版本	V1.2
日期	202301

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
V1.2	2023-1-9	1、 添加demo函数的说明 2、 更新部分示意图 3、 修改错误的描述 4、 更新页眉页脚	wangch

目录

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、电气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。

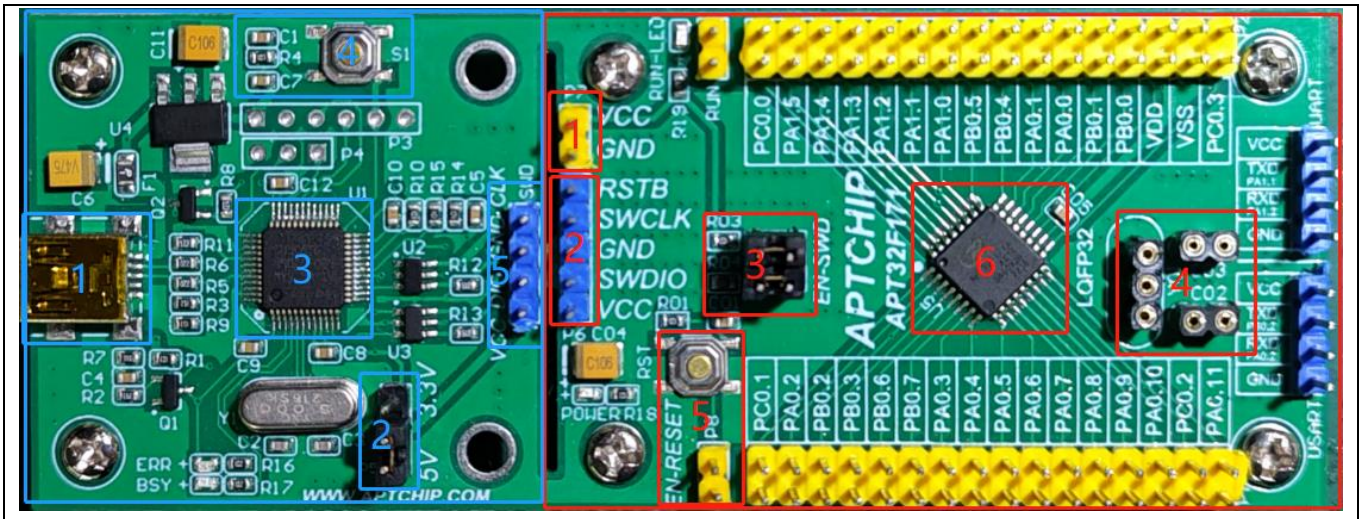


Figure 1 开发板实物图

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（SWDIO）和 PA0.3（SWCLK）。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。

5、 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。

6、 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/product?id=3864775351511420928&type=soft

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。这里面包含了不同的组件，包括chip组件、evb组件、demo组件等等，在demo组件下我们提供了171所有模块的使用示例。

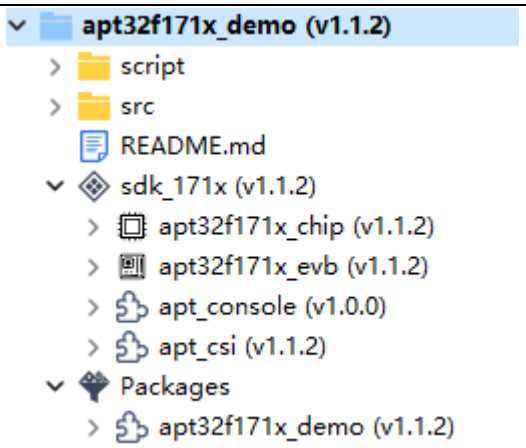


Figure 2 代码结构

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，然后点击下方的”Update Remote Repo”刷新一下（因为SDK版本会不定时更新，所以要先刷新到最新的SDK版本）。

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。

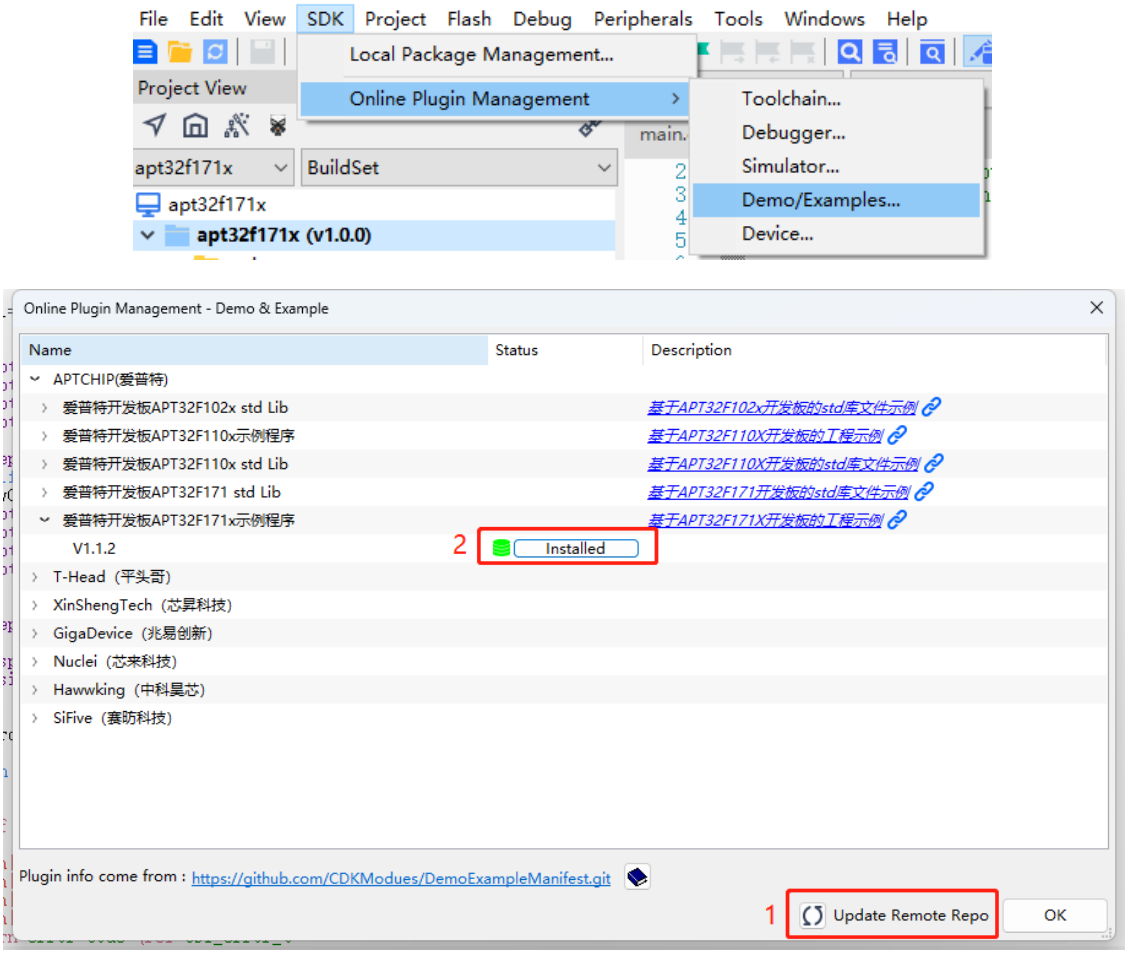


Figure 3 创建新的 workspace

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。

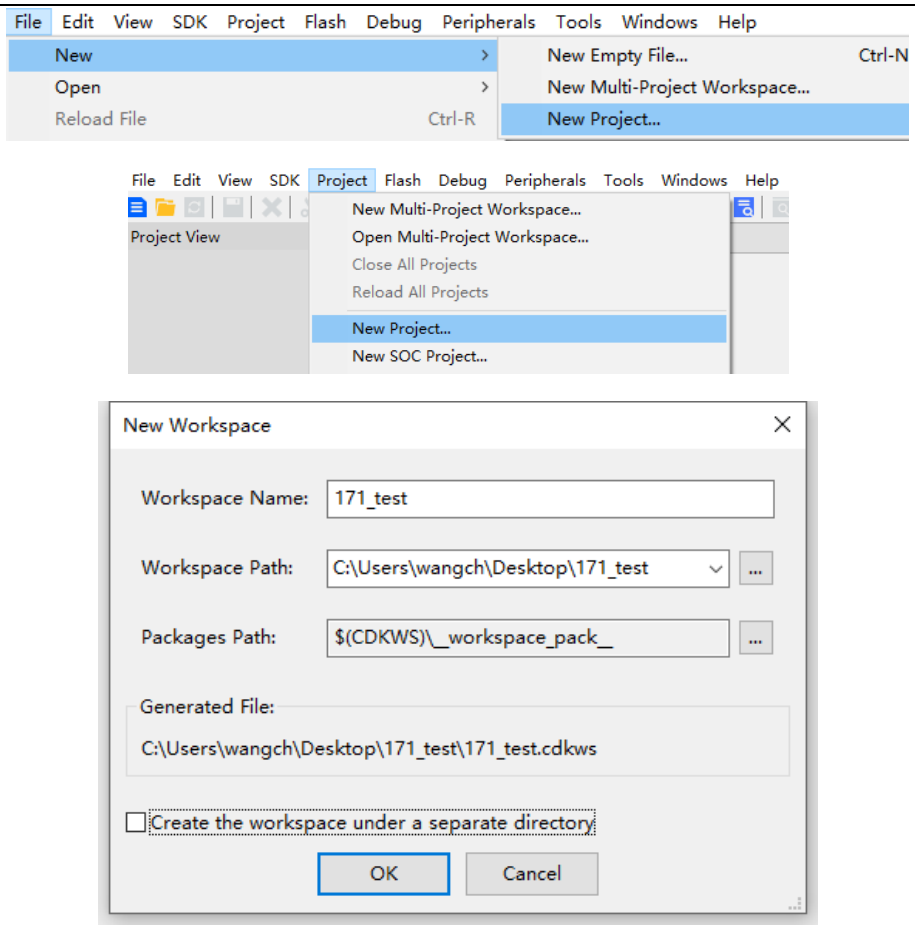


Figure 4 创建新的 Project

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”。

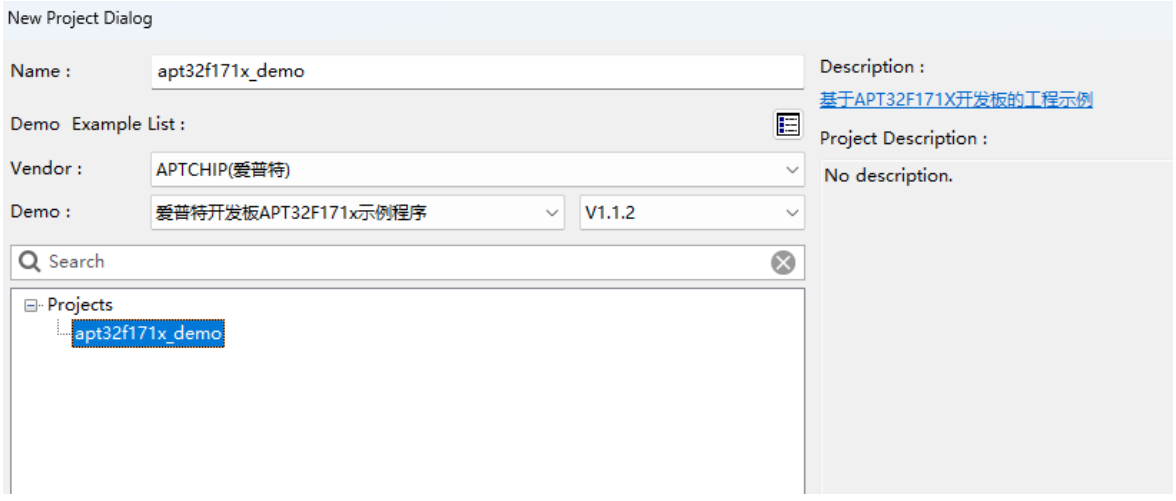


Figure 5 配置工程参数

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。

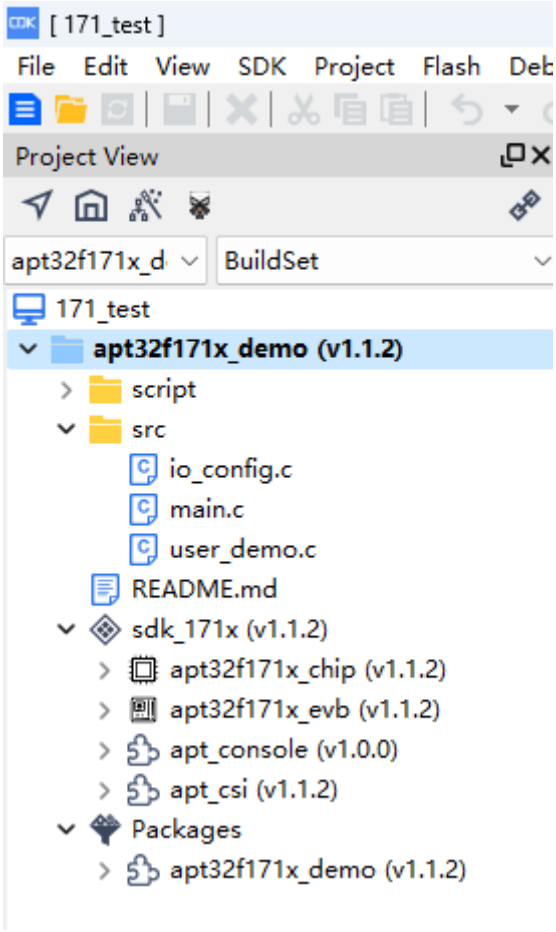


Figure 6 创建好的工程目录树

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
- 点击图中标注2
- 标注3按照图中配置所选择
- 标注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。

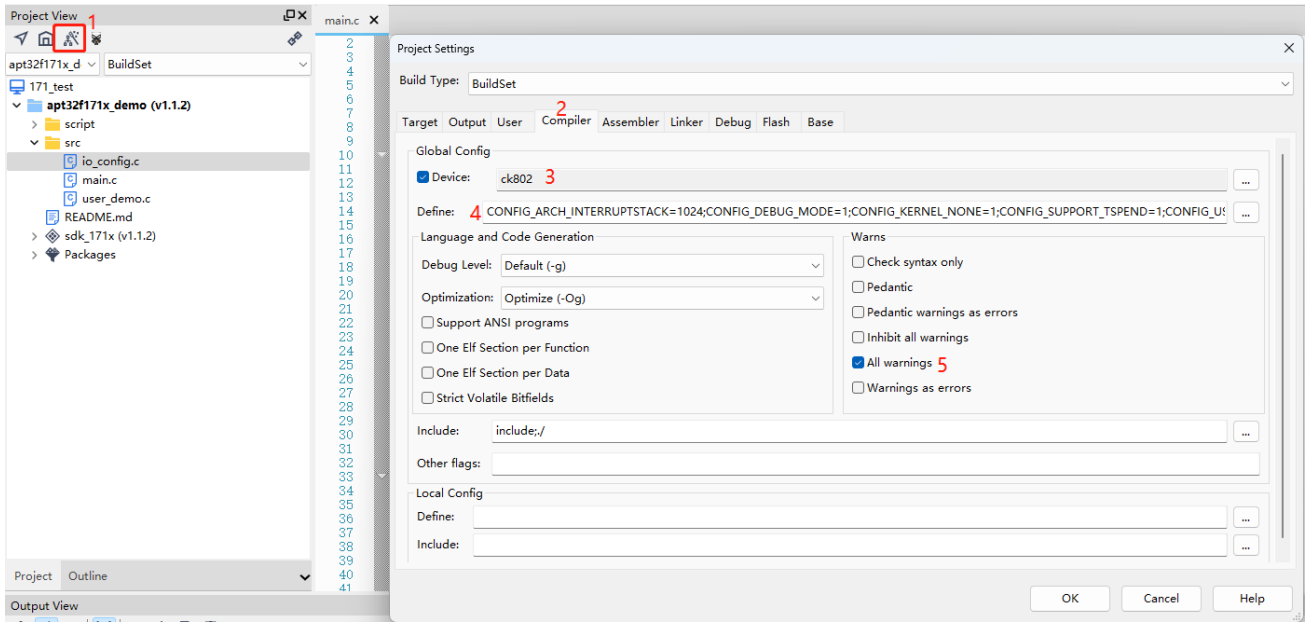


Figure 7 工程配置

2. 配置选项Output

- 标注6，7选择需要的输出文件

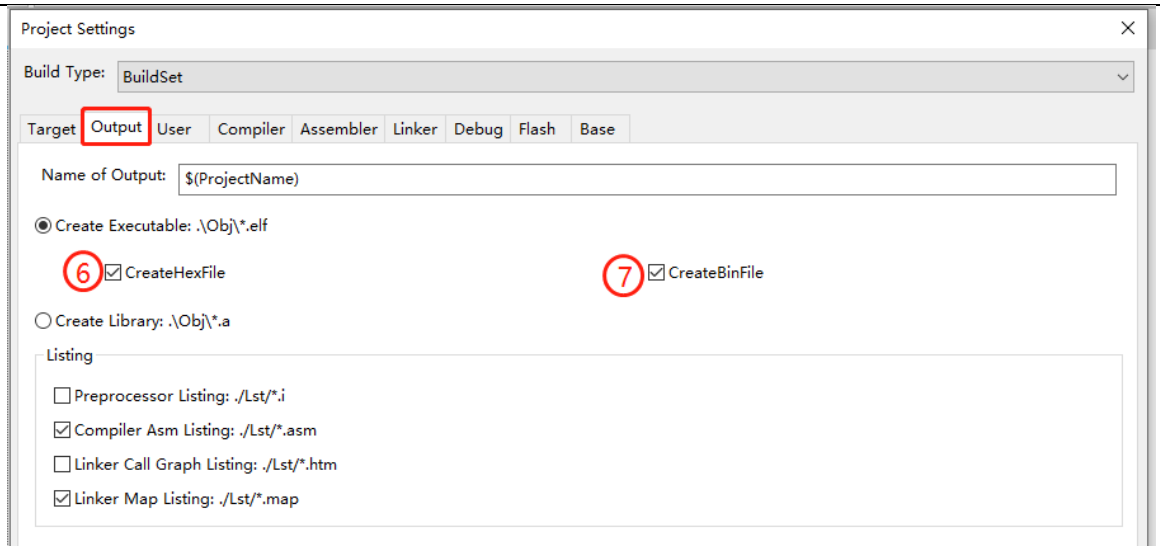


Figure 8 工程配置

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

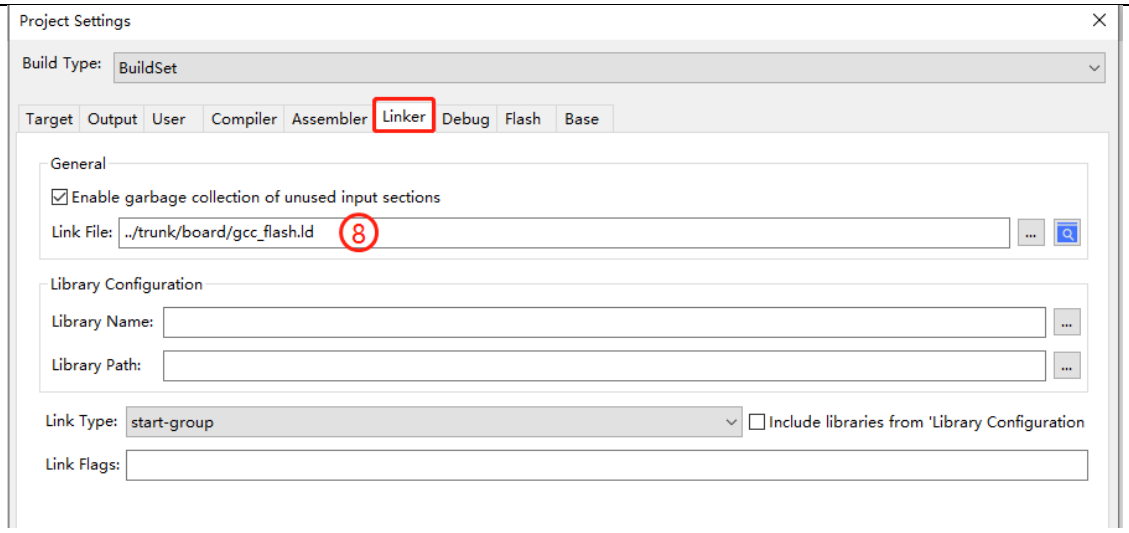
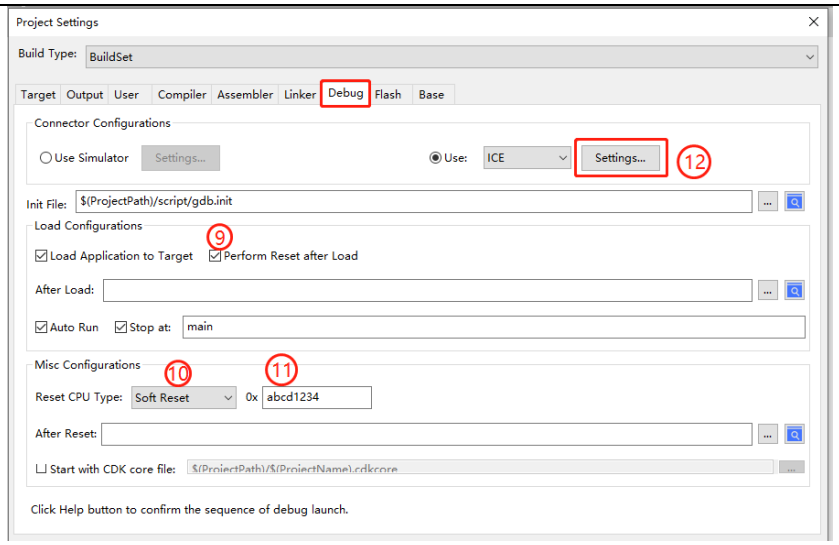


Figure 9 工程配置

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



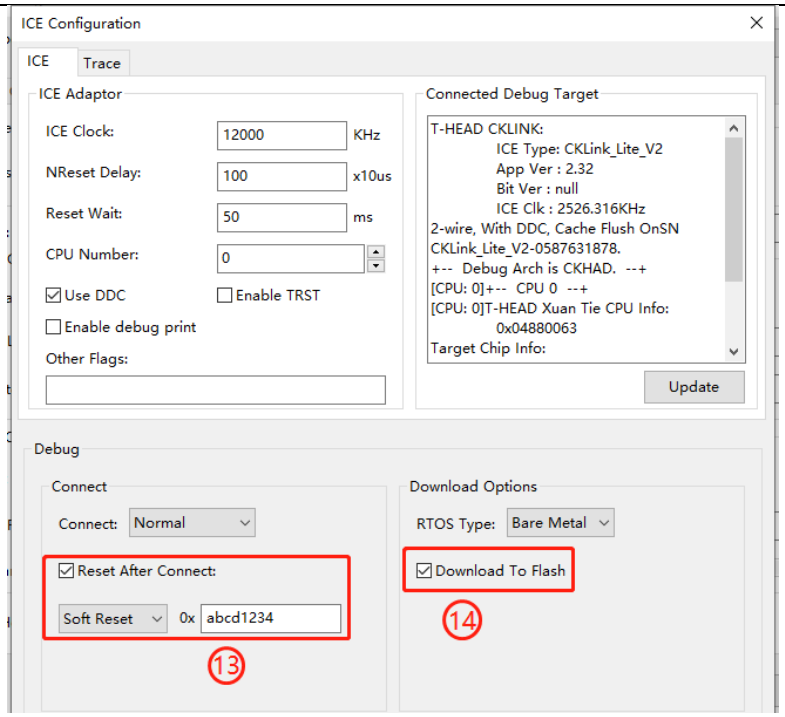


Figure 10 工程配置

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

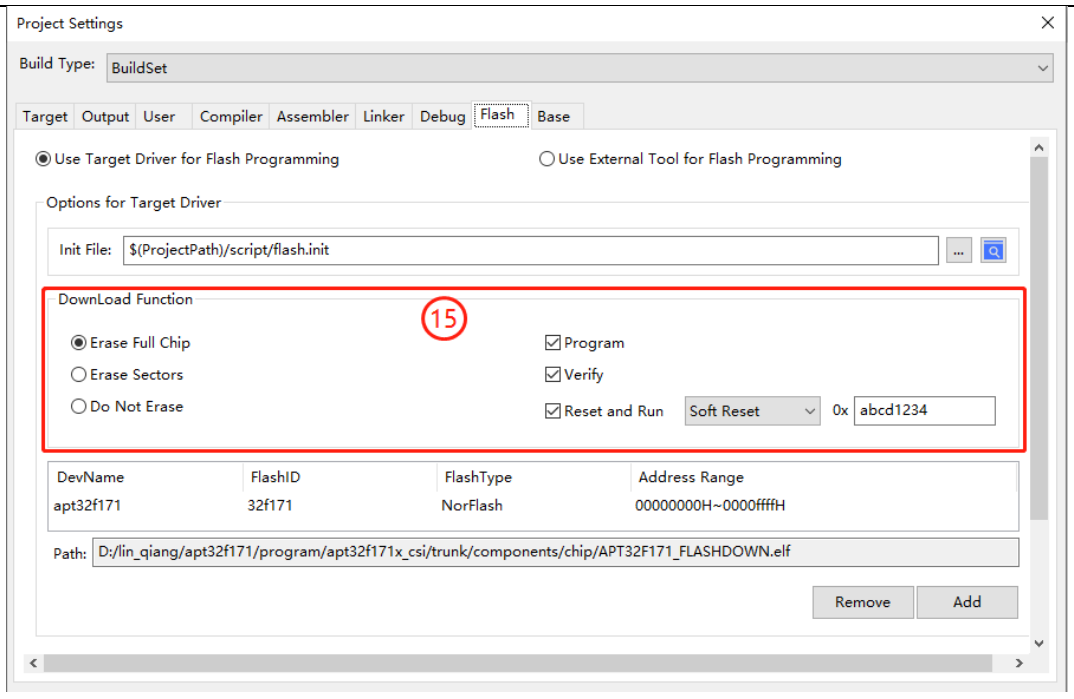


Figure 11 工程配置

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);
//-------------------------------------------------------------


int main()
{
    system_init();
    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。

Table1 system_init函数说明

函数	说明	位置
system_init()	__attribute__((weak)) void system_init(void) {	system.c

```
    //config system clk, close interrupt
    CK_CPU_DISALLNORMALIRQ;          //disable total interrupt
    csi_iwdt_close();                //close iwdt
    csi_sysclk_config(tClkConfig);   //sysclk config
    csi_get_sclk_freq();             //get system clk
    csi_get_pclk_freq();             //get pclk
    csi_tick_init();                 //init system ticks (coret)
    CK_CPU_ENALLNORMALIRQ;           //enable total interrupt
    //user add init
}
```

其中，csi_sysclk_config(tClkConfig)函数包含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。

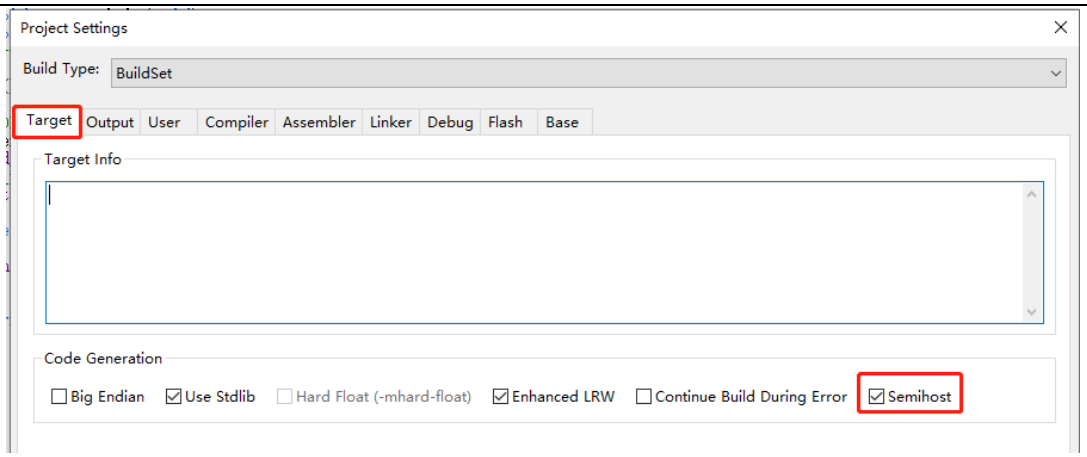
```
/// system clock configuration parameters to define source, source freq(if selectable), sdiv and pdiv
csi_clk_config_t tClkConfig =
    {SRC_HFOSC, HFOSC_48M_VALUE, SCLK_DIV1, PCLK_DIV1, 48000000, 48000000};
    //{SRC_EMOSC, EMOSC_VALUE, SCLK_DIV1, PCLK_DIV2, 5556000, 5556000};
    //{SRC_IMOSC, 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，不需要连接外部的硬件串口。使用时需要勾选Semihost配置，并且打印的时候使用printf，例如printf("hello world!\n")，调试时打开View->debugger pane窗口，然后在output页面查看调试信息



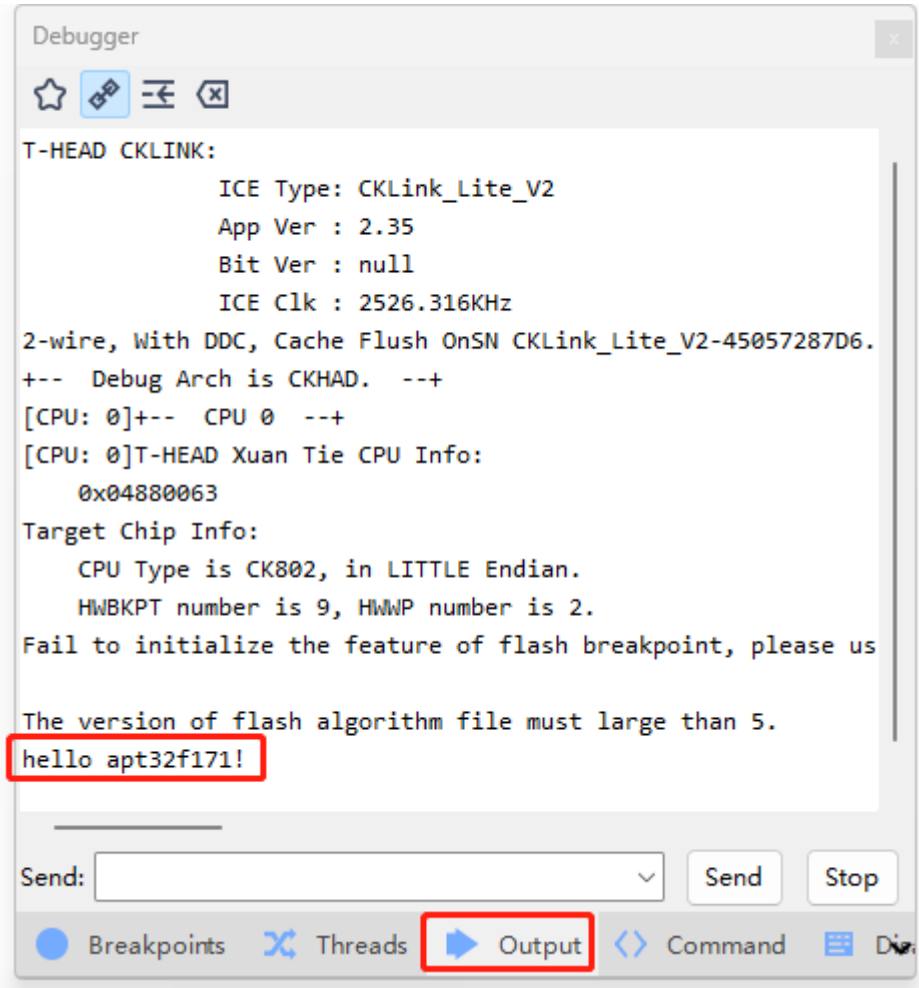


Figure 14 勾选 Semihost 并查看打印信息

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");

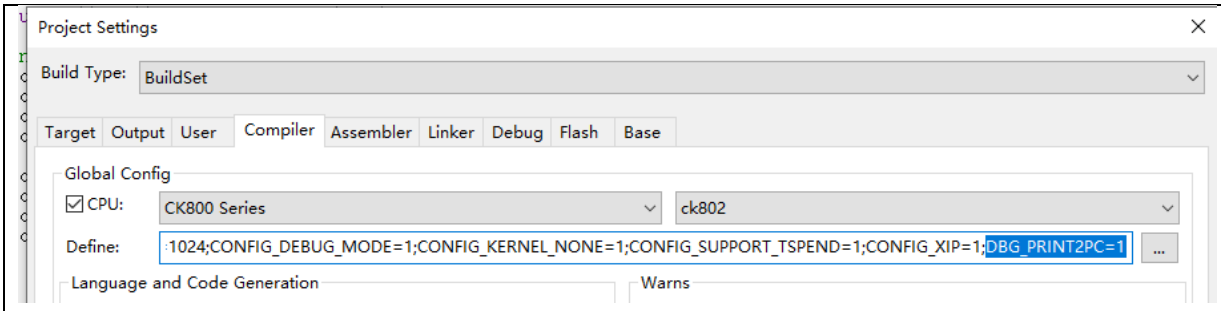


Figure 15 DBG_PRINT2PC宏

Table2 board_init函数说明

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BT0。

```
void BT0IntHandler(void)
{
#if BT0_INT_HANDLE_EN
    // ISR content ...
    bt_irqhandler(BT0);
#endif
}
```

```
#define BT0_INT_HANDLE_EN              1       //BT0
#define BT1_INT_HANDLE_EN              1       //BT1
#define BT2_INT_HANDLE_EN              1       //BT0
#define BT3_INT_HANDLE_EN              1       //BT1
```

Figure 16 中断处理函数举例

在上述例子中：

1、 需要先打开BT0_INT_HANDLE_EN宏定义，代码中所有中断的宏定义默认都是打开的，用户可在board_config.h中进行配置。

2、 bt_irqhandler(BT0)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体进行重写，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。

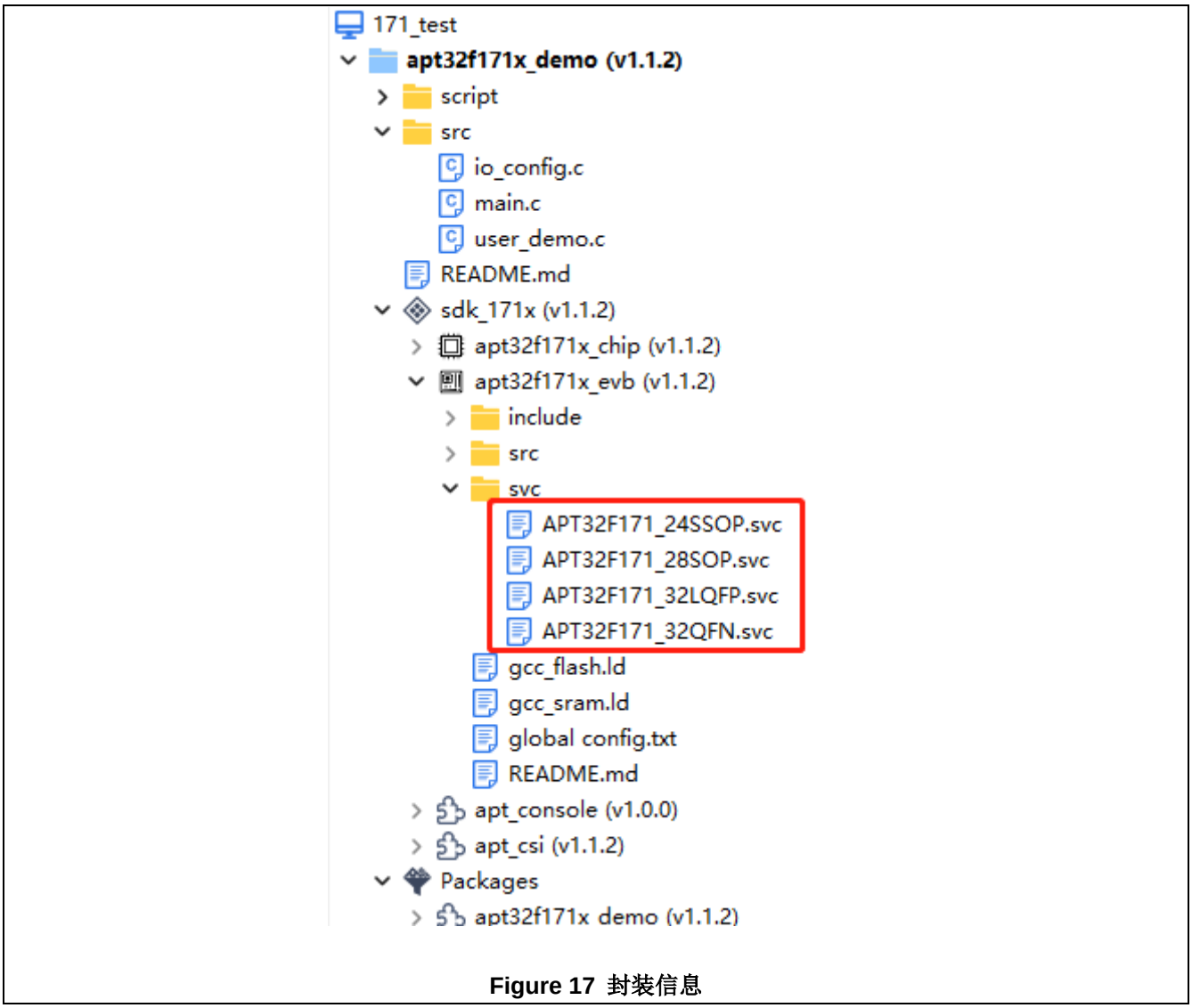


Figure 17 封装信息

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. 双击图中红色1标注地方：apt32f171x_evb->svc下->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

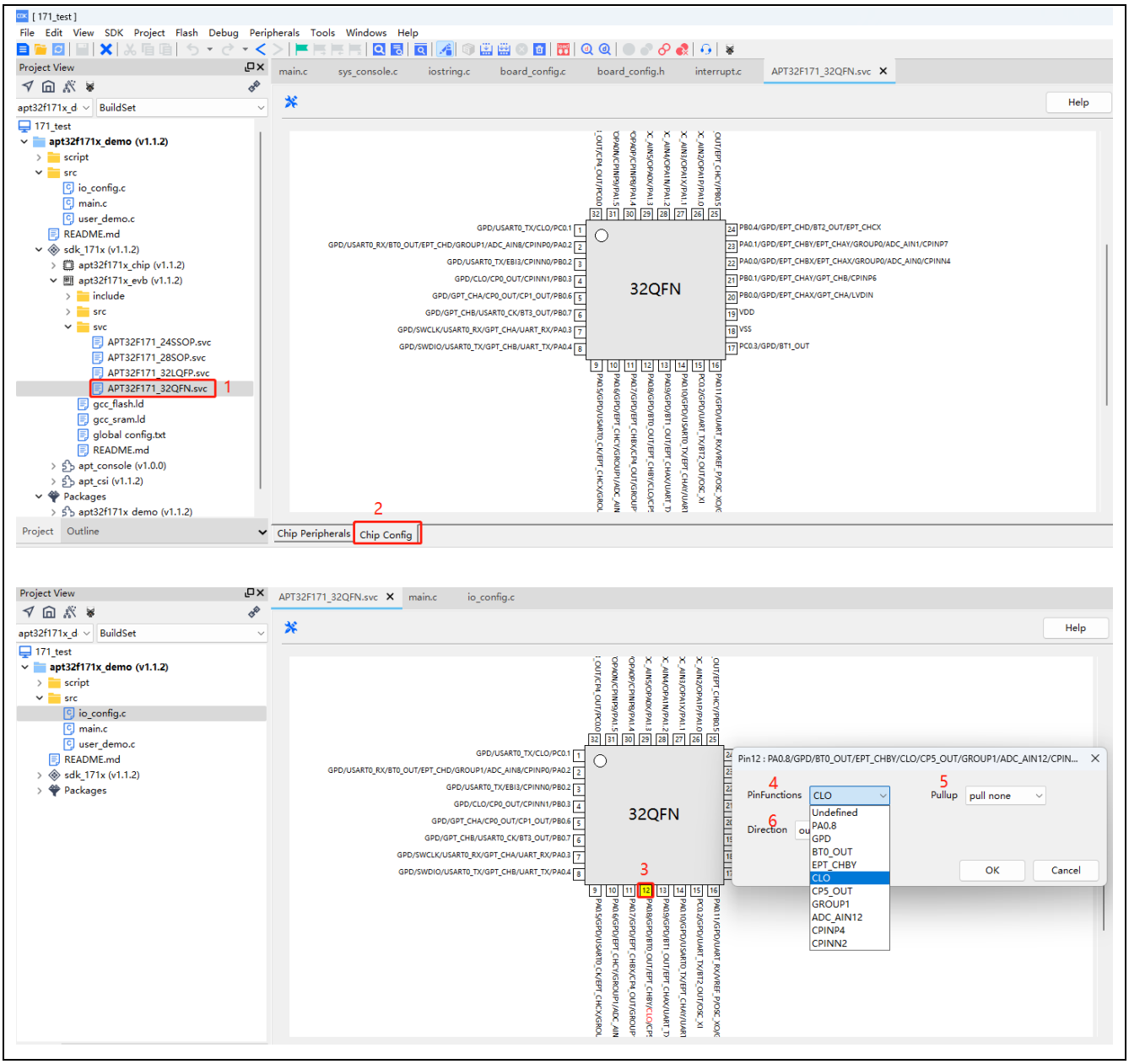
上拉/下拉/禁止上下拉功能

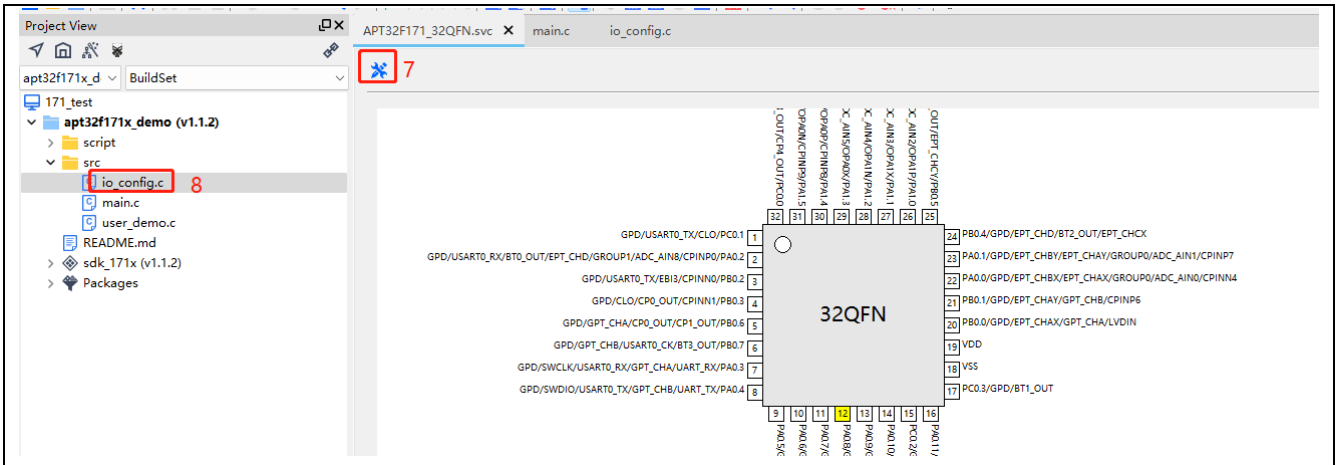
3. 点击红色标注6 设置输入还是输出，最后点击OK，完成配置。

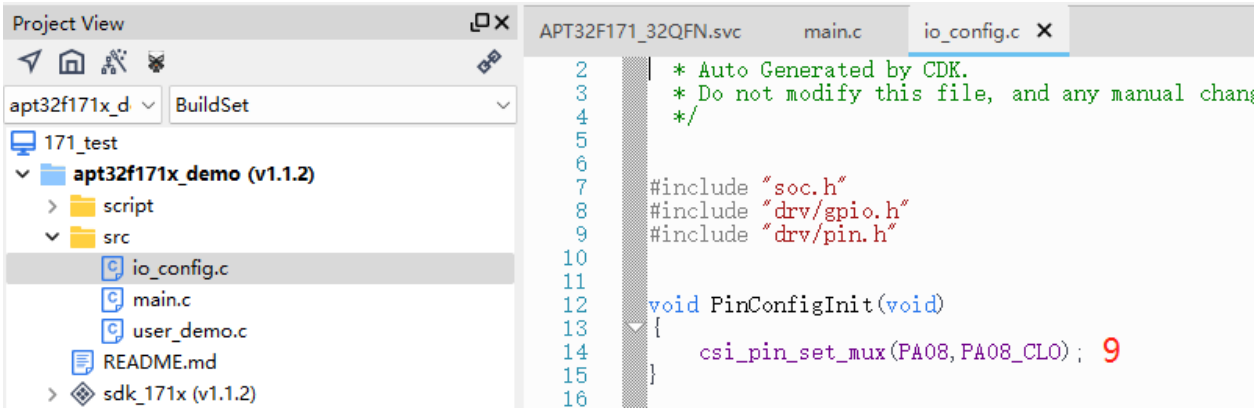
4. 再点击图中红色标注7的蓝色图标，在弹出对话框中点yes选项，将在工程根目录src文件夹下生成io_config.c文件。配置完成后，对应管脚的颜色将会变成黄色，对应选择功能会变成红色。

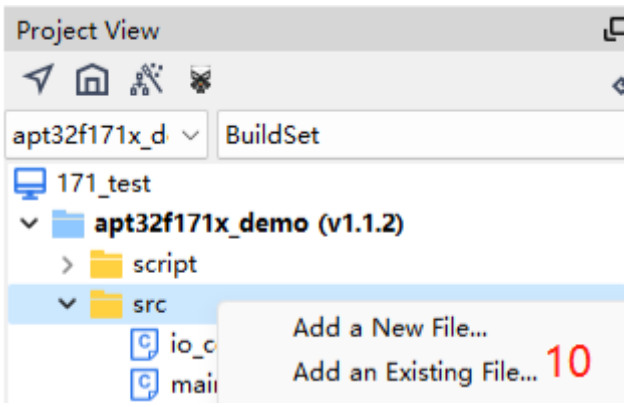
5. io_config.c中生成两个函数PinConfigInit和__ChipInitHandler函数。配置语句为图中红色标注9，配置语句在PinConfigInit函数中，用户初始化时直接调用即可。

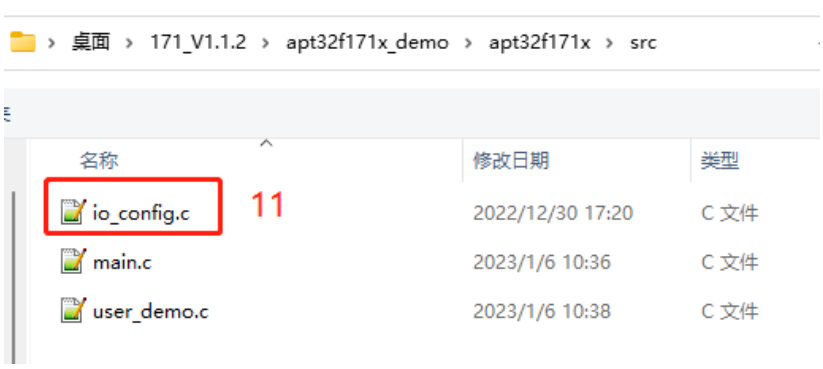
6. 编译时候如果提示找不到对应参考函数，则需要添加工程目录下的io_config.c文件(添加一次即可)，本示例中，该文件位于：$PROJ$\apt32f171x_demo\apt32f171x\src，详细操作参考标注10，11，12




Project View
APT32F171_32QFN.svc
main.c
io_config.c
7
apt32f171x_d
BuildSet
171_test
apt32f171x_demo (v1.1.2)
script
src
io_config.c
8
main.c
user_demo.c
README.md
sdk_171x (v1.1.2)
Packages
32QFN


Project View
APT32F171_32QFN.svc
main.c
io_config.c
apt32f171x_d
BuildSet
171_test
apt32f171x_demo (v1.1.2)
script
src
io_config.c
main.c
user_demo.c
README.md
sdk_171x (v1.1.2)
* Auto Generated by CDK.
* Do not modify this file, and any manual chang
*/
#include "soc.h"
#include "drv/gpio.h"
#include "drv/pin.h"
void PinConfigInit(void)
{
csi_pin_set_mux(PA08,PA08_CLO);
9
}


Project View
apt32f171x_d
BuildSet
171_test
apt32f171x_demo (v1.1.2)
script
src
Add a New File...
Add an Existing File...
10


桌面 › 171_V1.1.2 › apt32f171x_demo › apt32f171x › src
名称
修改日期
类型
io_config.c
11
2022/12/30 17:20
C 文件
main.c
2023/1/6 10:36
C 文件
user_demo.c
2023/1/6 10:38
C 文件

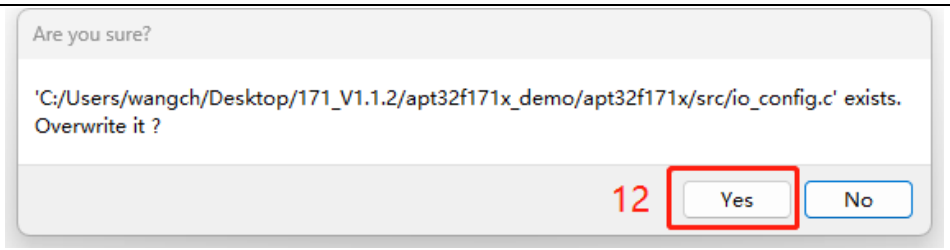


Figure 18 可视化配置步骤

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。

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。

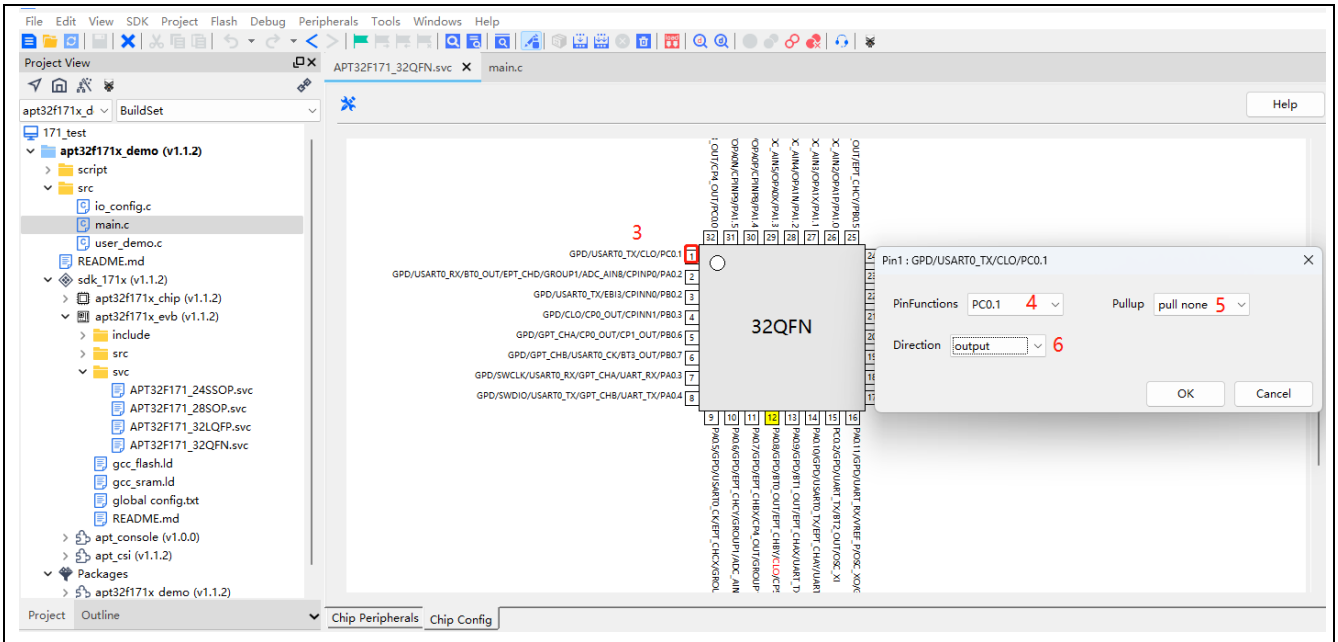


Figure 19 输入/ 输出配置，其余参阅 Figure 18

5.5 外设使用示例代码说明

外设使用的示例代码在apt32f171x_demo这个组件中。main函数中调用了user_demo()函数，该函数罗列了所有的示例代码，可以通过“打开、关闭”注释的方式调用相应的示例代码。示例代码的使用说明请参考第六章“Appendix 1”相关内容。

5.6 编译工程

点击“Build Project”即可实现工程的编译和链接。

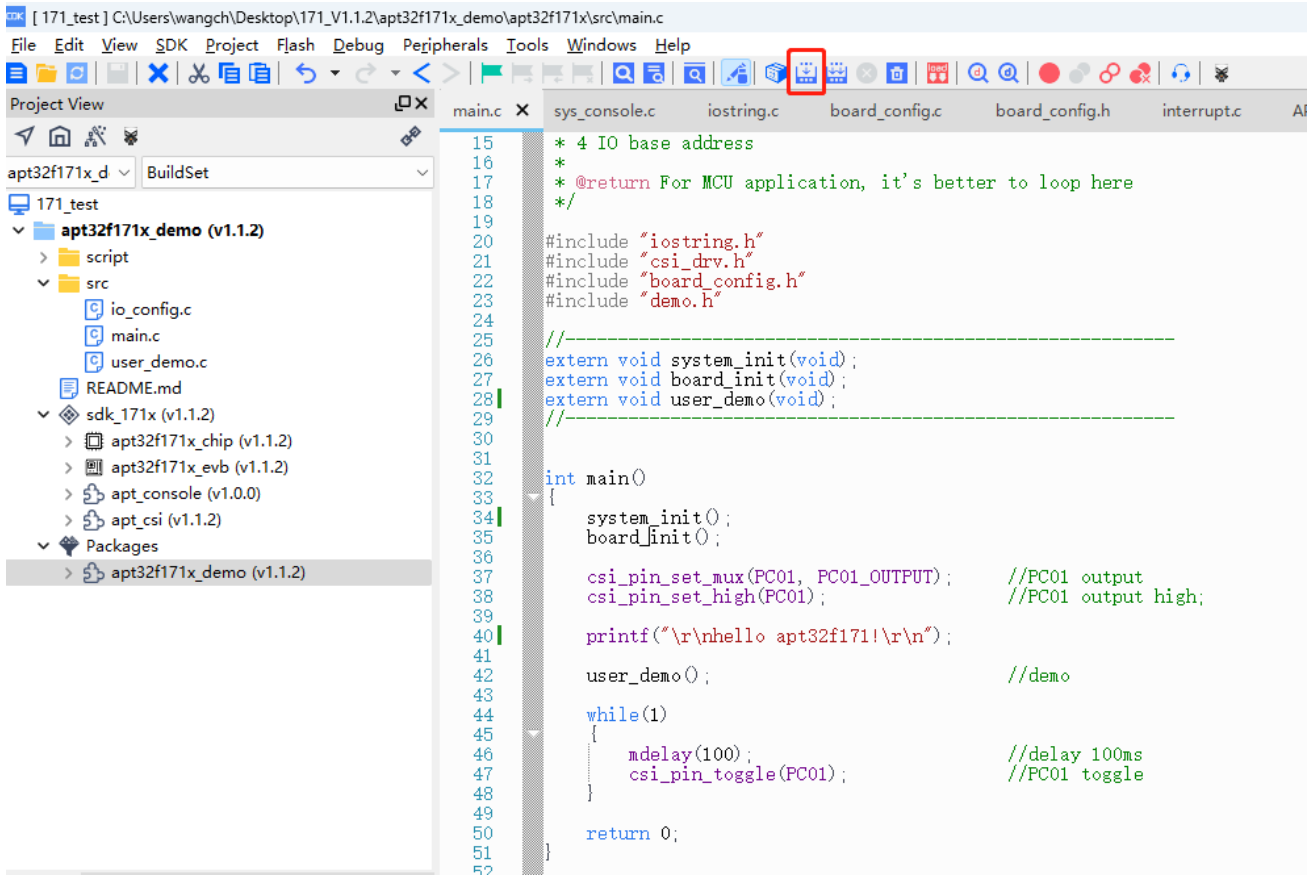


Figure 20 build

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。

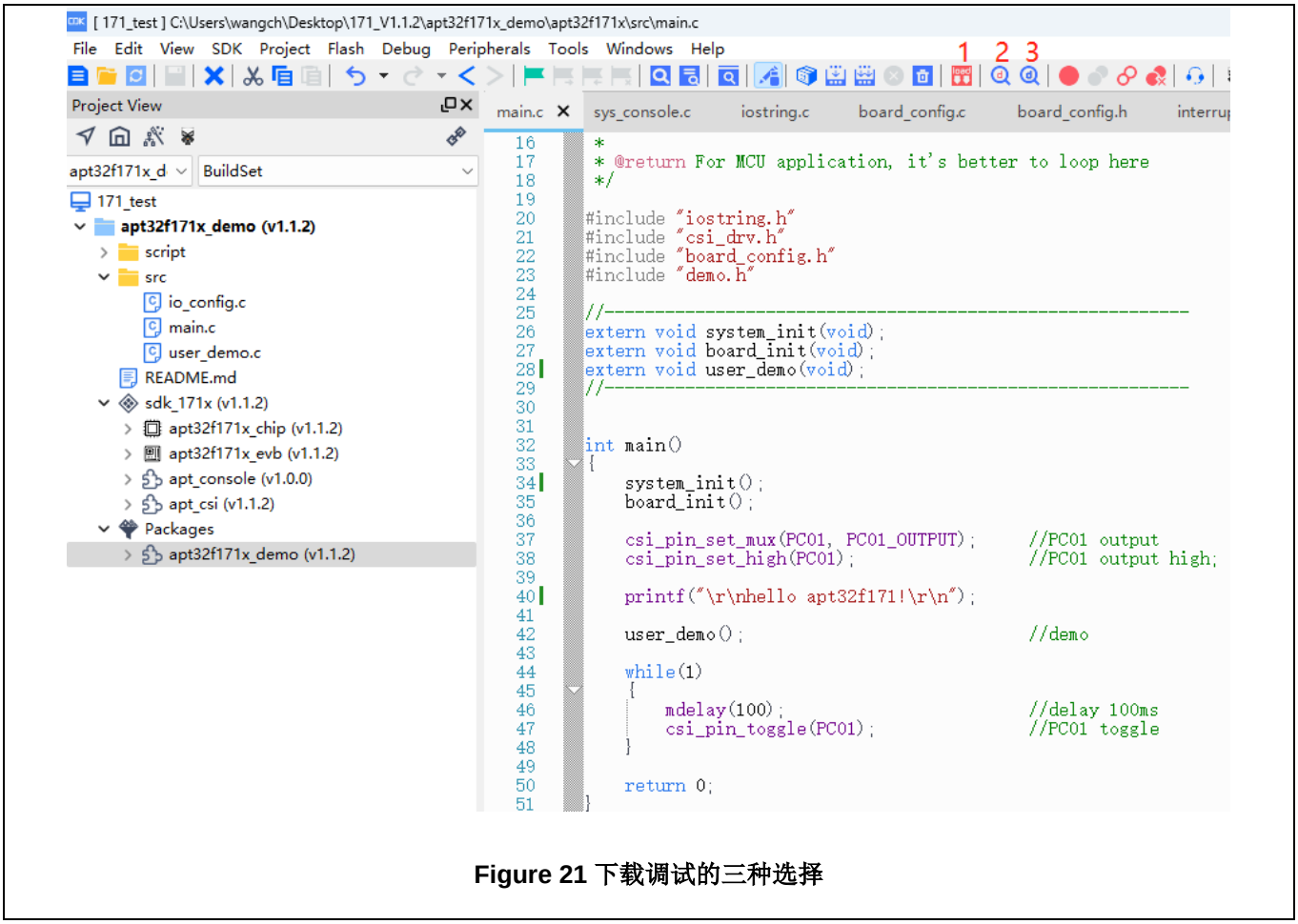


Figure 21 下载调试的三种选择

6. Appendix 1

6.1 GPIO

6.1.1 gpio_port_ouput_demo

1、 函数位置

gpio_demo.c

2、功能描述

多个GPIO输出功能示例。可配置多个GPIO为输出功能，并输出高电平或低电平。

3、参考用法

```
uint32_t wPinMask = PINMASK_PA00 | PINMASK_PA02;          //GPIOA0端口，PA00/PA02
csi_gpio_port_dir(GPIOA0, wPinMask, GPIO_DIR_OUTPUT);     //端口配置为输出
csi_gpio_port_set_high(GPIOA0, wPinMask);                 //输出高
csi_gpio_port_set_low(GPIOA0, wPinMask);                  //输出低
```

详见gpio_port_ouput_demo函数。

6.1.2 gpio_port_input_demo

1、函数位置

gpio_demo.c

2、功能描述

多个GPIO输入功能示例。可配置多个GPIO为输入功能，并使能上拉或者下拉。

3、 参考用法

```
uint32_t wPinMask = PINMASK_PA00 | PINMASK_PA02;          //GPIOA0端口，PA00/PA02
csi_gpio_port_dir(GPIOA0, wPinMask, GPIO_DIR_INPUT);      //端口配置为输入
csi_gpio_port_pull_mode(GPIOA0, wPinMask, GPIO_PULLNONE); //无上下拉
csi_gpio_port_pull_mode(GPIOA0, wPinMask, GPIO_PULLUP);   //上拉使能
csi_gpio_port_pull_mode(GPIOA0, wPinMask, GPIO_PULLDOWN); //下拉使能
```

详见gpio_port_input_demo函数。

6.1.3 gpio_port_irq_demo

1、函数位置

gpio_demo.c

2、 功能描述

多个GPIO中断功能示例。可配置多个GPIO为外部中断功能，并配置中断边沿。

3、参考用法

```
uint32_t wPinMask = PINMASK_PA00 | PINMASK_PA02;                    //GPIOA0端口，PA00/PA02
csi_gpio_port_dir(GPIOA0, wPinMask, GPIO_DIR_INPUT);                //端口配置为输入
csi_gpio_port_pull_mode(GPIOA0, wPinMask, GPIO_PULLUP);             //上拉使能
csi_gpio_port_irq_mode(GPIOA0,wPinMask,GPIO_IRQ_FALLING_EDGE);      //下降沿触发中断
csi_gpio_port_irq_enable(GPIOA0,wPinMask,ENABLE);                   //使能端口对应外部中断
```

详见gpio_port_irq_demo函数。

6.1.4 pin_output_demo

1、函数位置

pin_demo.c

2、功能描述

单个GPIO输出功能示例。可配置单个GPIO为输出功能，并输出高电平或低电平。

3、参考用法

```
csi_pin_set_mux(PA05,PA05_OUTPUT);          //PA05配置为输出
csi_pin_set_high(PA05);                     //输出高
csi_pin_set_low(PA05);                      //输出低
```

详见pin_output_demo函数。

6.1.5 pin_input_demo

1、函数位置

pin_demo.c

2、功能描述

单个GPIO输入功能示例。可配置单个GPIO为输入功能，并使能上拉或者下拉。

3、参考用法

```
csi_pin_set_mux(PA05,PA05_INPUT);           //PA05配置为输入
csi_pin_pull_mode(PA05,GPIO_PULLNONE);      //无上下拉
csi_pin_pull_mode(PA05,GPIO_PULLUP);        //上拉使能
csi_pin_pull_mode(PA05,GPIO_PULLDOWN);      //下拉使能
```

详见pin_input_demo函数。

6.1.6 pin_irq_demo

1、函数位置

pin_demo.c

2、功能描述

单个GPIO中断功能示例。可配置单个GPIO为外部中断功能，并配置中断边沿。

3、参考用法

```
csi_pin_set_mux(PB02, PB02_INPUT);                          //PB02配置为输入
csi_pin_pull_mode(PB02, GPIO_PULLUP);                       //上拉使能
csi_pin_irq_mode(PB02, EXI_GRP2, GPIO_IRQ_FALLING_EDGE);    //下降沿触发中断，选择中断组2
csi_pin_irq_enable(PB02, ENABLE);                           //PB02中断使能
csi_pin_vic_irq_enable(EXI_GRP2, ENABLE);                   //VIC中断使能，选择中断组2
```

详见pin_irq_demo函数。

6.1.7 pin_ioremap_demo

1、函数位置

pin_demo.c

2、功能描述

GPIO重定义功能示例。可配置GPIO引脚重定义功能。

关于GPIO重定义功能的更多信息，请查阅用户手册SYSCON章节。

3、参考用法

```
csi_pin_set_iomap(PA10, IOMAP0_USART0_TX);      //IOMAP GROUP0
csi_pin_set_iomap(PA011, IOMAP1_EPT_CHAX);      //IOMAP GROUP1
```

详见pin_ioremap_demo函数。

6.2 Clock

1、 函数位置

user_demo.c

2、功能描述

系统时钟输出功能示例。通过PB03引脚输出系统时钟，主要用于调试。

3、参考用法

```
csi_pin_set_mux(PB03, PB03_CLO);            //选择PB03为CLO功能
csi_clo_config(CLO_SYSCLK, CLO_DIV4);       //选择CLO时钟源及分频系数
```

6.3 Reliability

6.3.1 lvd_demo

1、函数位置

reliability_demo.c

2、功能描述

掉电监测功能示例。用于监控外部电源的电压值，在外部供电电压低于或高于设置值时，产生中断信号。

3、参考用法

```
csi_lvd_int_enable(LVD_INTF,LVD_39);          //VDD掉电到3.9V及以下，触发LVD中断
```

详见lvd_demo函数。

6.3.2 lvr_demo

1、函数位置

reliability_demo.c

2、功能描述

掉电复位功能示例。用于监控外部电源的电压值，在外部供电电压低于设置值时，产生芯片复位信号。

3、参考用法

```
csi_lvr_enable(LVR_28);                  //VDD掉电到2.8V及以下，芯片复位
```

详见lvr_demo函数。

6.3.3 memorycheck_demo

1、函数位置

reliability_demo.c

2、功能描述

内存可靠性监测功能示例。通过硬件校验电路对存储单元的数据写入或读取进行校验，校验合格后，数据才会被写入系统总线，若校验失败，控制器会根据设置进行重试，当重试计数达到预设值时，系统将会强制复位。

3、参考用法

```
csi_flashcheck_set_times(10);        //开启flash check功能，检查错误次数上限10
csi_flashcheck_rst();                //错误到达上限，芯片复位
csi_sramcheck_set_times( 8);         //开启sram check功能，检查错误次数上限8
csi_sramcheck_rst();                 //错误到达上限，芯片复位
```

详见memorycheck_demo函数。

6.3.4 emcm_demo

1、函数位置

reliability_demo.c

2、功能描述

外部时钟可靠性监测功能示例。当外部时钟失效时，可复位芯片或切换到内部时钟运行。

3、参考用法

```
csi_pin_set_mux(PC02, PC02_OSC_XI);
csi_pin_set_mux(PA011, PA011_OSC_XO);
csi_emosc_enable(16000000);              //使能外部晶振驱动电路，输入频率参数，以调整内部增益
csi_emcm_2_imosc_int();                  //一旦检测到外部晶振失常，系统时钟切到IMOSC，并触发中断
csi_emcm_rst();                          //一旦检测到外部晶振失常，系统复位
```

详见emcm_demo函数。

6.4 IWDT

6.4.1 iwdt_normal_demo

1、函数位置

iwdt_demo.c

2、功能描述

独立看门狗定时喂狗功能示例。如果超时没有喂狗，产生系统复位信号。

3、参考用法

```
csi_iwdt_init(IWDT_TO_1024);        //初始化看门狗，溢出时间为1024ms(系统复位时间)
csi_iwdt_open();                    //打开看门狗
csi_iwdt_feed();                    //喂狗
```

详见iwdt_normal_demo函数。

6.4.2 iwdt_irq_demo

1、函数位置

iwdt_demo.c

2、功能描述

独立看门狗报警功能示例。当计数值达到报警设置值时，会产生一个报警中断信号，提醒用户及时喂狗，防止芯片复位。

3、参考用法

```
csi_iwdt_init(IWDT_TO_1024);                    //初始化看门狗，溢出时间为1024ms(系统复位时间)
csi_iwdt_irq_enable(IWDT_ALARMTO_2_8, ENABLE);  //使能看门狗报警中断，报警时间为2/8溢出时间
csi_iwdt_open();                                //打开看门狗
```

详见iwdt_irq_demo函数。

6.5 WWDT

1、函数位置

wwdt_demo.c

2、功能描述

窗口看门狗功能示例。用于监测当前程序运行状况。

2、 参考用法

```
csi_wwdt_init(80);                      //设置timeout时间为80ms 时间设置过大会返回错误
csi_wwdt_debug_enable(ENABLE);          //可以配置在debug模式下，wwdt是否继续计时
csi_wwdt_set_window_time(40);           //设置窗口值为40ms
csi_wwdt_open();                        //WWDT一旦使能，软件将不能停止
```

详见wwdt_demo函数。

6.6 IFC

6.6.1 ifc_read_demo

1、函数位置

ifc_demo.c

2、功能描述

flash读操作功能示例。操作单位为word。

3、参考用法

```
csi_ifc_read(IFC,0x00000000, wReadBuf, 2);   //从0x0地址读取2个word数据
```

详见ifc_read_demo函数。

6.6.2 ifc_dflash_page_program_demo

1、函数位置

ifc_demo.c

2、功能描述

dflash页编程功能示例。起始地址必须word对齐，数据类型为word，数据必须在一个page内，不允许跨页。

3、参考用法

```
csi_ifc_dflash_page_program(IFC, 0x10000000,wWriteData, 5); //从0x10000000地址写入5个word数据
```

详见ifc_dflash_page_program_demo函数。

6.6.3 ifc_dflash_page_parallel_program_demo

1、函数位置

ifc_demo.c

2、功能描述

dflash并行模式页编程功能示例。只有dflash支持并行模式，在并行模式下，dflash擦写的同时，CPU仍然可以从pflash取址运行。起始地址必须word对齐，数据类型为word，数据必须在一个page内，不允许跨页。

3、参考用法

```
csi_ifc_dflash_paramode_enable(IFC, ENABLE);                    //使能dflash并行模式
csi_ifc_dflash_page_program(IFC, 0x10000000,wWriteData, 5);   //从0x10000000地址写入5个word数据
```

详见ifc_dflash_page_parallel_program_demo函数。

6.6.4 ifc_pflash_page_program_demo

1、函数位置

ifc_demo.c

2、功能描述

pflash页编程功能示例。起始地址必须word对齐，数据类型为word，数据必须在一个page内，不允许跨页。

3、参考用法

```
csi_ifc_pflash_page_program(IFC, 0x0000F000,wWriteData, 5);    //从0x0000F000地址写入5个word数据
```

详见ifc_pflash_page_program_demo函数。

6.6.5 ifc_page_erase_demo

1、函数位置

ifc_demo.c

2、功能描述

flash页擦除功能示例。无论dflash还是pflash，编程操作都自带erase步骤，不需要额外调用该函数，否则会影响flash寿命。

3、参考用法

```
csi_ifc_page_erase(IFC,0x10000000);            //擦除dflash第一个page
```

详见ifc_page_erase_demo函数。

6.6.6 ifc_program_demo

1、函数位置

ifc_demo.c

2、功能描述

flash写操作功能示例。起始地址必须word对齐，数据类型为word，支持跨页。

3、参考用法

```
csi_ifc_program(IFC, 0xfef8, wWriteData, 3);          //从0xfe78地址（PFLASH)开始，写3个word
csi_ifc_program(IFC, 0x10000078, wWriteData, 5); //从0x10000078地址（DFLASH)开始，写5个word
```

详见ifc_program_demo函数。

6.7 PM

6.7.1 lp_exi_wakeup_demo

1、函数位置

lowpower_demo.c

2、功能描述

低功耗唤醒功能示例。系统进入低功耗模式（SLEEP或者DEEP_SLEEP），通过外部中断引脚唤醒。

3、 参考用法

```
csi_pin_set_mux(PA05,PA05_INPUT);                                  //PA05 输入使能
csi_pin_pull_mode(PA05, GPIO_PULLUP);                              //PA05 上拉使能
csi_pin_irq_mode(PA05,EXI_GRP5, GPIO_IRQ_FALLING_EDGE);    //PA05 下降沿产生中断，选择中断组5
csi_pin_irq_enable(PA05, ENABLE);                                  //PA05 gpio中断使能
csi_pin_vic_irq_enable(EXI_GRP5,ENABLE);                           //PA05 vic中断使能
......
//配置不同低功耗模式下的唤醒源
......
csi_pm_enter_sleep(_LOW_POWER_MODE_);                              //进入低功耗模式
```

详见lp_exi_wakeup_demo函数

6.7.2 lp_iwdt_wakeup_demo

1、函数位置

lowpower_demo.c

2、功能描述

低功耗唤醒功能示例。系统进入低功耗模式（DEEP_SLEEP），通过iwdt定时唤醒。

4、 参考用法

```
csp_clk_pm_enable(SYSCON, ISOSC_STP, ENABLE);     //使能ISOSC在DEEP-SLEEP模式下工作
csi_iwdt_init(IWDT_TO_4096);                       //初始化看门狗，溢出时间为4096ms(系统复位时间)
csi_iwdt_irq_enable(IWDT_ALARMTO_2_8, ENABLE);     //使能看门狗报警中断，报警时间为2/8溢出时间
csi_iwdt_open();                                   //打开看门狗，记得在syscon_irqhandler函数中（在reliability.c中）喂狗
```

```
csi_pm_config_wakeup_source(WKUP_IWDT, ENABLE);   //设置唤醒源
csi_pm_enter_sleep(_LOW_POWER_MODE_);              /进入低功耗模式
```

详见lp_iwdt_wakeup_demo函数

6.8 BT

6.8.1 bt_timer_demo

1、函数位置

bt_demo.c

2、功能描述

BT0基本定时功能示例。

3、参考用法

```
csi_bt_timer_init(BT0, 5000);     // BT0定时5000us
csi_bt_start(BT0);                //启动定时器
```

详见bt_timer_demo函数。

6.8.2 bt_pwm_demo

1、函数位置

bt_demo.c

2、功能描述

BT0 PWM输出功能示例。可配置PWM周期和占空比，占空比只能为整数，如果有更细致的配置要求可以通过csi_bt_prdr_cmp_updata()实现。

3、参考用法

```
csi_pin_set_mux(PA02, PA02_BT0_OUT);            //PA02作为BT0 PWM输出引脚
......
//初始化tPwmCfg，设置PWM电平参数，周期和占空比
......
csi_bt_pwm_init(BT0, &tPwmCfg);                 //初始化BT0 PWM输出
csi_bt_start(BT0);                              //启动BT0
csi_bt_pwm_updata(BT0, 1000, 20);               //修改PWM参数为周期1KHz，占空比为20%
csi_bt_pwm_duty_cycle_updata(BT0, 40);          //修改PWM占空比为40%
csi_bt_prdr_cmp_updata(BT0, 10000, 5000);       //修改PWM周期为10000，比较值为5000
```

详见bt_pwm_demo函数。

6.8.3 bt_sync_start_demo

1、函数位置

bt_demo.c

2、功能描述

BT0触发启动功能示例。可用外部中断EXI事件通过ETCB模块触发BT0 Sync Port0，即BT0启动计数。

3、参考用法

```
//PA07配置为外部中断功能，选择中断组7
......
csi_exi_set_evtrg(EXI_TRGOUT3, TRGSRC_EXI7, 3);              //EXI7触发EXI_TRGOUT3事件
csi_bt_timer_init(BT0,5000);                                  //BT0定时5ms
csi_bt_set_sync(BT0, BT_TRG_SYNCIN0, BT_TRG_ONCE, BT_AREARM_DIS);
//外部触发BT0启动(SYNCIN0),禁止自动REARM
......
//ETCB配置，设置目标事件(EXI_TRGOUT3)和源事件(BT0 SYNCIN0)：EXI_TRGOUT3 -> ETCB -> BT0 SYNCIN0
......
csi_etb_init();                                               //ETCB初始化
```

详见bt_sync_start_demo函数。

6.8.4 bt_sync_stop_demo

1、函数位置

bt_demo.c

2、功能描述

BT0触发停止功能示例。可用外部中断EXI事件通过ETCB模块触发BT0 Sync Port1，即BT0停止计数。

3、参考用法

```
//PB01配置为外部中断功能，选择中断组17
......
csi_exi_set_evtrg(EXI_TRGOUT5, TRGSRC_EXI17, 0);                 //EXI17(PB01)触发EXI_TRGOUT5
csi_bt_timer_init(BT0,2000);                                      //BT0定时2ms
csi_bt_set_sync(BT0, BT_TRG_SYNCIN1, BT_TRG_ONCE, BT_AREARM_DIS);    //外部触发停止BT0(SYNCIN1)
csi_bt_start(BT0);                                                //启动定时器
......
//ETCB配置，设置目标事件(EXI_TRGOUT5)和源事件(BT0 SYNCIN1)：EXI_TRGOUT5 -> ETCB -> BT0 SYNCIN1
......
csi_etb_init(); //ETCB初始化
```

详见bt_sync_stop_demo函数。

6.8.5 bt_sync_count_demo

1、函数位置

bt_demo.c

2、功能描述

BT0触发计数值加1功能示例。可用外部中断EXI事件通过ETCB模块触发BT0 Sync Port2，BT0计数加1。

3、参考用法

```
//PB01配置为外部中断功能，选择中断组18
......
csi_exi_set_evtrg(EXI_TRGOUT4, TRGSRC_EXI18, 0);                    //EXI18(PB01)触发EXI_TRGOUT4
csi_bt_timer_init(BT0,20);                                           //BT0定时
csi_bt_set_sync(BT0, BT_TRG_SYNCIN2, BT_TRG_CONTINU, BT_AREARM_DIS);//外部触发BT0计数(SYNCIN2),
SYNCIN2不支持一次性触发，硬件自动REARM无意义，即最后一个参数无意义
csi_bt_start(BT0);                                                   //启动定时器
......
//ETCB配置，设置目标事件(EXI_TRGOUT4)和源事件(BT0 SYNCIN2)：EXI_TRGOUT4 -> ETCB -> BT0 SYNCIN2
......
csi_etb_init(); //ETCB初始化
```

详见bt_sync_count_demo函数。

6.8.6 bt_trg_out_demo

1、函数位置

bt_demo.c

2、功能描述

BT0触发输出功能示例。可用BT0 PEND事件通过ETCB模块触发BT1 Sync Port0，即BT1启动。

3、参考用法

```
csi_bt_timer_init(BT0,100000);                                       //BT0定时100ms，默认连续计数模式
csi_bt_set_evtrg(BT0, BT_TRGOUT, BT_TRGSRC_PEND);                    //BT0 PEND事件触发输出
csi_bt_start(BT0);                                                   //启动BT0定时器
csi_bt_timer_init(BT1,50000);                                        //BT1定时50ms,默认连续计数模式
csi_bt_count_mode(BT1, BT_CNT_ONCE);                                 //单次计数模式
csi_bt_set_sync(BT1, BT_TRG_SYNCIN0, BT_TRG_CONTINU, BT_AREARM_DIS);//外部触发BT1启动(SYNCIN0)
......
//ETCB配置，设置目标事件(BT0 PEND)和源事件(BT1 SYNCIN0)：BT0 PEND -> ETCB -> BT1 SYNCIN0
......
csi_etb_init(); //ETCB初始化
```

详见bt_trg_out_demo函数。

6.8.7 bt_sync1_arearm_sync0_demo

1、函数位置

bt_demo.c

2、功能描述

BT0触发事件自动REARM功能示例。可用Sync1事件自动REARM Sync0事件。

3、 参考用法

```
//PA07配置为外部中断功能，选择中断组7
......
csi_exi_set_evtrg(EXI_TRGOUT3, TRGSRC_EXI7, 0);          //EXI7  触发EXI_TRGOUT3
csi_bt_timer_init(BT0,5000);                                 //BT0 5ms定时
csi_bt_set_sync(BT0, BT_TRG_SYNCIN0, BT_TRG_ONCE, BT_AREARM_SYNC1);
//外部触发BT0启动(SYNCIN0),使用SYNCIN1事件自动REARM
//ETCB配置，设置目标事件(EXI_TRGOUT3)和源事件(BT0 SYNCIN0)：EXI_TRGOUT3 -> ETCB -> BT0 SYNCIN0
......
csi_etb_init(); //ETCB初始化
......
//PB01配置为外部中断功能，选择中断组17
......
csi_exi_set_evtrg(EXI_TRGOUT5, TRGSRC_EXI17, 0);
//EXI17(PB01)触发EXI_TRGOUT5(PB01用EXI17触发输出)
csi_bt_set_sync(BT0, BT_TRG_SYNCIN1, BT_TRG_ONCE, BT_AREARM_PEND);
//外部触发停止BT0(SYNCIN1)，使用PEND事件自动REARM
......
//ETCB配置，设置目标事件(EXI_TRGOUT5)和源事件(BT0 SYNCIN1)：EXI_TRGOUT5 -> ETCB -> BT0 SYNCIN1
......
```

详见bt_sync1_arearm_sync0_demo函数。

6.8.8 bt_sync0_arearm_sync1_demo

1、函数位置

bt_demo.c

2、功能描述

BT0触发事件自动REARM功能示例。可用Sync0事件自动REARM Sync1事件。

4、 参考用法

```
//PB01配置为外部中断功能，选择中断组17
......
csi_exi_set_evtrg(EXI_TRGOUT5, TRGSRC_EXI17, 0);//EXI17(PB01)触发EXI_TRGOUT5(PB01用EXI17触发输出)
csi_bt_timer_init(BT0,5000);                                 //BT0定时
csi_bt_set_sync(BT0, BT_TRG_SYNCIN1, BT_TRG_ONCE, BT_AREARM_SYNC0); //外部触发停止BT0(SYNCIN1),
使用SYNCIN0事件自动REARM
csi_bt_start(BT0);                                           //启动定时器
//ETCB配置，设置目标事件(EXI_TRGOUT5)和源事件(BT0 SYNCIN1)：EXI_TRGOUT5 -> ETCB -> BT0 SYNCIN1
......
csi_etb_init(); //ETCB初始化
......
//PA07配置为外部中断功能，选择中断组7
......
csi_exi_set_evtrg(EXI_TRGOUT3, TRGSRC_EXI7, 3); //EXI7  触发EXI_TRGOUT3
csi_bt_set_sync(BT0, BT_TRG_SYNCIN0, BT_TRG_ONCE, BT_AREARM_PEND);
//外部触发BT0启动(SYNCIN0),使用PEND事件自动REARM
```

```
.....
//ETCB配置，设置目标事件(EXI_TRGOUT3)和源事件(BT0 SYNCIN0)：EXI_TRGOUT3 -> ETCB -> BT0 SYNCIN0
```

详见bt_sync0_arearm_sync1_demo函数。

6.9 GPTA

6.9.1 gpta_timer_demo

1、函数位置

gpta_demo.c

2、功能描述

GPTA基本定时功能示例。

3、参考用法

```
csi_gpta_timer_init(GPTA0, 10000);          //初始化GPTA0，定时10000us
csi_gpta_start(GPTA0);                      //启动定时器
```

详见gpta_timer_demo函数。

6.9.2 gpta_capture_demo

1、函数位置

gpta_demo.c

2、功能描述

GPTA捕获功能示例。由外部中断EXI事件通过ETCB模块触发GPTA SYNCIN2，产生一次捕获。

3、参考用法

```
//PA01配置为外部中断功能，选择中断组1
.....
csi_exi_set_evtrg(EXI_TRGOUT1, TRGSRC_EXI1, 1);   //EXI1触发EXI_TRGOUT1
//ETCB配置，设置目标事件(EXI_TRGOUT1)和源事件(GPTA0_SYNCIN2)：EXI_TRGOUT1 -> ETCB ->
GPTA0_SYNCIN2
.....
csi_etb_init(); //ETCB初始化
.....
//初始化tPwmCfg，设置GPTA0的捕获参数
.....
csi_gpta_capture_init(GPTA0, &tPwmCfg);
csi_gpta_set_sync(GPTA0, GPTA_TRG_SYNCEN2, GPTA_TRG_CONTINU, GPTA_AUTO_REARM_ZRO);
//使能SYNCIN2外部触发
csi_gpta_start(GPTA0);                          //启动GPTA0
```

详见gpta_capture_demo函数。

6.9.3 gpta_pwm_demo

1、函数位置

gpta_demo.c

2、功能描述

GPTA波形输出功能示例。可配置PWM周期和占空比，占空比只能为整数，如果有更细致的配置要求可以通过csi_gpta_prdr_cmp_update ()实现。

3、参考用法

```
csi_pin_set_mux(PB00,PB00_GPT_CHA);        //PB00作为PWM输出引脚
......
//初始化tPwmCfg，设置PWM周期和占空比
......
csi_gpta_wave_init(GPTA0, &tPwmCfg);
......
//初始化tGptachannelCfg，设置PWM波形输出
......
csi_gpta_channel_config(GPTA0,&tGptachannelCfg,GPTA_CHANNEL_1);
csi_gpta_start(GPTA0);
csi_gpta_change_ch_duty(GPTA0,GPTA_COMPA, 20);                    //修改PWM1占空比为20%
csi_gpta_prdr_cmp_update(GPTA0,GPTA_COMPA,1200,800);              //修改PWM1周期为1200，比较值为800
```

详见gpta_pwm_demo函数。

6.9.4 gpta_soft_trgout_demo

1、函数位置

gpta_demo.c

2、功能描述

GPTA软件触发事件输出功能示例。可用GPTA软件触发BT1 Sync Port0，即BT1启动。

3、参考用法

```
csi_bt_start_sync(BT1, 10);
csi_bt_set_sync(BT1,BT_TRG_SYNCIN0, BT_TRG_ONCE, BT_AREARM_PEND);    //外部触发BT1启动(SYNCIN0),使用PEND事件自动REARM
......
//初始化tPwmCfg，设置PWM周期和占空比
......
csi_gpta_wave_init(GPTA0, &tPwmCfg);
//ETCB配置，设置目标事件(GPTA_TRGOUT0)和源事件(BT1 SYNCIN0)：GPTA_TRGOUT0 -> ETCB -> BT1 SYNCIN0
......
```

```
csi_etb_init();
.....
csi_gpta_evtrg_enable(GPTA0, GPTA_TRGOUT0, ENABLE);
csi_gpta_swf_trg(GPTA0, GPTA_TRGOUT0);
```

详见gpta_soft_trgout_demo函数。

6.9.5 gpta_pwm_syncin4_demo

1、函数位置

gpta_demo.c

2、功能描述

GPTA触发功能示例。可用BT0 PEND事件通过ETCB模块触发GPTA SYNCIN4，产生内部T1事件，控制PWM波形输出。

3、参考用法

```
csi_bt_timer_init(BT0, 1000);            //初始化BT0，定时1000us
csi_bt_start(BT0);                       //启动定时器
csi_bt_set_evtrg(BT0, 0, BT_TRGSRC_PEND);//BT0 PEND触发输出PEND事件
//ETCB配置，设置目标事件(BT0 PEND)和源事件(GPTA0 SYNCIN4)：BT0 PEND0 -> ETCB -> GPTA0 SYNCIN4
.....
csi_etb_init();
.....
//初始化tPwmCfg，设置PWM周期和占空比
.....
csi_gpta_wave_init(GPTA0, &tPwmCfg);
.....
//初始化tGptachannelCfg，设置PWM波形输出
.....
csi_gpta_channel_config(GPTA0,&tGptachannelCfg,GPTA_CHANNEL_1);
csi_gpta_set_sync(GPTA0, GPTA_TRG_SYNCEN4, GPTA_TRG_CONTINU, GPTA_AUTO_REARM_ZRO);
//使能SYNCIN4外部触发
csi_gpta_start(GPTA0);
```

详见gpta_pwm_syncin4_demo函数。

6.9.6 gpta_pwm_syncin0_demo

1、函数位置

gpta_demo.c

2、功能描述

GPTA触发功能示例。可用BT0 PEND事件通过ETCB模块触发GPTA SYNCIN0，触发计数器重置/分频器重置/Shadow寄存器更新。

3、参考用法

```
csi_bt_timer_init(BT0, 800);                //初始化BT0，定时800us
csi_bt_start(BT0);                          //启动定时器
csi_bt_set_evtrg(BT0, 0, BT_TRGSRC_PEND);//BT0 PEND触发输出PEND事件
//ETCB配置，设置目标事件(BT0 PEND)和源事件(GPTA0 SYNCIN0)：BT0 PEND0 -> ETCB -> GPTA0 SYNCIN0
......
csi_etb_init();
......
//初始化tPwmCfg，设置PWM周期和占空比
......
csi_gpta_wave_init(GPTA0, &tPwmCfg);
......
//初始化tGptachannelCfg，设置PWM波形输出
......
csi_gpta_channel_config(GPTA0,&tGptachannelCfg,GPTA_CHANNEL_1);
csi_gpta_set_sync(GPTA0, GPTA_TRG_SYNCEN0, GPTA_TRG_CONTINU, GPTA_AUTO_REARM_ZRO);
//使能SYNCIN0外部触发
csi_gpta_start(GPTA0);
```

详见gpta_pwm_syncin0_demo函数。

6.10 EPT

6.10.1 ept_capture_demo

1、函数位置

ept_demo.c

5、 功能描述

EPT捕获功能示例。由外部中断EXI事件通过ETCB模块触发EPT SYNCIN2，产生一次捕获。

6、 参考用法

```
//PA01配置为外部中断功能，选择中断组1
......
csi_exi_set_evtrg(EXI_TRGOUT1, TRGSRC_EXI1, 1);
//ETCB配置，设置目标事件(EXI TRGOUT1)和源事件(EPT0 SYNCIN2)：EXI TRGOUT1 -> ETCB -> EPT0 SYNCIN2
......
csi_etb_init();
......
//初始化tPwmCfg，设置GPTA0的捕获参数
......
csi_ept_capture_init(EPT0, &tPwmCfg);
csi_ept_set_sync (EPT0, EPT_TRG_SYNCEN2, EPT_TRG_CONTINU,EPT_AUTO_REARM_ZRO);
csi_ept_start(EPT0);                        //启动EPT0
```

详见ept_capture_demo函数。

6.10.2 ept_pwm_demo

1、 函数位置

ept_demo.c

2、功能描述

EPT波形输出功能示例。可配置PWM周期和占空比，占空比只能为整数，如果有更细致的配置要求可以通过csi_ept_prdr_cmp_update()实现。

3、参考用法

```
csi_pin_set_mux(PA09,PA09_EPT_CHAX);            //PA09作为PWM输出引脚
......
//初始化tPwmCfg，设置PWM周期和占空比
......
csi_ept_wave_init(EPT0, &tPwmCfg);
......
//初始化channel，设置PWM波形输出
......
csi_ept_channel_config(EPT0, &channel,  EPT_CHANNEL_1);
csi_ept_start(EPT0);
csi_ept_change_ch_duty(EPT0,EPT_COMPA, 50);              //修改PWM1占空比为50%
csi_ept_prdr_cmp_update(EPT0,EPT_COMPA,2400,600);    //修改PWM1周期为2400，比较值为600
```

详见ept_pwm_demo函数。

6.10.3 ept_pwm_dz_demo

1、 函数位置

ept_demo.c

2、功能描述

EPT波形输出+死区控制功能示例。可配置PWM波形和死区时间。

3、参考用法

```
csi_pin_set_mux(PA09,PA09_EPT_CHAX);            //PA09作为PWM输出引脚
......
//初始化tPwmCfg，设置PWM周期和占空比
......
csi_ept_wave_init(EPT0, &tPwmCfg);
......
//初始化channel，设置PWM波形输出
......
csi_ept_channel_config(EPT0, &channel,  EPT_CHANNEL_1);
......
//初始化tEptDeadZoneTime，设置死区时间与边沿
......
```

```
csi_ept_dz_config(EPT0,&tEptDeadZoneTime);
csi_ept_start(EPT0);
```

详见ept_pwm_dz_demo函数。

6.10.4 ept_pwm_dz_em_demo

1、 函数位置

ept_demo.c

2、功能描述

EPT波形输出+死区控制+紧急模式功能示例。可配置PWM波形和死区时间，以及紧急模式的处理。

3、参考用法

```
csi_pin_set_mux(PA09,PA09_EPT_CHAX);            //PA09作为PWM输出引脚
......
//初始化tPwmCfg，设置PWM周期和占空比
......
csi_ept_wave_init(EPT0, &tPwmCfg);
......
//初始化channel，设置PWM波形输出
......
csi_ept_channel_config(EPT0, &channel,  EPT_CHANNEL_1);
......
//初始化tEptDeadZoneTime，设置死区时间与边沿
......
csi_ept_dz_config(EPT0,&tEptDeadZoneTime);
......
//初始化tEptEmergencyCfg，设置紧急模式的输入源与处理方式
......
csi_ept_emergency_config(EPT0,&tEptEmergencyCfg);
csi_ept_start(EPT0);
```

详见ept_pwm_dz_em_demo函数。

6.11 ADC

6.11.1 adc_samp_oneshot_demo

1、 函数位置

adc_demo.c

2、 功能描述

ADC单次采样功能示例。启动后进行整个序列的采样，采样完成后停止。

3、参考用法

```
csi_pin_set_mux(PA10, PA10_ADC_AIN2); // PA10作为ADC输入引脚
......
//初始化tAdcConfig，配置ADC参数，单次采样
......
csi_adc_init(ADC0, &tAdcConfig);            //初始化ADC参数配置
csi_adc_start(ADC0);
```

详见adc_samp_oneshot_demo函数。

6.11.2 adc_samp_continuous_demo

1、函数位置

adc_demo.c

2、 功能描述

ADC连续采样功能示例。启动后进行整个序列的采样，采样完成后继续从序列第一个通道开始，如此循环。

3、参考用法

```
csi_pin_set_mux(PA10, PA10_ADC_AIN2); // PA10作为ADC输入引脚
......
//初始化tAdcConfig，配置ADC参数，连续采样
......
csi_adc_init(ADC0, &tAdcConfig);            //初始化ADC参数配置
csi_adc_start(ADC0);
```

详见adc_samp_continuous_demo函数。

6.11.3 adc_samp_oneshot_int_demo

1、 函数位置

adc_demo.c

2、功能描述

ADC单次采样中断模式功能示例。启动后进行整个序列的采样，采样完成后停止。

3、参考用法

```
csi_pin_set_mux(PA10, PA10_ADC_AIN2); // PA10作为ADC输入引脚
......
//初始化tAdcConfig，配置ADC参数，单次采样，中断模式
......
csi_adc_init(ADC0, &tAdcConfig);            //初始化ADC参数配置
csi_adc_start(ADC0);
```

详见adc_samp_oneshot_int_demo函数。

6.11.4 adc_samp_continuous_int_demo

1、 函数位置

adc_demo.c

2、功能描述

ADC连续采样中断模式功能示例。启动后进行整个序列的采样，采样完成后继续从序列第一个通道开始，如此循环。

3、参考用法

```
csi_pin_set_mux(PA10, PA10_ADC_AIN2); // PA10作为ADC输入引脚
......
//初始化tAdcConfig，配置ADC参数，连续采样，中断模式
......
csi_adc_init(ADC0, &tAdcConfig);            //初始化ADC参数配置
csi_adc_start(ADC0);
```

详见adc_samp_continuous_int_demo函数。

6.12 CRC

6.12.1 crc_demo

1、 函数位置

crc_demo.c

2、 功能描述

CRC功能示例。

3、参考用法

```
csi_crc_init();     //CRC模块初始化
//进行CRC-32模式计算，种子值0xffffffff，byTransData数组前3个数据，数据长度3个字节
csi_crc32_be(0xffffffff, byTransData, 3);
//进行CRC-16/CCITT模式计算，种子值0x00，byTransData数组前16个数据，数据长度16个字节
csi_crc16_ccitt(0x00, byTransData, 16);
//进行CRC-16模式计算，种子值0x00,byTransData数组前5个数据，数据长度5个字节
csi_crc16(0x00, byTransData, 5);
//进行CRC-16 XMODEM模式计算，种子值0x00，byTransData数组前3个数据，数据长度3个字节
csi_crc16_itu(0x00,   byTransData, 3);
```

详见crc_demo函数。

6.13 ETCB

6.13.1 etcb_one_trg_one_demo0

1、 函数位置

etcb_demo.c

2、 功能描述

ETCB一对一触发模式功能示例0。外部中断EXI事件通过ETCB模块触发BT0 SYNC PORT0，即BT0启动；BT0 PEND事件通过ETCB模块触发BT1 SYNC PORT0，即BT1启动。

3、参考用法

```
//PA01配置为外部中断功能，选择中断组1
......
csi_exi_set_evtrg(EXI_TRGOUT1, TRGSRC_EXI1, 1);
csi_bt_start_sync(BT0, 10);
csi_bt_set_sync(BT0,BT_TRG_SYNCIN0, BT_TRG_ONCE, BT_AREARM_PEND); //外部触发BT0启动(SYNCIN0),使用PEND事件REARM
csi_bt_set_evtrg(BT0, BT_TRGOUT, BT_TRGSRC_PEND); //BT0 PEND事件触发输出
csi_bt_start_sync(BT1, 10);
csi_bt_set_sync(BT1,BT_TRG_SYNCIN0, BT_TRG_ONCE, BT_AREARM_PEND); //外部触发BT1启动(SYNCIN0),使用PEND事件REARM
//ETCB配置，设置目标事件(EXI TRGOUT1)和源事件(BT0 SYNCIN0)：EXI TRGOUT1 -> ETCB -> BT0 SYNCIN0
//ETCB配置，设置目标事件(BT0 PEND)和源事件(BT1 SYNCIN0)：BT0 PEND -> ETCB -> BT1 SYNCIN0
......
csi_etb_init();
......
```

详见etcb_one_trg_one_demo0函数。

6.13.2 etcb_one_trg_one_demo1

1、函数位置

etcb_demo.c

2、功能描述

ETCB一对一触发模式功能示例1。EPT PEND事件通过ETCB模块触发ADC采样一次。

3、参考用法

```
etcb_ept_config();              //EPT初始化，设置EPT0 PRD事件触发输出
etcb_adc_config();              //ADC初始化，设置ADC_SYNCIN0触发输入
//ETCB配置，设置目标事件(EPT0 PRD)和源事件(ADC_SYNCIN0)：EPT0 PRD -> ETCB -> ADC_SYNCIN0
......
csi_etb_init();
......
```

详见etcb_one_trg_one_demo1函数。

6.13.3 etcb_one_trg_more_demo

1、函数位置

etcb_demo.c

2、功能描述

ETCB一对多触发模式功能示例。EPT PEND事件通过ETCB模块触发ADC SYNCIN0和ADC SYNCIN1。

3、参考用法

```
etcb_ept_config();       //EPT初始化，设置EPT0 PRD事件触发输出
etcb_adc_config12();  //ADC初始化，设置ADC SYNCIN0和ADC SYNCIN1触发输入
//ETCB配置，设置目标事件(EPT0 PRD)和源事件(ADC_SYNCIN0 / ADC SYNCIN1)：EPT0 PRD -> ETCB -> ADC_SYNCIN0 & ADC SYNCIN1
......
csi_etb_init();
......
```

详见etcb_one_trg_more_demo函数。

6.13.4 etcb_mix_demo

1、函数位置

etcb_demo.c

2、功能描述

ETCB一对一、一对多混合触发模式功能示例。外部中断EXI事件通过ETCB模块触发BT0 Sync Port0，即BT0启动；BT0 PEND事件通过ETCB模块触发BT1/BT2 Sync Port0，即BT1/BT2启动。

3、参考用法

```
//PA01配置为外部中断功能，选择中断组1
......
csi_bt_start_sync(BT0, 200);
csi_bt_set_sync(BT0,BT_TRG_SYNCIN0, BT_TRG_ONCE, BT_AREARM_PEND);
//外部触发BT0启动(SYNCIN0),使用PEND事件REARM
csi_bt_set_evtrg(BT0, BT_TRGOUT, BT_TRGSRC_PEND);//BT0 PEND事件触发输出
csi_bt_start_sync(BT1, 100);
csi_bt_set_sync(BT1,BT_TRG_SYNCIN0, BT_TRG_ONCE, BT_AREARM_PEND);
//外部触发BT1启动(SYNCIN0),使用PEND事件REARM
csi_bt_start_sync(BT2, 50);
csi_bt_set_sync(BT2,BT_TRG_SYNCIN0, BT_TRG_ONCE,
BT_AREARM_PEND);//外部触发BT2启动(SYNCIN0),使用PEND事件REARM
//ETCB配置，一对一模式，设置目标事件(EXI TRGOUT1)和源事件(BT0 SYNCIN0)：EXI TRGOUT1 -> ETCB -> BT0 SYNCIN0
```

```
//ETCB配置，一对多模式，设置目标事件(BT0 PEND)和源事件(BT1/BT2 SYNCIN0)：BT0 PEND -> ETCB -> BT1/BT2 SYNCIN0
......
csi_etb_init();
......
```

详见etcb_mix_demo函数。

6.14 UART

6.14.1 uart_char_demo

1、函数位置

uart_demo.c

2、功能描述

UART发送/接收一个字符功能示例。使用轮询方式。

3、 参考用法

```
//初始化tUartConfig，设置UART基本参数：波特率115200，奇校验，发送和接收都是轮询模式，不开启中断
......
csi_uart_init(UART0, &tUartConfig);                 //初始化串口
csi_uart_start(UART0, UART_FUNC_RX_TX);   //开启UART的RX和TX功能，也可单独开启
byRecv = csi_uart_getc(UART0);                      //接收一个字符
csi_uart_putc(UART0, byRecv+1);                     //发送一个字符
```

详见uart_char_demo函数。

6.14.2 uart_send_demo

1、函数位置

uart_demo.c

2、功能描述

UART发送一串数据功能示例。使用轮询方式。

4、 参考用法

```
//初始化tUartConfig，设置UART基本参数：波特率115200，奇校验，发送和接收都是轮询模式，不开启中断
......
csi_uart_init(UART0, &tUartConfig);                 //初始化串口
csi_uart_start(UART0, UART_FUNC_RX_TX);       //开启UART的RX和TX功能，也可单独开启
csi_uart_send(UART0,(void *)bySendData,18);       //发送18个字节的数据
```

详见uart_send_demo函数。

6.14.3 uart_send_int_demo

1、函数位置

uart_demo.c

2、功能描述

UART发送一串数据功能示例。使用中断方式。

3、 参考用法

```
//初始化tUartConfig，设置UART基本参数：波特率115200，奇校验，发送中断模式，接收轮询模式，使用TX中断
......
csi_uart_init(UART0, &tUartConfig);                    //初始化串口
csi_uart_start(UART0, UART_FUNC_RX_TX);      //开启UART的RX和TX功能，也可单独开启
csi_uart_send(UART0,(void *)bySendData,28);      //采用中断方式。调用该函数时，UART发送中断使能
```

详见uart_send_int_demo函数。

6.14.4 uart_receive_demo

1、函数位置

uart_demo.c

2、功能描述

UART接收功能示例。使用轮询方式，接收指定长度数据。

3、参考用法

```
//初始化tUartConfig，设置UART基本参数：波特率115200，奇校验，发送和接收都是轮询模式，不开启中断
......
csi_uart_init(UART0, &tUartConfig);                    //初始化串口
csi_uart_start(UART0, UART_FUNC_RX_TX);     //开启UART的RX和TX功能，也可单独开启
csi_uart_receive(UART0,byRecvData,16,2000);   //接收16字节数据，超时时间2000ms
```

详见uart_receive_demo函数

6.14.5 uart_recv_rx_int_demo

1、函数位置

uart_demo.c

2、功能描述

UART接收功能示例。使用接收中断方式。

4、 参考用法

```
//初始化tUartConfig，设置UART基本参数：波特率115200，奇校验，发送轮询模式，接收中断模式，使用RX中断
......
csi_uart_init(UART0, &tUartConfig);              //初始化串口
csi_uart_start(UART0, UART_FUNC_RX_TX); //开启UART的RX和TX功能，也可单独开启
```

详见uart_recv_rx_int_demo函数。

6.14.6 uart_recv_rxfifo_int_demo

1、函数位置

uart_demo.c

2、功能描述

UART接收功能示例。使用接收FIFO中断方式。

2、 参考用法

```
//初始化tUartConfig，设置UART基本参数：波特率115200，奇校验，发送轮询模式，接收中断模式，使用RXFIFO中断和接收超时中断
......
csi_uart_init(UART0, &tUartConfig);              //初始化串口
csi_uart_start(UART0, UART_FUNC_RX_TX);   //开启UART的RX和TX功能，也可单独开启
```

详见uart_recv_rxfifo_int_demo函数。

6.15 USART

6.15.1 usart_char_demo

1、函数位置

usart_demo.c

2、功能描述

USART发送/接收一个字符功能示例。使用轮询方式。

3、参考用法

```
//初始化tUsartCfg，设置USART基本参数：波特率115200，偶校验，发送和接收都是轮询模式，不使用中断
......
csi_usart_init(USART0, &tUsartCfg);                //初始化串口
csi_usart_start(USART0, USART_FUNC_RX_TX);   //开启USART的RX和TX功能，也可单独开启
byRecv = csi_usart_getc(USART0);                   //接收一个字符
csi_usart_putc(USART0, byRecv+1);                  //将接收到的字符加1后发送出去
```

详见usart_char_demo函数。

6.15.2 usart_send_demo

1、函数位置

usart_demo.c

2、功能描述

USART发送一串数据功能示例。使用轮询方式。

3、参考用法

```
//初始化tUsartCfg，设置USART基本参数：波特率115200，偶校验，发送和接收都是轮询模式，不使用中断
......
csi_usart_init(USART0, &tUsartCfg);                    //初始化串口
csi_usart_start(USART0, USART_FUNC_RX_TX);   //开启USART的RX和TX功能，也可单独开启
csi_usart_send(USART0,(void *)bySdData,18);        //发送18字节的数据
```

详见usart_send_demo函数。

6.15.3 usart_send_int_demo

1、函数位置

usart_demo.c

2、功能描述

USART发送一串数据功能示例。使用中断方式。

3、 参考用法

```
//初始化tUsartCfg，设置USART基本参数：波特率115200，偶校验，发送中断模式，接收轮询模式，使用TXFIFO中断
......
csi_usart_init(USART0, &tUsartCfg);                    //初始化串口
csi_usart_start(USART0, USART_FUNC_RX_TX);   //开启USART的RX和TX功能，也可单独开启
csi_usart_send(USART0,(void *)bySdData,18);        //发送18字节的数据
```

详见usart_send_int_demo函数。

6.15.4 usart_recv_demo

1、函数位置

usart_demo.c

2、功能描述

USART接收功能示例。使用轮询方式，接收指定长度数据。

3、参考用法

```
//初始化tUsartCfg，设置USART基本参数：波特率115200，偶校验，发送和接收都为轮询模式，不使用中断
......
csi_usart_init(USART0, &tUsartCfg);                    //初始化串口
csi_usart_start(USART0, USART_FUNC_RX_TX);             //开启USART的RX和TX功能，也可单独开启
csi_usart_receive (USART0, (void *)byRxBuf,16,1000);   //接收16字节数据，超时时间1000ms
```

详见usart_recv_demo函数。

6.15.5 usart_recv_rx_int_demo

1、函数位置

usart_demo.c

2、功能描述

USART接收功能示例。使用接收中断方式。

4、 参考用法

```
//初始化tUsartCfg，设置USART基本参数：波特率115200，偶校验，发送轮询模式，接收中断模式，使用RX中断
......
csi_usart_init(USART0, &tUsartCfg);                    //初始化串口
csi_usart_start(USART0, USART_FUNC_RX_TX);             //开启USART的RX和TX功能，也可单独开启
```

详见usart_recv_rx_int_demo函数。

6.15.6 usart_recv_rxfifo_int_demo

1、函数位置

usart_demo.c

2、功能描述

USART接收功能示例。使用接收FIFO中断方式。

2、 参考用法

```
//初始化tUsartCfg，设置USART基本参数：波特率115200，偶校验，发送轮询模式，接收中断模式，使用RXFIFO中断和接收超时中断
......
csi_usart_init(USART0, &tUsartCfg);                    //初始化串口
csi_usart_start(USART0, USART_FUNC_RX_TX);             //开启USART的RX和TX功能，也可单独开启
```

详见usart_recv_rxfifo_int_demo函数。

6.16 CMP

6.16.1 cmp_base_demo

1、函数位置

cmp_demo.c

2、功能描述

CMP基本功能示例。比较器的P极和N极输入不同电平值，如果P>N，输出高电平;如果P<N，输出低电平。

3、 参考用法

```
csi_pin_set_mux(PB02,PB02_CPINN0);    //设置PB02为比较器的N级输入
csi_pin_set_mux(PA02,PA02_CPINP0);    //设置PA02为比较器的P级输入
csi_pin_set_mux(PB03,PB03_CP0_OUT);   //设置PB03为比较器的输出
//初始化tCmpCfg，设置CMP基本参数：输入端口，参考电压，输出极性等
......
csi_cmp_init(CMP,&tCmpCfg,CMP_IDX0);
csi_cmp_start(CMP,CMP_IDX0);
```

详见cmp_base_demo函数。

6.16.2 cmp_dfcr_demo

1、函数位置

cmp_demo.c

2、功能描述

CMP滤波功能示例。该滤波器是一个积分滤波器，滤波的固定深度为3。

4、 参考用法

```
csi_pin_set_mux(PB02,PB02_CPINN0);    //设置PB02为比较器的N级输入
csi_pin_set_mux(PA02,PA02_CPINP0);    //设置PA02为比较器的P级输入
csi_pin_set_mux(PB06,PB06_CP1_OUT);   //设置PB06为比较器的输出
//初始化tCmpCfg，设置CMP基本参数：输入端口，参考电压，输出极性等
......
csi_cmp_init(CMP,&tCmpCfg,CMP_IDX1);
//初始化tCmpFltcrCfg，设置滤波器的时钟及滤波周期
......
csi_cmp_fltcr_config(CMP,ENABLE,&tCmpFltcrCfg,CMP_IDX1);
csi_cmp_start(CMP,CMP_IDX1);
```

详见cmp_dfcr_demo函数。

6.16.3 cmp_wfcr_demo

1、函数位置

cmp_demo.c

2、功能描述

CMP窗口滤波功能示例。BT0通过ETCB模块触发比较器窗口滤波功能。在特定的触发窗口内，比较器的输出将通过一个采样寄存器输出（该寄存器的采样时钟为PCLK）。一旦该窗口关闭，寄存器将保持该输出状态或者以指定逻辑输出，直到下次的窗口有效。捕获窗口可以设置相应的延时时间，在触发信号有效后，延时特定的时间，再使能捕获窗口。该功能适用于在某些强干扰环境中，对比较器输出在特定时间内进行分析的应用。

3、 参考用法

```
csi_pin_set_mux(PB02,PB02_CPINN0);      //设置PB02为比较器的N级输入
csi_pin_set_mux(PA02,PA02_CPINP0);      //设置PA02为比较器的P级输入
csi_pin_set_mux(PA07,PA07_CP4_OUT);   //设置PA07为比较器的输出
//初始化tCmpCfg，设置CMP基本参数：输入端口，参考电压，输出极性等
......
csi_cmp_init(CMP,&tCmpCfg,CMP_IDX4);
//初始化tCmpWfcrCfg，设置窗口滤波器的时钟分频及延迟参数等
......
csi_cmp_wfcr_config(CMP,&tCmpWfcrCfg,CMP_IDX4);
csi_cmp_start(CMP,CMP_IDX4);
csi_bt_timer_init(BT0, 40000);                          //初始化BT0，定时40000us
csi_bt_start(BT0);                                      //启动定时器
csi_bt_set_evtrg(BT0, 0, BT_TRGSRC_PEND);          //BT0 PEND事件触发输出
//ETCB配置，设置目标事件(BT0 PEND)和源事件(CMP4 SYNCIN)：BT0 PEND -> ETCB -> CMP4 SYNCIN
......
csi_etb_init();
```

详见cmp_wfcr_demo函数。

6.16.4 cmp_trg_out_demo

1、函数位置

cmp_demo.c

2、功能描述

CMP触发输出功能示例。CMP事件输出通过ETCB模块触发BT0 SYNC PORT0，即BT0启动。

4、 参考用法

```
//初始化tCmpCfg，设置CMP基本参数：输入端口，参考电压，输出极性等
......
csi_cmp_init(CMP,&tCmpCfg,CMP_IDX5);
csi_cmp_start(CMP,CMP_IDX5);
csi_cmp_set_evtrg(CMP, CMP_TRGOUT, CMP_TRGSRC_CMP5OUT5_6,ENABLE);
csi_bt_timer_init(BT0, 40000);                          //初始化BT0，定时40000us
```

```
csi_bt_start(BT0);                                    //启动定时器
csi_bt_set_evtrg(BT0, 0, BT_TRGSRC_PEND);        //BT0 PEND事件触发输出
//ETCB配置，设置目标事件(CMP TRGOUT6)和源事件(BT0 SYNCIN0)：CMP TRGOUT6 -> ETCB -> BT0 SYNCIN0
......
csi_etb_init();
```

详见cmp_trg_out_demo函数。

6.17 OPA

6.17.1 opa_internal_gain_mode_test

1、函数位置

opa_demo.c

2、功能描述

OPA内部增益模式功能示例。使用内部增益，使用ADC采集输入，设置需要放大的倍数。

3、 参考用法

```
//配置OPA0/1的输入输出引脚
......
//初始化tAdcConfig，设置分频系数及参考电压等
......
csi_adc_init(ADC0, &tAdcConfig);
//初始化ptOpaConfig_t，使用内部增益，选择增益倍数
csi_opa_init(OPA0,&ptOpaConfig_t);
csi_opa_start(OPA0);
csi_opa_init(OPA1,&ptOpaConfig_t);
csi_opa_start(OPA1);
csi_adc_start(ADC0);
```

详见opa_internal_gain_mode_test函数。

6.17.2 opa_external_gain_mode_test

1、函数位置

opa_demo.c

2、功能描述

OPA外部增益模式功能示例。使用外部增益，放大倍数由外部反馈电阻确定。

4、 参考用法

```
//配置OPA0/1的输入输出引脚
......
```

```
//初始化tAdcConfig，设置分频系数及参考电压等
......
csi_adc_init(ADC0, &tAdcConfig);
//初始化ptOpaConfig_t，使用外部增益
csi_opa_init(OPA0,&ptOpaConfig_t);
csi_opa_start(OPA0);
csi_opa_init(OPA1,&ptOpaConfig_t);
csi_opa_start(OPA1);
csi_adc_start(ADC0);
```

详见opa_external_gain_mode_test函数。