标注7的蓝色图标，在弹出对话框中点yes选项，将在工程根目录src文件夹下生成io_config.c文件。配置完成后，对应管脚的颜色将会变成黄色，对应选择功能会变成红色。

5. io_config.c中生成两个函数PinConfigInit和__ChipInitHandler函数。配置语句为图中红色标注9，配置语句在PinConfigInit函数中，用户初始化时直接调用即可。

6. 编译时候如果提示找不到对应参考函数，则需要添加工程目录下的io_config.c文件(添加一次即可)，本示例中，该文件在D:\lin_qiang\apt32f171\program\apt32f171x_csi\apt32f171x_test\src,详细操作参考标注10，11，12




[apt32f171x]
File Edit View SDK Project Flash Debug Peripherals Tools Windows Help
Project View
apt32f171x
apt32f171x_test (v1.0.0)
include
script
src
io_config.c
main.c
README.md
sdk_171x (v1.0.0)
apt32f171x_chip (v1.0.0)
apt32f171x_evb (v1.0.0)
include
src
svc
APT32F171_24SSOP.svc
APT32F171_28SOP.svc
APT32F171_32LQFP.svc
APT32F171_32QFN.svc
gcc_flash.ld
gcc_sram.ld
global config.txt
README.md
apt_console (v1.0.0)
apt_csi (v1.0.0)
Packages
Project
BuildSet
main.c
board_config.c
system.c
sys_clk.h
board_config.h
interrupt.c
tick.c
cmp.c
bt.c
startup.S
mem_init.c
Help
32QFN
Pin12 : PA0.8/GPD/BT0_OUT/EPT_CHBY/CLO/CP5_OUT/GROUP1/ADC_AIN12...
PinFunctions
Undefined
Pullup
pull none
Direction
Undefined
PA0.8
GPD
BT0_OUT
EPT_CHBY
CLO
CP5_OUT
GROUP1
ADC_AIN12
CPINP4
CPINN2
OK
Cancel
output file name
Specify the name c
duplicate names.
Chip Peripherals
Chip Config
①
②
③
④
⑤
⑥
⑦
⑧

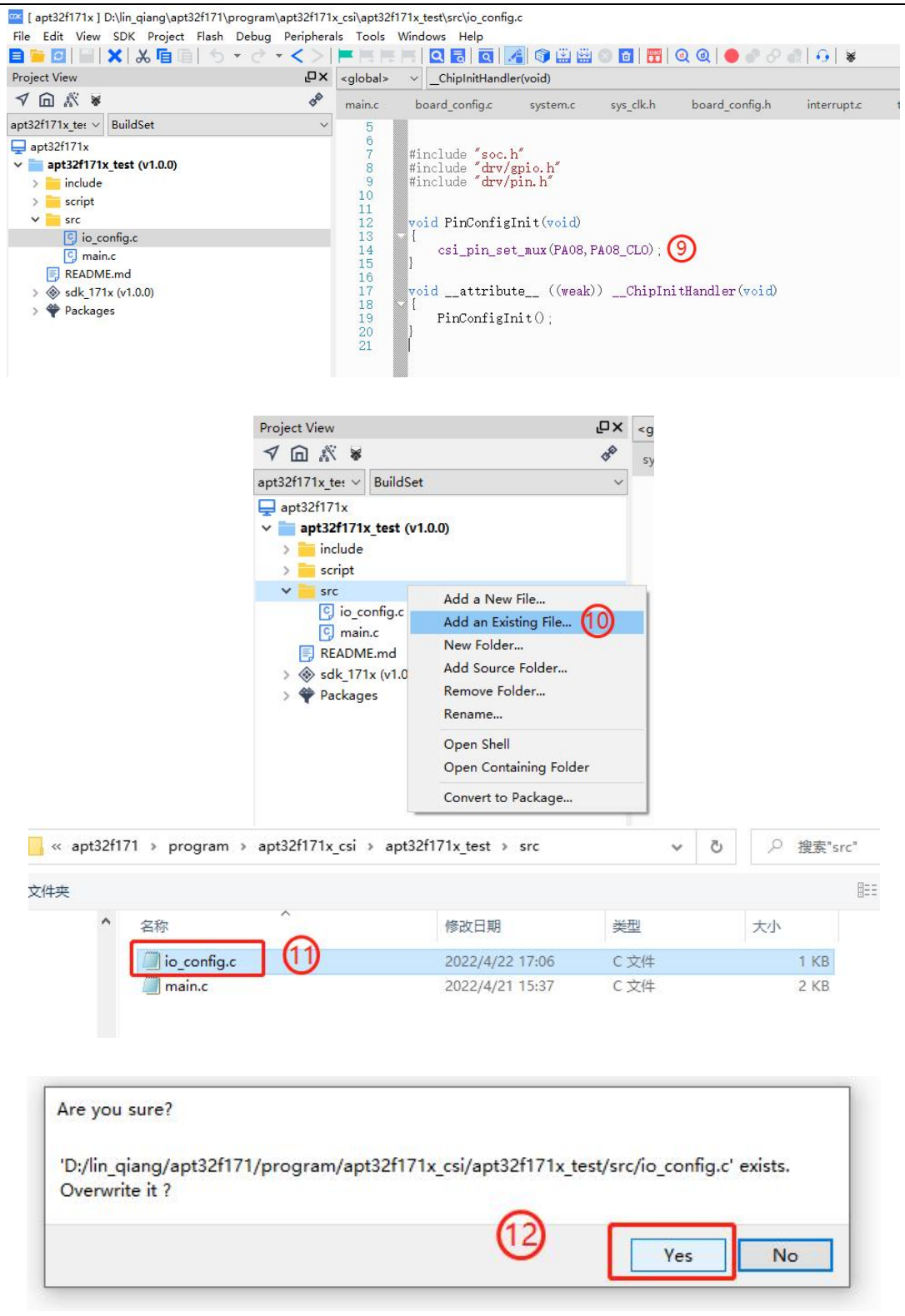


Figure 18 可视化配置步骤

5.4.3 GPIO配置为输入/输出

GPIO配置为输入/输出功能时，具体步骤和复用功能基本相同，步骤中不同的是配置步骤图片中的第二幅图，具体实例以配置32脚（PC0.1）为OUTPUT，下拉使能。

配置完成后，对应管脚的颜色将会变成黄色，对应选择功能会变成红色。io_config.c中函数PinConfigInit中将生成对应的配置语句。

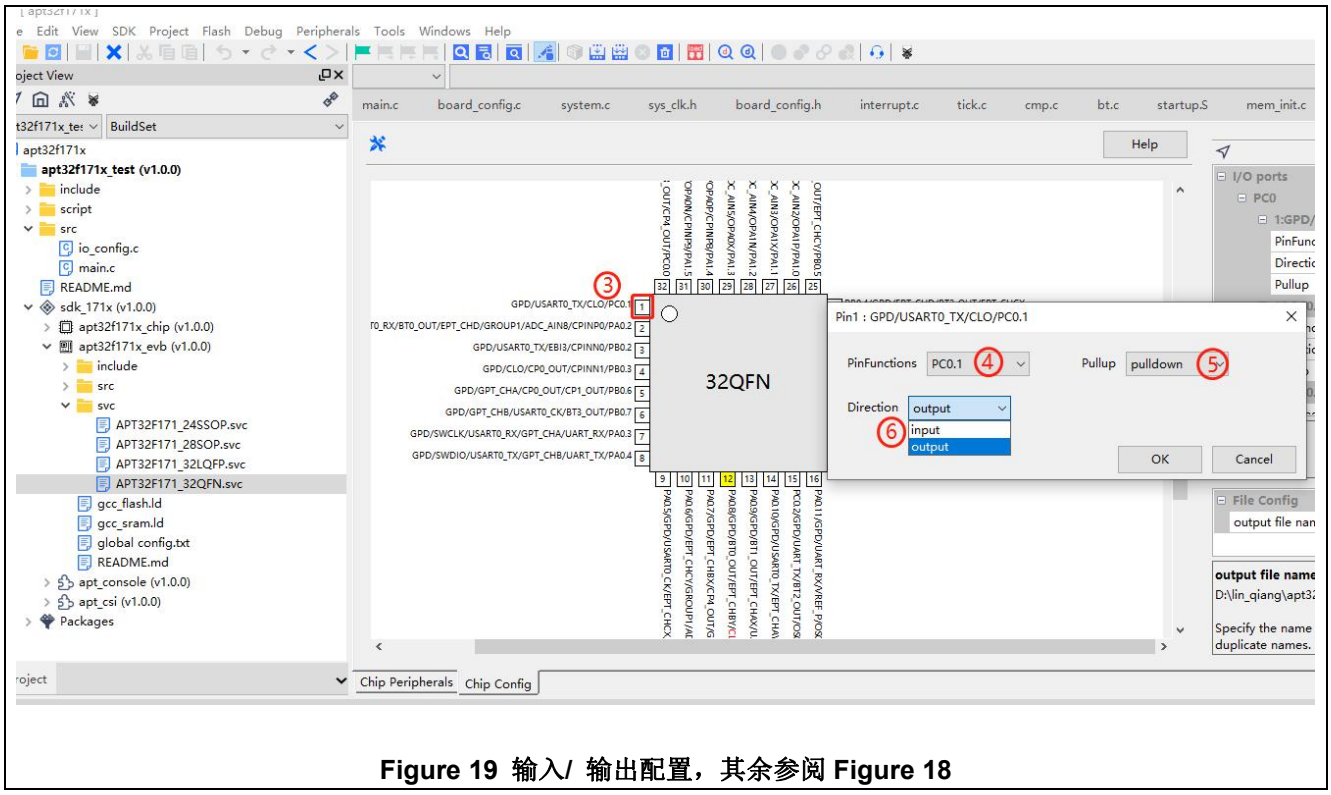


Figure 19 输入/ 输出配置，其余参阅 Figure 18

5.5 外设使用

外设使用的示例代码在APT32F171x_demo这个组件中。user_demo()函数罗列了所有的示例函数，可以通过“打开、关闭”注释的方式调用相应的示例函数。

5.6 编译工程

点击“Build Project”既可实现工程的编译和连接。

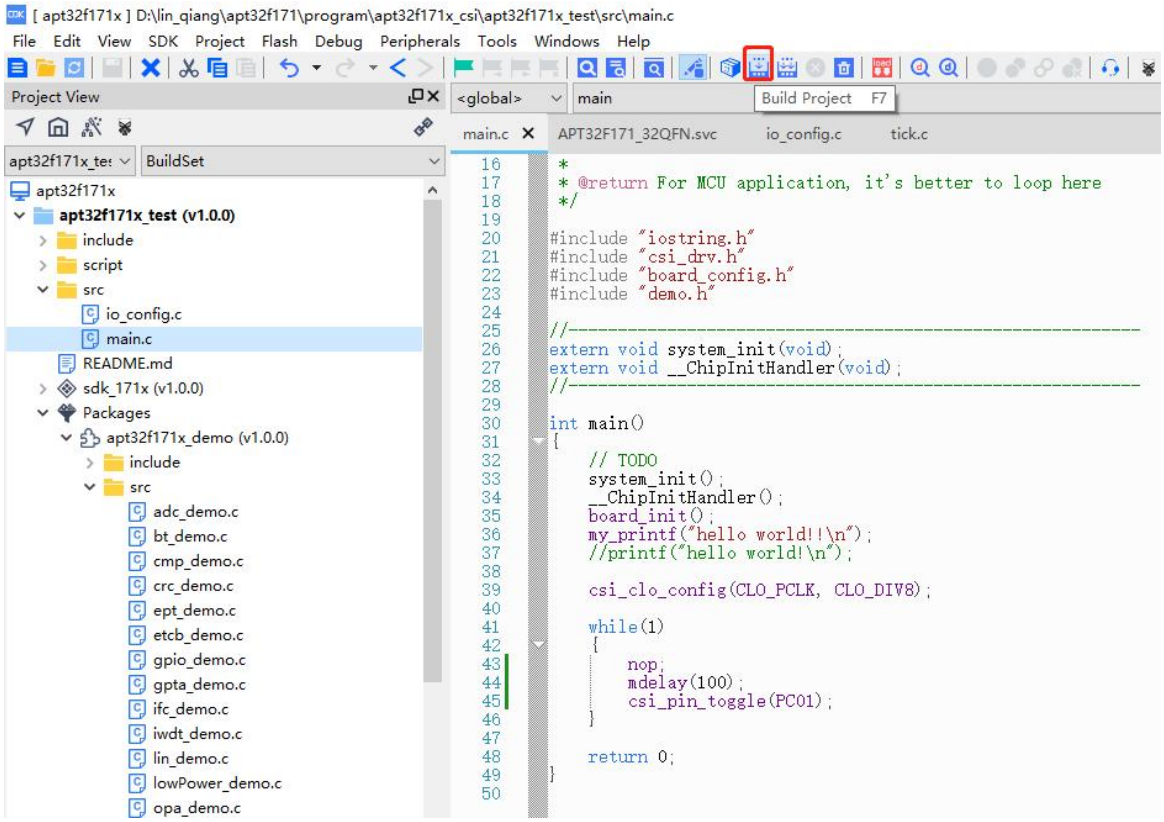


Figure 20 build

5.7 下载调试

视应用情况，可以选择三种下载方式：

1. 将代码下载到flash区，不进入debug模式
2. 将代码下载到flash区，随后进入debug模式
3. 仍然使用芯片内flash数据，直接进入debug模式（这个方式可用来回读芯片内flash内容）

下图中的1，2，3分别对应上面三种下载方式的操作菜单。

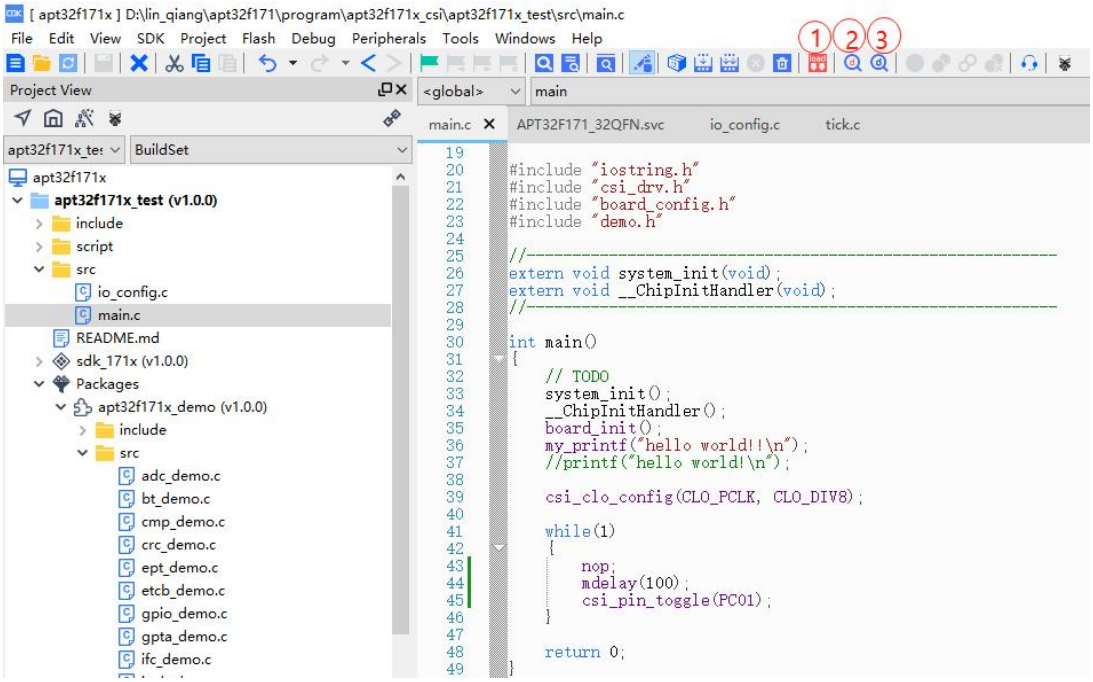


Figure 21 下载调试的三种选择