| 版本 | V1.0 |
| --- | --- |
| 日期 | 202305 |

# Quick Start

# APT32F173x 系列

**Revision History**

| 版本 | 日期 | 描述 | 作者 |
|---|---|---|---|
| V1.0 | 2023-5-15 | 新建使用说明 | YYM |
| | | | |
| | | | |

# 目录

1. 文档说明 ..... 5

2. 资源列表 ..... 5

3. 硬件环境 ..... 5

3.1 调试模块（蓝色部分） ..... 6

3.2 芯片（红色部分） ..... 6

4. 软件环境 ..... 7

4.1 CDK 开发环境 ..... 7

4.2 代码结构 ..... 7

5. 例程运行 ..... 9

5.1 例程从原厂获取 ..... 9

5.2 例程从 CDK 中获取 ..... 9

5.3 工程配置 ..... 12

5.4 导入 flash 算法 ..... 14

5.5 适配代码 ..... 15

5.6 外设使用 ..... 21

5.7 编译工程 ..... 21

5.8 下载调试 ..... 22

6. APPENDIX ..... 23

6.1 GPIO ..... 23

6.2 Reliability ..... 25

6.3 IWDT ..... 27

6.4 WWDT ..... 28
6.5 IFC ..... 28
6.6 PM ..... 30
6.7 BT ..... 31
6.8 CNTA ..... 34
6.9 GPTA ..... 35
6.10 GPTB ..... 39
6.11 RTC ..... 41
6.12 ADC ..... 43
6.13 FVR ..... 45
6.14 DAC ..... 46
6.15 CRC ..... 46
6.16 ETCB ..... 47
6.17 UART ..... 48
6.18 USART ..... 50
6.19 CAN ..... 53
6.20 SPI ..... 55
6.21 IIC ..... 58
6.22 LED ..... 61
6.23 SIO ..... 61
6.24 CMP ..... 66
6.25 OPA ..... 68
6.26 WIZARD ..... 69

# 1. 文档说明

该文档基于爱普特CSI驱动代码架构，适用于APT32F173x系列芯片

# 2. 资源列表

为了使用APT32F173x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173x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。
- **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 APT32F173x 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。

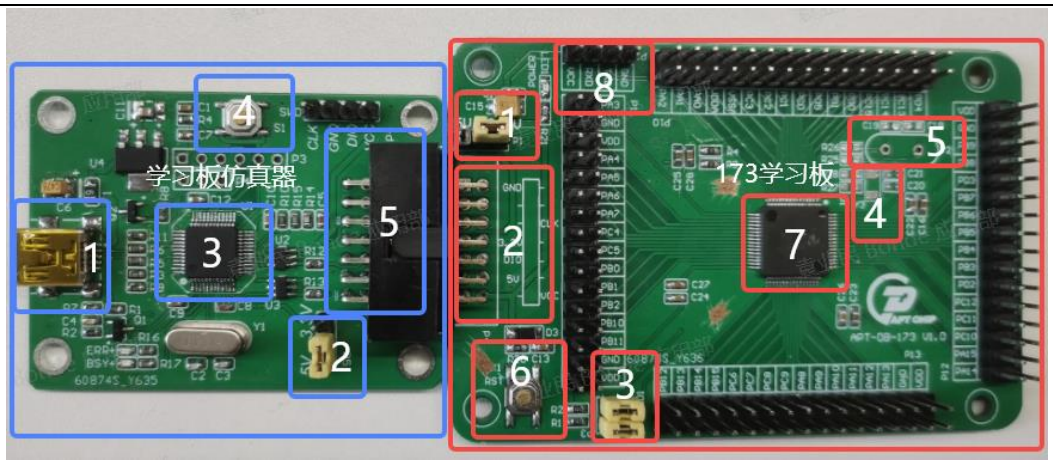



**Figure 1 学习板仿真器及 173 学习板**

## 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、 反馈电压，5V/3V3 可选。

2、 SWD 调试接口。与调试模块连接的接口。

3、 SWD 跳线。APT32F173x 系列只有 1 组 SWD。SWD 口为 PA14（SWCLK）和 PA13（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 PA13 和 PA14。使用时需用跳线连接

4、 副晶振电路和跳线。APT32F173x 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、 主晶振电路和跳线。APT32F173x 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。

6、 复位电路和跳线。APT32F173x 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、 APT32F173x 系列芯片。

8、 调试串口，使用的为 UART1,,其中 PA2 为 TX,PA3 为 RX。

# 4. 软件环境

## 4.1 CDK开发环境

### 4.1.1 CDK下载

APT32F173x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=4176310636235526144

### 4.1.2 CDK开发环境注意事项

- CDK 不支持中文路径，安装路径及工程路径上不可以有空格或中文字符。
- 使用CDK2.20以后的版本，以确保debug server版本5.16.8+。
- 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。
- 软复位只复位 CPU，如果要复位整个芯片，请使用硬复位功能
- 如果要调试 IWDT，WWDT 等可以触发复位的功能，需要在 ICE Configuration/ICE/Other Flags 下添加 -setresethaltreq-always-on 。

## 4.2 代码结构

下图为工程视图。这里面包含了6个组件。

apt32f173x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。

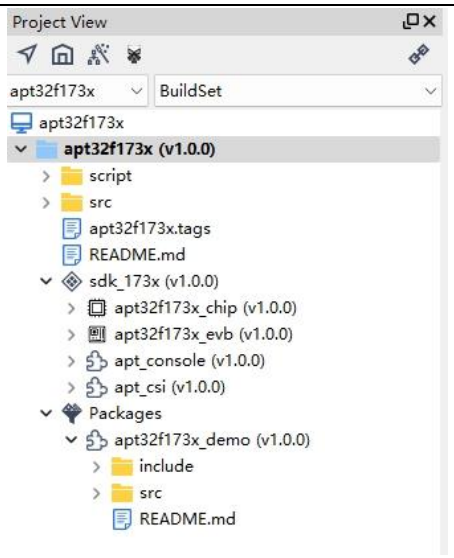



**Figure 2** 代码结构

# 5. 例程运行

## 5.1 例程从原厂获取

从原厂FAE处，获取最新的例程。

## 5.2 例程从CDK中获取

### 5.2.1 SDK包获取

1、打开CDK，更新demo工程（CDK版本为V2.19.5）

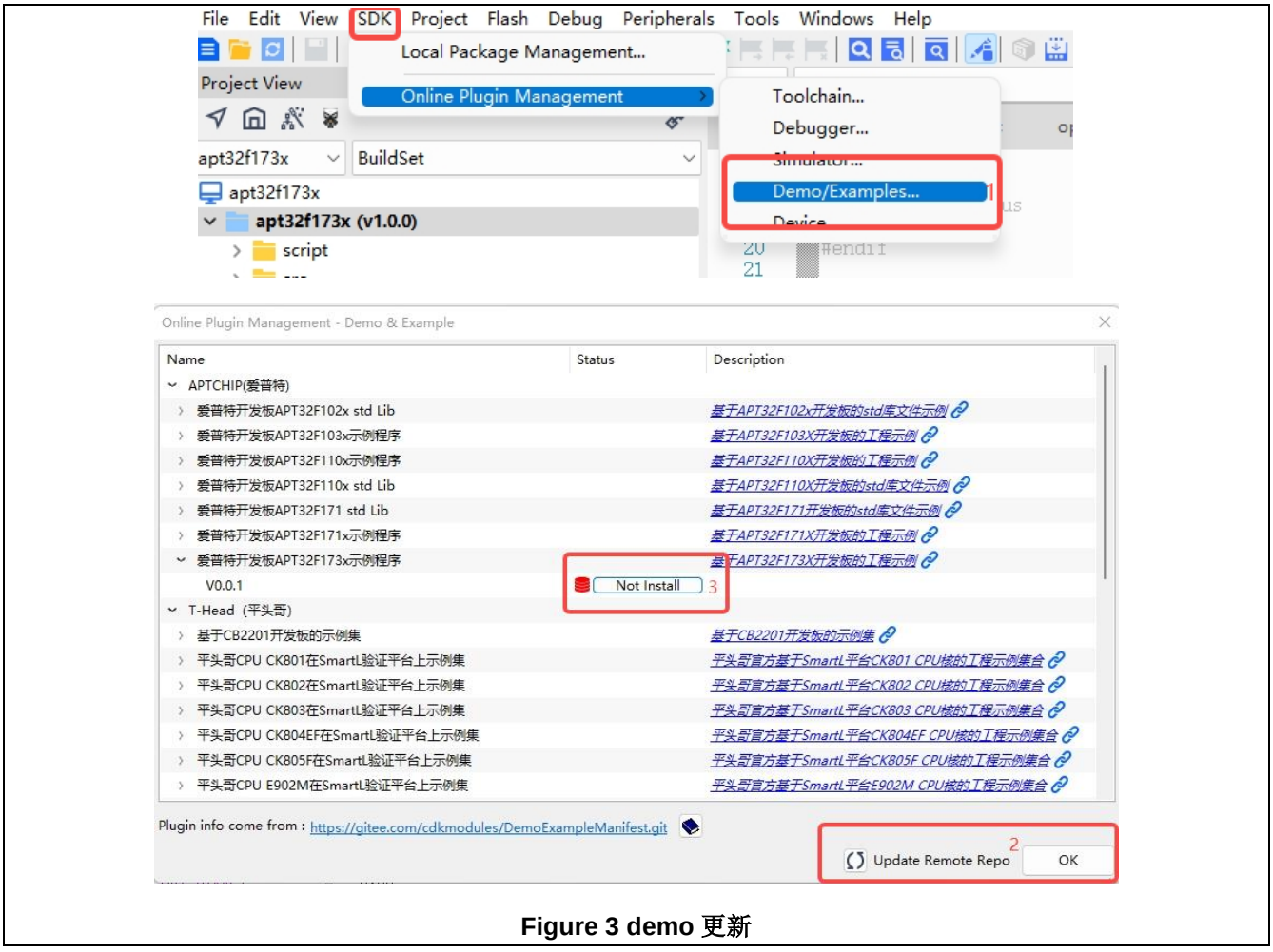



**Figure 3 demo 更新**

2、Demo工程更新完成会显示Installed，更新完成的Demo & Example页面如下图

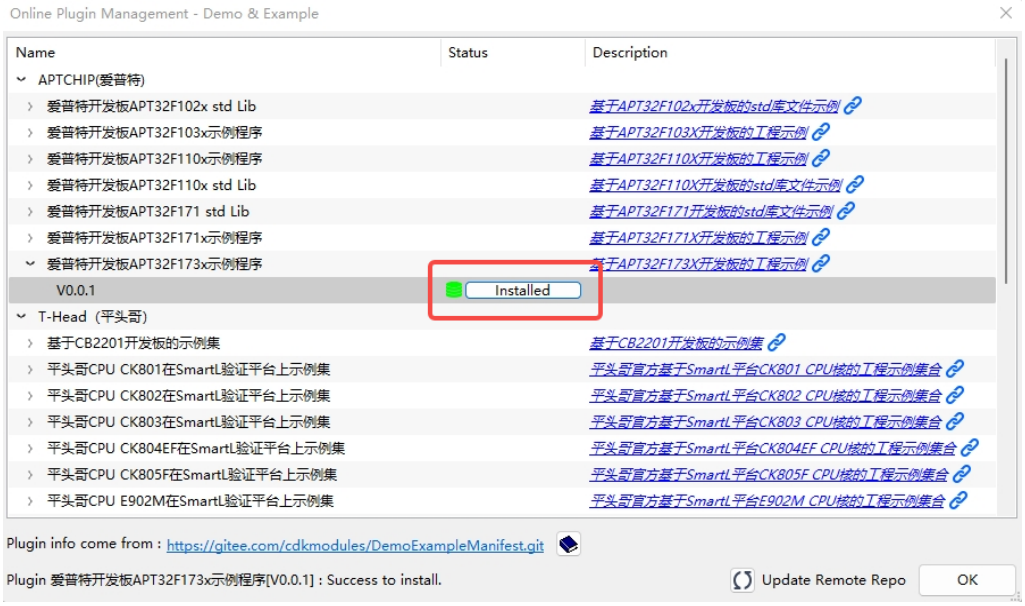
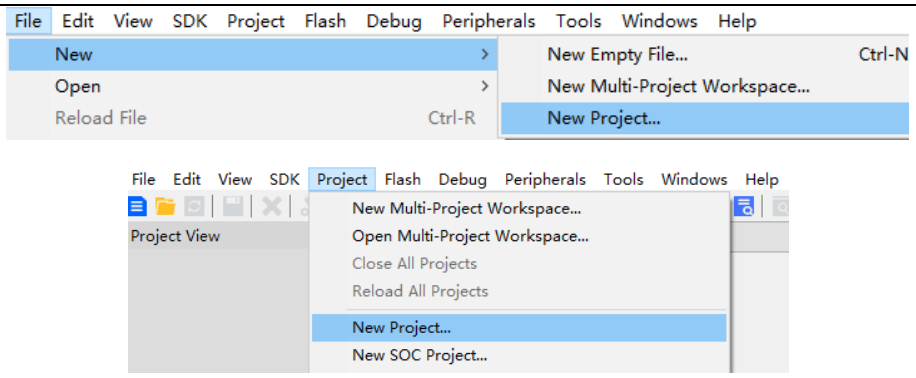



**Figure 4 更新完成页面**

### 5.2.2 创建工程

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。



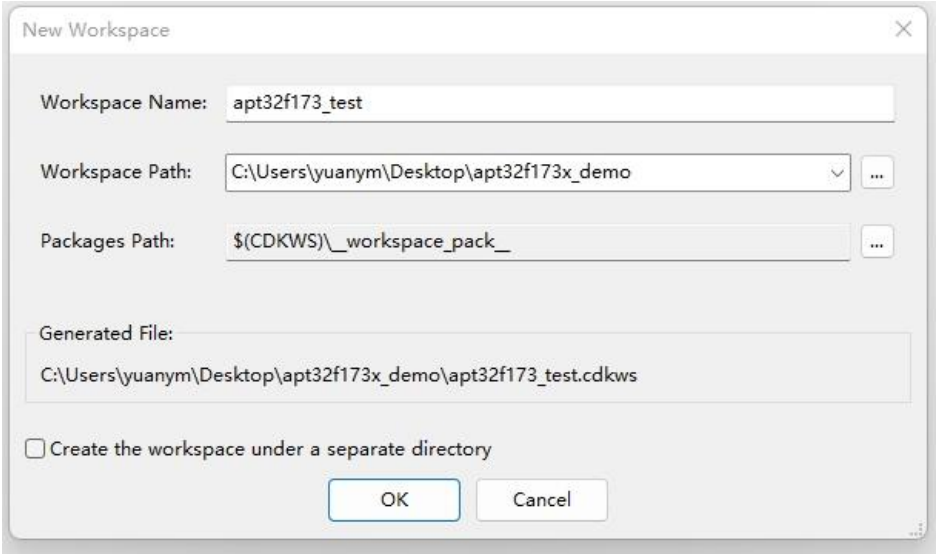



**Figure 5 创建新的 Project**

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”apt32f173x_demo”。

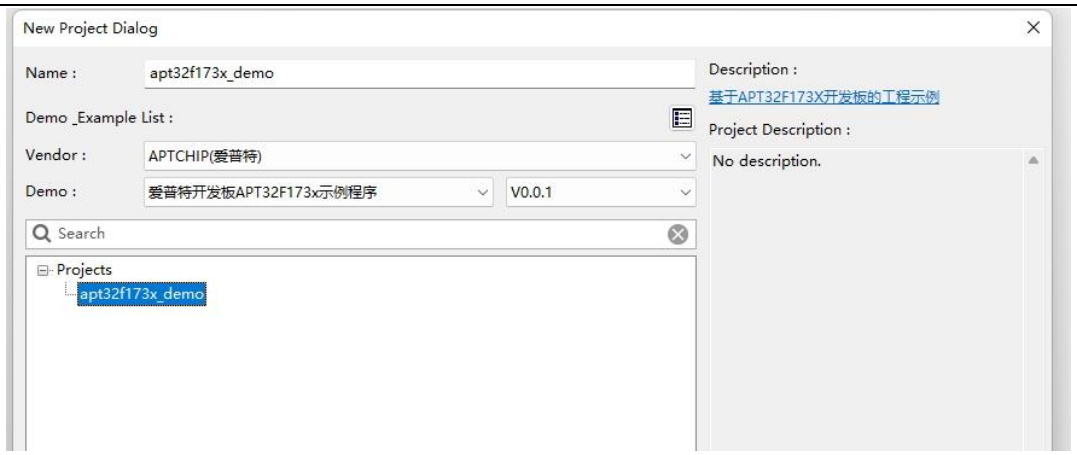



**Figure 6 配置工程参数**

3、 到这里，我们就完成了APT32F173x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。

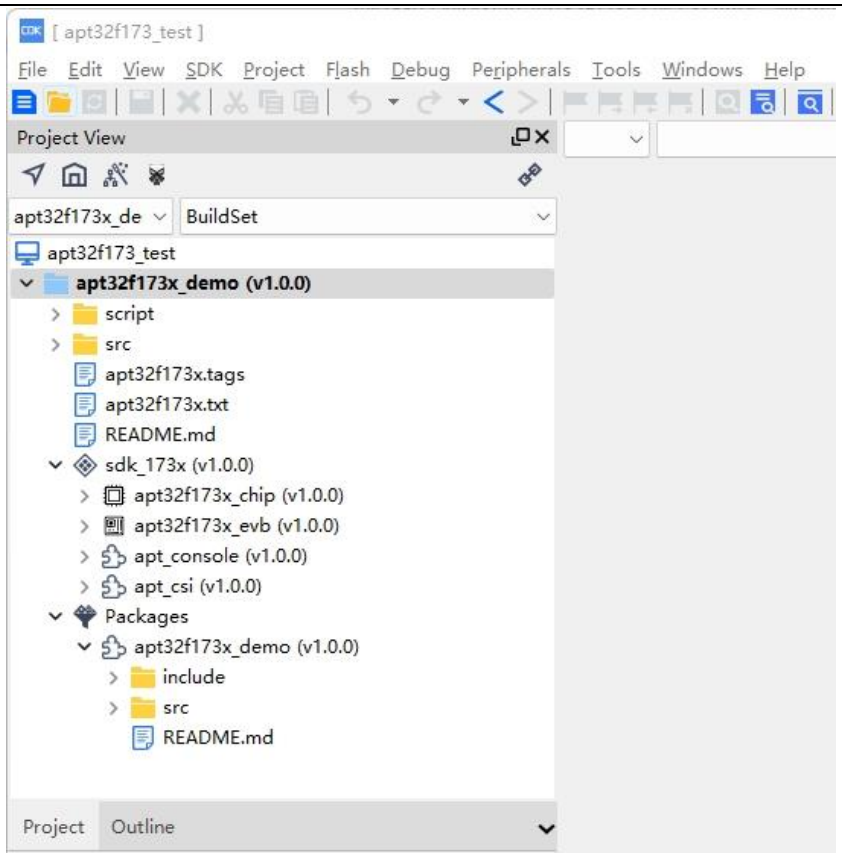



**Figure 7 创建好的工程目录树**

## 5.3 工程配置

1. 配置选项Compiler

- 标注1，Debug Level：编译的调试选项；包含None -g -g1 -g3，None表示不包含调试信息，-g3包含最全面的的调试信息；不同的调试选项会影响生成elf的文件大小，但是不影响最终下载到目标板的image大小，所以建议开最大即可。若不开调试信息选项，将导致无法进行源码调试，只能进行汇编调试；
- 标注2，Optimization：编译的优化选项；分为-O0，-Og，-O1，-O2，-O3，-Os六种类型：

  例如：-Os，该优化会以代码密度为优先，尽可能优化生成的代码。在此选项下，生成的代码较小，比较适合空间敏感的 MCU 程序编译。

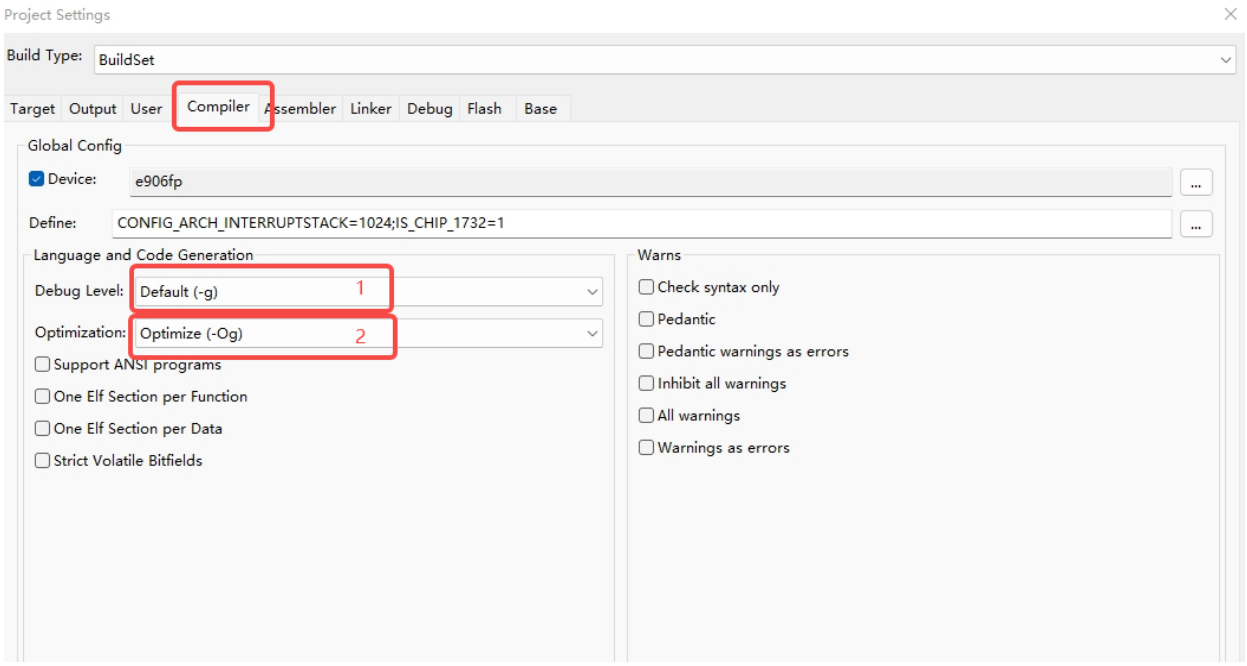



**Figure 8 工程配置**

2. 配置选项Output

- 标注3，表示工程编译后，会输出hex文件；
- 标注4，表示工程编译后，会输出bin文件;
- 标注5，表示工程编译后，会输出.asm汇编文件，可查看函数的汇编运行指令;
- 标注6，表示工程编译后，会输出.map汇编文件,可通过此文件，可查看变量和函数空间分配等信息.

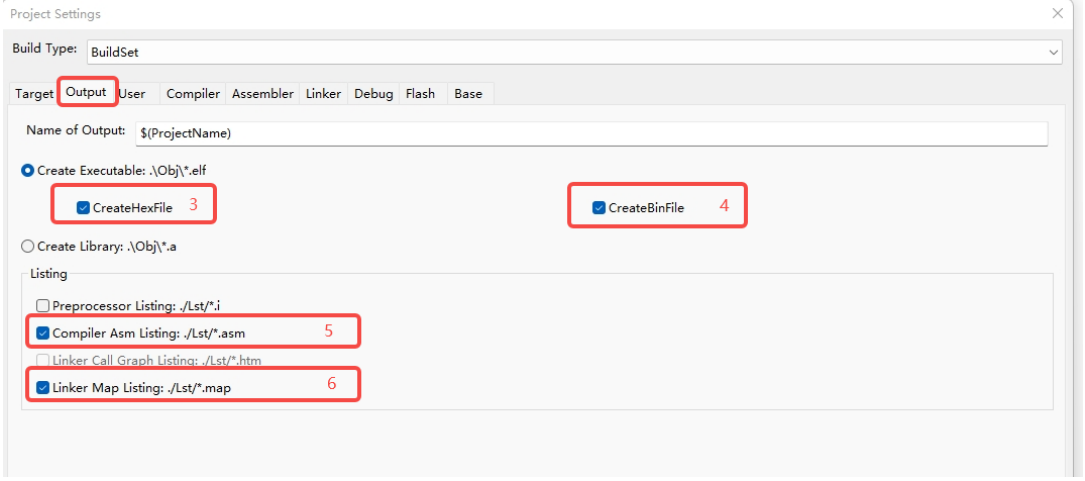



**Figure 9 工程配置**

3. 配置选项Linker

- 标注7选择Linker文件,一般在mcu厂商提供的SDK包的board目录下，名称叫gcc_flash_1732.ld。点击标注8按钮，在标注9窗口，可根据需求，选择合适的ld文件。

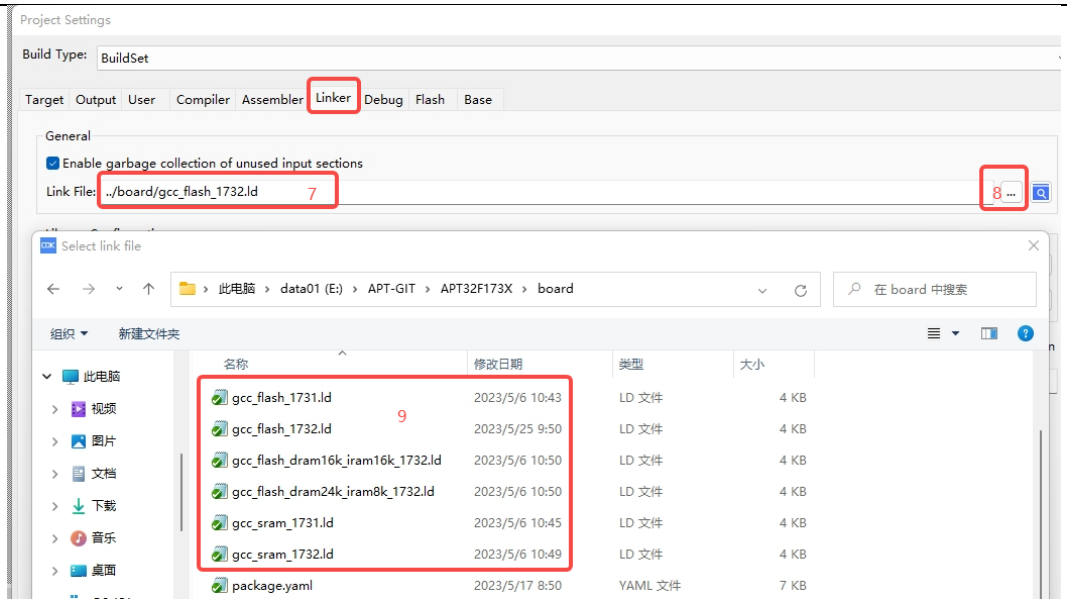



**Figure 10 工程配置**

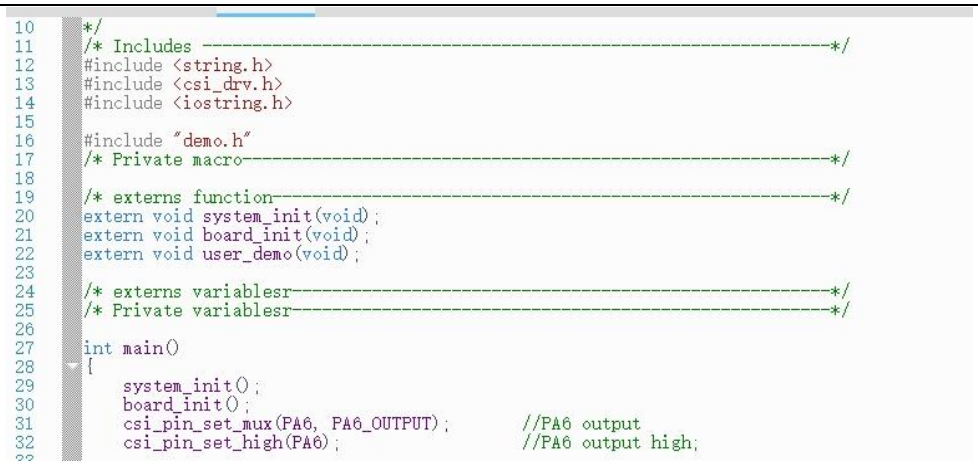
## 5.4 导入flash算法

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。

## 5.5 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。



```
*/
/* Includes ------------------------------------------------------------------*/
#include <string.h>
#include <csi_drv.h>
#include <iostring.h>

#include "demo.h"
/* Private macro-------------------------------------------------------------*/

/* externs function---------------------------------------------------------*/
extern void system_init(void);
extern void board_init(void);
extern void user_demo(void);

/* externs variablesr-------------------------------------------------------*/
/* Private variablesr-------------------------------------------------------*/

int main()
{
    system_init();
    board_init();
    csi_pin_set_mux(PA6, PA6_OUTPUT);       //PA6 output
    csi_pin_set_high(PA6);                  //PA6 output high;
```

**Figure 11 main 函数**

### 5.5.1 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)<br>{<br>uint32_t i;<br>csi_icache_enable ();<br>#ifdef CODE_REMAP_TO_IRAM<br>csi_iram_init(); //需要与 gcc_flash_dram16k_iram16k 或 gcc_flash_dram24k_iram8k.ld 配套使用<br>#endif<br>__disable_excp_irq();<br>uint32_t mstatus = __get_MSTATUS();<br>mstatus \|= (1 << 13); //使能 mstatus FS 功能（必须有）<br>__set_MSTATUS(mstatus);<br>/*从信息中获取中断级别*/<br>CLIC->CLICCFG = (((CLIC->CLICINFO & CLIC_INFO_CLICINTCTLBITS_Msk) >><br>CLIC_INFO_CLICINTCTLBITS_Pos) << CLIC_CLICCFG_NLBIT_Pos);<br>for (i = 0; i < 64; i++) {<br>CLIC->CLICINT[i].IP = 0;<br>CLIC->CLICINT[i].ATTR = 1; // 使用矢量中断 （必须有）<br>}<br>#ifdef CONFIG_IRQ_LOOKUP //Table lookup method for interrupt processing<br>irq_vectors_init();<br>#endif<br>csi_iwdt_close(); //关看门狗 iwdt（调试时用）<br>csi_sysclk_config(g_tClkConfig); //配置系统时钟（必须有）<br>csi_tick_init(); //初始化 systick<br>__enable_excp_irq(); //开启全局中断<br>} | system.c |

其中csi_sysclk_config()会去读取一个时钟配置结构体变量g_tClkConfig（位于board_config.c）。可以通过修改g_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
 *
 * such as: g_tClkConfig.eSdiv = SCLK_DIV1, g_tClkConfig.wSclk = g_tClkConfig.wFreq / 1
 *          g_tClkConfig.ePdiv = SCLK_DIV4, g_tClkConfig.wPclk = g_tClkConfig.wSclk / 4
 *
 */
csi_clk_config_t g_tClkConfig =
    //{SRC_HFOSC, HFOSC_24M_VALUE, SCLK_DIV1, PCLK_DIV1, HFOSC_24M_VALUE, HFOSC_24M_VALUE};
    //{SRC_HFOSC, HFOSC_12M_VALUE, SCLK_DIV1, PCLK_DIV1,HFOSC_12M_VALUE, HFOSC_12M_VALUE};
    //{SRC_HFOSC, HFOSC_6M_VALUE, SCLK_DIV1, PCLK_DIV1,HFOSC_6M_VALUE, HFOSC_6M_VALUE};
    //{SRC_HFOSC, HFOSC_3M_VALUE, SCLK_DIV1, PCLK_DIV1,HFOSC_3M_VALUE, HFOSC_3M_VALUE};
    {SRC_AUTO_HF_PLL, PLL_VALUE, SCLK_DIV1, PCLK_DIV1, PLL_VALUE, PLL_VALUE};
    //{SRC_AUTO_EM_PLL, PLL_VALUE, SCLK_DIV1, PCLK_DIV1, PLL_VALUE, PLL_VALUE};
    //{SRC_MANUAL_PLL, PLL_VALUE, SCLK_DIV1, PCLK_DIV1, PLL_VALUE, PLL_VALUE};
    //{SRC_EMOSC, EMOSC_VALUE, SCLK_DIV2, PCLK_DIV2, EMOSC_VALUE/2, EMOSC_VALUE/4};
    //{SRC_IMOSC, IMOSC_5M_VALUE, SCLK_DIV1, PCLK_DIV1,IMOSC_5M_VALUE, IMOSC_5M_VALUE};
    //{SRC_IMOSC, IMOSC_4M_VALUE, SCLK_DIV1, PCLK_DIV1,IMOSC_4M_VALUE, IMOSC_4M_VALUE};
```

**Figure 12 时钟配置结构体**

### 5.5.2 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，使用时需要勾选semihost配置，同时需要在使用打印之前，添加框图所示的代码，并且打印的时候使用printf，例如printf("Hello APT32F173x\n")，Semihost打印信息在Output窗口查看。

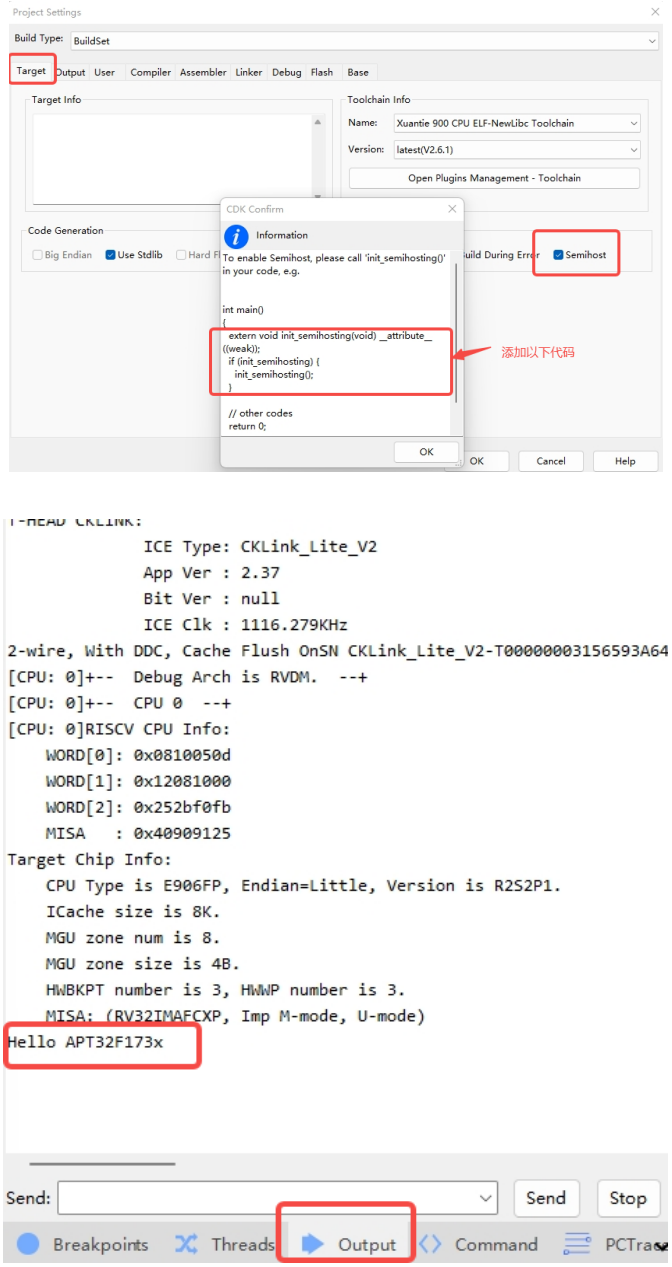



**Figure 13 勾选 Semihost**

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UART1，PA2（TXD）和PA3（RXD）。注意查看MCU 型号是否有UART1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");

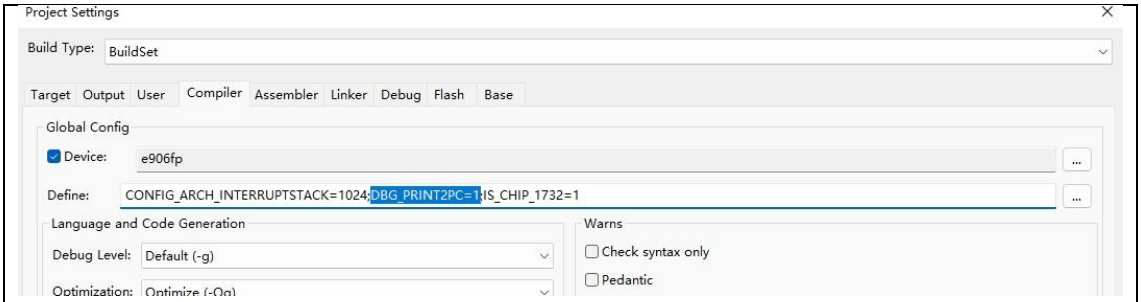



**Figure 14 DBG_PRINT2PC宏**

| 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。<br>所有的宏在 board_config.h 中定义。<br>__attribute__((weak)) void board_init(void)<br>{<br>//console config for print<br>g_tConsole.uart_id = (uint32_t)CONSOLE_IDX;<br>g_tConsole.baudrate = 115200U;<br>g_tConsole.tx.pin = CONSOLE_TXD;<br>g_tConsole.tx.func = CONSOLE_TXD_FUNC;<br>g_tConsole.rx.pin = CONSOLE_RXD;<br>g_tConsole.rx.func = CONSOLE_RXD_FUNC;<br>g_tConsole.uart = (csp_uart_t *)(APB_UART0_BASE + CONSOLE_IDX * 0x1000);<br>console_init(&g_tConsole);<br>} | board_config.c |

### 5.5.3 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架子，特别是某些功能实现和中断强相关的外设，如ADC0。

```
ATTRIBUTE_ISR void adc0_int_handler(void)
{
#if ADC0_INT_HANDLE_EN
    // ISR content ...
    adc_irqhandler(ADC0);
#endif
}
```

**Figure 15 中断处理函数举例**

在上述例子中，adc_irqhandler()函数在驱动代码（adc_demo.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adc0_int_handler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usart0_int_handler ()为例，如果应用中没有用到USART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 usart0_int_handler (void)

2、 srartup.S中将原来usart0_int_handler替换为Default_Handler

- 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。

## 5.6 外设使用

外设使用的示例代码在APT32F173x_demo这个组件中。main()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。

## 5.7 编译工程

点击“Build Project”既可实现工程的编译和连接。

按钮1：工程构建按钮；对工作空间中的活动工程进行构建；

按钮2：工程重新构建按钮；对工作空间中的活动工程进行重新构建。

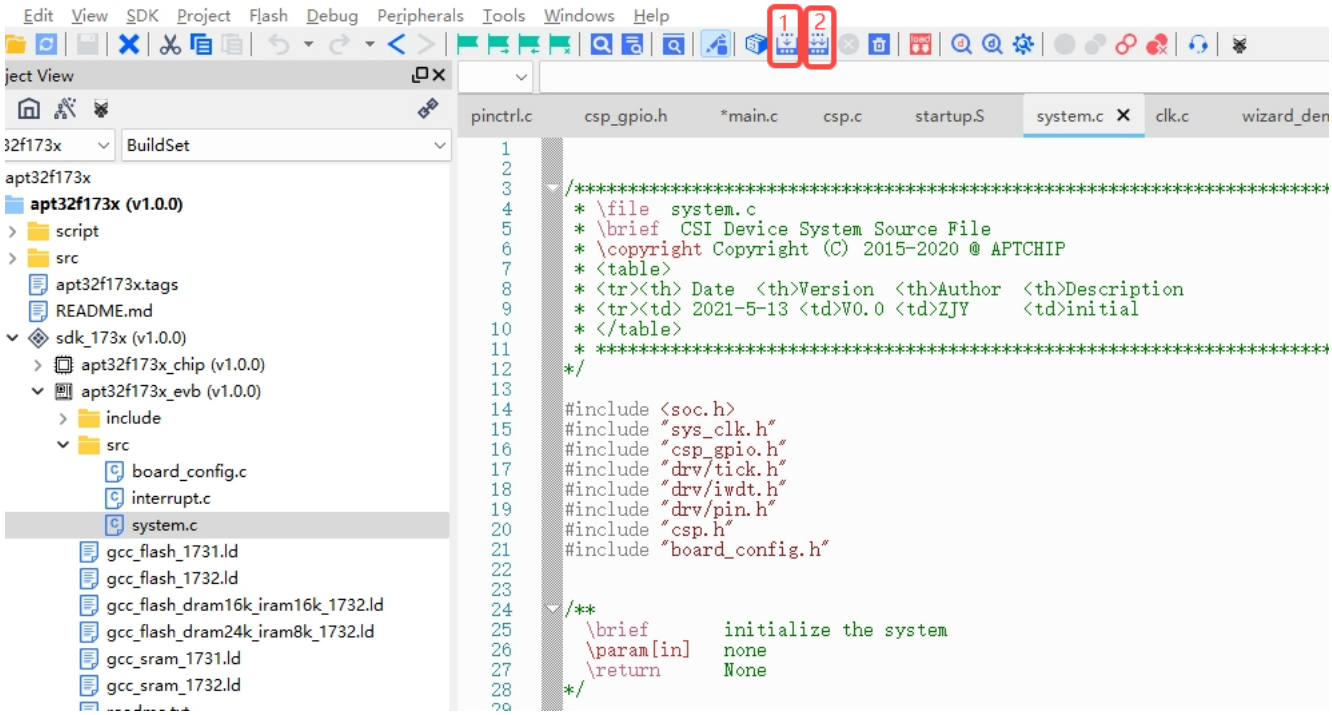



**Figure 16 build**

## 5.8 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. Attach操作，不重新下载代码，不对芯片做任何复位操作，直接进入调试模式，查看芯片当前运行状态下图中的1，2，3分别对应上面三种下载方式的操作菜单。

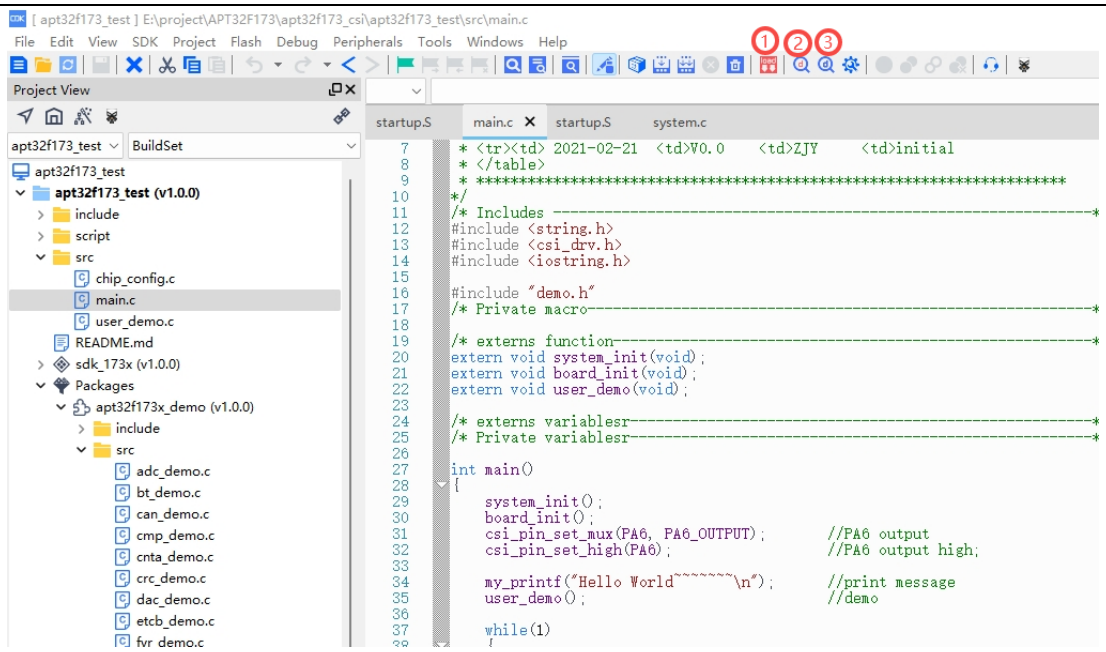



**Figure 17 下载调试的三种选择**

# 6. Appendix

这部分内容用于说明芯片外设示例代码的使用方法，示例代码位于apt32f173x_demo组件下。在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。

## 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，PA0/PA2
csi_gpio_port_dir(GPIOA, wPinMask, GPIO_DIR_OUTPUT);     //端口配置为输出
csi_gpio_port_set_high(GPIOA, wPinMask);                  //输出高
csi_gpio_port_set_low(GPIOA, 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。

1、 参考用法

```
uint32_t wPinMask = PINMASK_PA00 | PINMASK_PA02;          //GPIOA0端口，PA0/PA2
csi_gpio_port_dir(GPIOA, wPinMask, GPIO_DIR_INPUT);       //端口配置为输入
csi_gpio_port_pull_mode(GPIOA, wPinMask, GPIO_PULLNONE);  //无上下拉
csi_gpio_port_pull_mode(GPIOA, wPinMask, GPIO_PULLUP);    //上拉使能
csi_gpio_port_pull_mode(GPIOA, 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，PA0/PA2
csi_gpio_port_dir(GPIOA, wPinMask, GPIO_DIR_INPUT);                 //端口配置为输入
csi_gpio_port_pull_mode(GPIOA, wPinMask, GPIO_PULLUP);              //上拉使能
csi_gpio_port_irq_mode(GPIOA,wPinMask,GPIO_IRQ_FALLING_EDGE);       //下降沿触发中断
csi_gpio_port_irq_enable(GPIOA,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(PA5,PA5_OUTPUT);            //PA5配置为输出
csi_pin_set_high(PA5);                      //输出高
csi_pin_set_low(PA5);                       //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(PA5,PA5_INPUT);             //PA5配置为输入
csi_pin_pull_mode(PA5,GPIO_PULLNONE);       //无上下拉
csi_pin_pull_mode(PA5,GPIO_PULLUP);         //上拉使能
csi_pin_pull_mode(PA5,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(PB2, PB2_INPUT);                          //PB2配置为输入
csi_pin_pull_mode(PB2, GPIO_PULLUP);                      //上拉使能
csi_pin_irq_mode(PB2, EXI_GRP2, GPIO_IRQ_FALLING_EDGE);   //下降沿触发中断，选择中断组2
csi_pin_irq_enable(PB2, ENABLE);                          //PB2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(PB5, IOMAP0_USART0_RX);       //IOMAP GROUP0
csi_pin_set_iomap(PA4, IOMAP1_GPTA0_CHA);       //IOMAP GROUP1
```

详见pin_ioremap_demo函数。

### 6.2 Reliability

#### 6.2.1 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_30);          //VDD掉电到3.0V及以下，触发LVD中断
```

详见lvd_demo函数。

### 6.2.2 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_31);                    //VDD掉电到3.1V及以下，芯片复位
```

详见lvr_demo函数。

### 6.2.3 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。

### 6.2.4 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(PD0, PD0_XIN);
csi_pin_set_mux(PD1, PD1_XOUT);
csi_emosc_enable(8000000);             //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。

### 6.2.5 escm_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(PC14, PC14_SXIN);
csi_pin_set_mux(PC15, PC15_SXOUT);
csi_escm_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_escm_rst();                  //一旦检测到外部晶振失常，系统复位。
csi_escm_disable();              //取消对外部晶振的检测
```

详见escm_demo函数。

### 6.2.6 syscon_cqcr_demo

1、函数位置

reliability_demo.c

2、功能描述

时钟源频率验证功能

3、参考用法

```
uint32_t wCqsrValue = 0;
csi_set_cqcr(CQCR_REFSEL_EM,CQCR_SRCSEL_IM,0x3ff); //参考时钟选择 EM,  源时钟选择控 IM
wCqsrValue = csi_get_cqsr();                       //读取参考源时钟计数值
my_printf("cqsr value =%d",wCqsrValue);
```

详见syscon_cqcr_demo函数。

## 6.3 IWDT

### 6.3.1 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。

#### 6.3.2 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。

### 6.4 WWDT

#### 6.4.1 wwdt_demo

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。

1、 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。

### 6.5 IFC

#### 6.5.1 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, s_wReadBuf, 2);   //从0x0地址读取2个word数据
```

详见ifc_read_demo函数。

### 6.5.2 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, s_wWriteData, 5); //从0x10000000地址写入5个word数据
```

详见ifc_dflash_page_program_demo函数。

### 6.5.3 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, s_wWriteData, 5);   //从0x10000000地址写入5个word数据
```

详见ifc_dflash_page_parallel_program_demo函数。

### 6.5.4 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, s_wWriteData, 5);     //从0x0000F000地址写入5个word数据
```

详见ifc_pflash_page_program_demo函数。

#### 6.5.5 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。

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

详见ifc_program_demo函数。

### 6.6 PM

#### 6.6.1 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。

2、 参考用法

```
csi_pin_set_mux(PB1,PB1_INPUT);                                  //PB1  输入使能
csi_pin_pull_mode(PB1, GPIO_PULLUP);                             // PB1上拉使能
csi_pin_irq_mode(PB1,EXI_GRP1, GPIO_IRQ_FALLING_EDGE);           // PB1下降沿产生中断，选择中断组1
csi_pin_irq_enable(PB1, ENABLE);                                 // PB1 gpio中断使能
csi_pin_vic_irq_enable(EXI_GRP1,ENABLE);                         // PB1vic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

### 6.6.2 lp_lpt_wakeup_deepsleep_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lpt定时唤醒。

3、 参考用法

```
csi_pm_config_wakeup_source(WKUP_LPT, ENABLE);     //设置唤醒源
csi_lpt_timer_init(LPT,LPT_CLK_ISCLK, 500);        //初始化lpt,选用内部超低功耗时钟,定时500ms,默认采用PEND中断
csi_lpt_start(LPT);
 delay_ums(200);
csi_pm_enter_sleep(_LOW_POWER_MODE_);              /进入低功耗模式
```

详见lp_lpt_wakeup_deepsleep_demo函数

## 6.7 BT

### 6.7.1 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, 1000);     // BT0定时1000us
csi_bt_start(BT0);                //启动定时器
```

详见bt_timer_demo函数。

### 6.7.2 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。

3、参考用法

```
csi_pin_set_mux(PA1, PA1_BT0_OUT);              //PA1作为BT1 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(BT1, &tPwmCfg);                 //初始化BT1 PWM输出
```

详见bt_pwm_demo函数。

### 6.7.3 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

```
//PB1配置为外部中断功能，选择中断组1
......
csi_exi_set_evtrg(EXI_TRGOUT1, TRGSRC_EXI1, 0);          //EXI1触发EXI_TRGOUT1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_TRGOUT1)和源事件(BT0 SYNCIN0)：EXI_TRGOUT1 -> ETCB -> BT0 SYNCIN0
......
csi_etb_init();                                           //ETCB初始化
```

详见bt_sync_start_demo函数。

### 6.7.4 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

```
//PB1配置为外部中断功能，选择中断组1
......
csi_exi_set_evtrg(1, TRGSRC_EXI1, 0);                    //EXI1 (PB1)触发EXI_TRGOUT1
csi_bt_timer_init(BT0,1000);                                    //BT0定时1ms
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_TRGOUT1)和源事件(BT0 SYNCIN1)：EXI_TRGOUT1 -> ETCB -> BT0 SYNCIN1
......
csi_etb_init(); //ETCB初始化
```

详见bt_sync_stop_demo函数。

### 6.7.5 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

```
//PB1配置为外部中断功能，选择中断组1
......
csi_exi_set_evtrg(1, TRGSRC_EXI1, 1);                    //EXI1 (PB1)触发EXI_TRGOUT1
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_TRGOUT1)和源事件(BT0 SYNCIN2)：EXI_TRGOUT1 -> ETCB -> BT0 SYNCIN2
......
csi_etb_init(); //ETCB初始化
```

详见bt_sync_count_demo函数。

### 6.7.6 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(BT0,10000);                                    //BT0定时10ms，默认连续计数模式
csi_bt_set_evtrg(BT0, BT_TRGSRC_PEND,ENABLE);                    //BT0 PEND事件触发输出
csi_bt_start(BT0);                                               //启动BT0定时器
......
csi_bt_pwm_init(BT1, &tPwmCfg);                                  //初始化BT1 PWM输出
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 CNTA

### 6.8.1 cnta_timer_demo

1、函数位置

cnta _demo.c

2、功能描述

CA0基本定时功能示例。

3、参考用法

```
......
csi_cnta_timer_init(CA0,&tTimerCfg);   //初始化CountA
csi_cnta_start(CA0);                   //启动CountA
```

详见cnta _timer_demo函数。

### 6.8.2 cnta _pwm_demo

1、函数位置

cnta_demo.c

2、功能描述

CA0 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。

3、参考用法

```
csi_pin_set_mux(PA10,PA10_CNTA_BUZ);          //PA10作为CA0 PWM输出引脚
......
//初始化tPwmCfg，设置PWM电平参数，周期和占空比
......
```

```
csi_cnta_pwm_init(CA0,&tPwmCfg);                    //初始化CountA
csi_cnta_start(CA0);                                //启动CountA
```

详见cnta_pwm_demo函数。

### 6.8.3 cnta_envelope_demo

1、函数位置

cnta_demo.c

2、功能描述

CounterA 和BT0搭配包络输出PWM示例

3、参考用法

```
//PB1配置为外部中断功能，选择中断组1
csi_pin_set_mux(PA0,    PA0_BT0_OUT);        //PA0    作为BT0 PWM输出引脚
......
csi_bt_pwm_init(BT0, &tBTPwmCfg);            //初始化BT0 PWM输出
csi_bt_start(BT0);                           //启动BT0

csi_pin_set_mux(PA10,PA10_CNTA_BUZ);     //PA10作为CA0 PWM输出引脚
......
csi_cnta_pwm_init(CA0,&tPwmCfg);             //初始化CountA
csi_cnta_start(CA0);                         //启动CountA
```

详见cnta_envelope_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_sync_demo0

1、函数位置

gpta_demo.c

2、功能描述

GPTA捕获功能示例。sync2 sync3不区分，实现4次捕获，由外部中断EXI事件通过ETCB模块触发GPTA SYNCIN3，产生捕获。

3、参考用法

```
//PA1配置为外部中断功能，选择中断组16
......
csi_exi_set_evtrg(5, TRGSRC_EXI16, 1);      //EXI16触发EXI_TRGOUT5
//ETCB配置，设置目标事件(EXI_TRGOUT5)和源事件(GPTA0_SYNCIN3)：EXI_TRGOUT5 -> ETCB -> GPTA0_SYNCIN3
......
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_SYNCEN3, 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_sync_demo0函数。

### 6.9.3 gpta_capture_sync_demo1

1、函数位置

gpta_demo.c

2、功能描述

GPTA捕获功能示例。sync2 sync3区分，实现2次捕获，由外部中断EXI事件通过ETCB模块触发GPTA SYNCIN3，产生捕获。

3、参考用法

```
//PA3配置为外部中断功能，选择中断组3
......
csi_exi_set_evtrg(0, TRGSRC_EXI3, 1);       //EXI3触发EXI_TRGOUT0
//PA3配置为外部扩展口中断功能，选择中断组16
......
csi_exi_set_evtrg(5, TRGSRC_EXI16, 1);      //EXI16触发EXI_TRGOUT5
//ETCB配置，设置目标事件(EXI_TRGOUT5)和源事件(GPTA0_SYNCIN3)：EXI_TRGOUT5 -> ETCB -> GPTA0_SYNCIN3
......
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);
c si_gpta_set_sync(GPTA0, GPTA_TRG_SYNCEN2, GPTA_TRG_CONTINU, GPTA_AUTO_REARM_ZRO);
```

```
//使能SYNCIN2外部触发
csi_gpta_set_sync(GPTA0, GPTA_TRG_SYNCEN3, GPTA_TRG_CONTINU, GPTA_AUTO_REARM_ZRO);
//使能SYNCIN3外部触发
csi_gpta_start(GPTA0);                    //启动GPTA0
```

详见gpta_ capture_sync_demo1函数。

### 6.9.4 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_change_ch_duty()实现。

3、参考用法

```
csi_pin_set_mux(PA0,PA0_GPT_CHA);       //PA0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%
```

详见gpta_pwm_demo函数。

### 6.9.5 gpta_pwm_waveform_demo

1、函数位置

gpta_demo.c

2、功能描述

GGPTA波形强制输出demo，包含一次性强制性输出和连续强制。

3、参考用法

```
csi_pin_set_mux(PA0,PA0_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);
//一次强制输出
csi_gpta_onetimesoftware_output(GPTA0,GPTA_OSTSF1,GPTA_HI);
//连续强制输出
csi_gpta_aqcsfload_config(GPTA0, GPTA_AQCSF_NOW);
csi_gpta_continuous_software_waveform(GPTA0, GPTA_CHANNEL_1, GPTA_AQCSF_L);
csi_gpta_continuous_software_waveform(GPTA0, GPTA_CHANNEL_1, GPTA_AQCSF_NONE);
```

详见gpta_pwm_waveform_demo函数。

### 6.9.6 gpta_reglk_demo

1、函数位置

gpta_demo.c

2、功能描述

GPTA链接代码实例。通过GPTA1链接GPTA0,实现波形的输出。

3、参考用法

```
csi_pin_set_mux(PA0,PA0_GPT_CHA);        //PA0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);
......
//链接配置，GPTA1链接GPTA0
csi_gpta_reglk_config(GPTA0,&FEGLKcfg);

csi_pin_set_mux(PA2, PA2_GPTA1_CHA);
......
csi_gpta_wave_init(GPTA1, &tPwmCfg);
......
csi_gpta_channel_config(GPTA0,&tGptachannelCfg,GPTA_CHANNEL_1);
csi_gpta_start(GPTA1);
//链接,改变GPTA1的波形，GPTA0也会随着改变
csi_gpta_change_ch_duty(GPTA1,GPTA_COMPA, 20);
```

详见gpta_reglk_demo函数。

## 6.10 GPTB

### 6.10.1 gptb_timer_demo

1、函数位置

gptb_demo.c

2、功能描述

GPTB基本定时功能示例。

3、参考用法

```
csi_gptb_timer_init(GPTB0, 10000);          //初始化GPTB0，定时10000us
csi_gptb_start(GPTB0);                      //启动定时器
```

详见gptb_timer_demo函数。

### 6.10.2 gptb_capture_demo

1、函数位置

gptb_demo.c

4、 功能描述

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

5、 参考用法

```
//PA1配置为外部中断功能，选择中断组1
......
csi_exi_set_evtrg(1, TRGSRC_EXI1, 1);
//ETCB配置，设置目标事件(EXI TRGOUT1)和源事件(GPTB0 SYNCIN2)：EXI TRGOUT1 -> ETCB -> GPTB0 SYNCIN2
......
csi_etb_init();
......
//初始化tPwmCfg，设置GPTB0的捕获参数
......
csi_gptb_capture_init(GPTB0, &tPwmCfg);
csi_gptb _set_sync (GPTB0, GPTB_TRGIN_SYNCEN2, GPTB_TRG_CONTINU,GPTB_AUTO_REARM_ZRO);
csi_gptb_start(GPTB0);                      //启动EPT0
```

详见gptb_capture_demo函数。

### 6.10.3 gptb_pwm_demo

1、 函数位置

gptb_demo.c

2、功能描述

GPTB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_gptb_prdr_cmp_update()实现。

3、参考用法

```
csi_pin_set_mux(PC13, PC13_GPTB0_CHAX);            //PC13作为PWM输出引脚
......
//初始化tPwmCfg，设置PWM周期和占空比
......
csi_gptb_wave_init(GPTB0, &tPwmCfg);
......
//初始化channel，设置PWM波形输出
......
csi_gptb_channel_config(GPTB0, &channel,   GPTB_CHANNEL_1);
csi_gptb_start(GPTB0)
csi_gptb_change_ch_duty(GPTB0,GPTB_COMPA, 20);            //修改PWM1占空比为50%
csi_gptb_prdr_cmp_update(GPTB0,GPTB_COMPA,1200,800);    //修改PWM1周期为12000，比较值为800
```

详见gptb_pwm_demo函数。

### 6.10.4 gptb_pwm_dz_demo

1、 函数位置

gptb_demo.c

2、功能描述

GPTB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(PC13, PC13_GPTB0_CHAX);            //PC13作为PWM输出引脚
......
//初始化tPwmCfg，设置PWM周期和占空比
......
csi_gptb_wave_init(GPTB0, &tPwmCfg);
......
//初始化channel，设置PWM波形输出
......
csi_gptb_channel_config(GPTB0, &channel,   GPTB_CHANNEL_1);
......
//初始化tGptbDeadZoneTime，设置死区时间与边沿
......
csi_gptb_dz_config(GPTB0,&tGptbDeadZoneTime);
csi_gptb_start(GPTB0);
```

详见gptb_pwm_dz_demo函数。

### 6.10.5 gptb _pwm_dz_em_demo

1、 函数位置

gptb _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(PC13, PC13_GPTB0_CHAX);            //PC13作为PWM输出引脚
......
//初始化tPwmCfg，设置PWM周期和占空比
......
csi_ gptb _wave_init(GPTB0, &tPwmCfg);
......
//初始化channel，设置PWM波形输出
......
csi_ gptb _channel_config(GPTB0, &channel,  GPTB _CHANNEL_1);
......
//初始化tGptbDeadZoneTime，设置死区时间与边沿
......
csi_ gptb _dz_config(GPTB0,&tGptbDeadZoneTime);
......
//初始化tGptbEmergencyCfg，设置紧急模式的输入源与处理方式
......
csi_gptb_emergency_config(GPTB0,&tGptbEmergencyCfg);
csi_gptb_start(GPTB0);
```

详见gpt _pwm_dz_em_demo函数。

## 6.11 RTC

### 6.11.1 rtc_set_time_demo

1、 函数位置

rtc_demo.c

2、 功能描述

设置rtc时间的示例代码：包括时钟源、时间模式、时间设置，当前时间回读函数。

3、参考用法

```
......
//初始化tRtcConfig，选择时钟源，选择时间模式
......
csi_rtc_init(RTC, &tRtcConfig);          //初始化RTC参数配置
......
csi_rtc_set_time(RTC, &tRtcTime);      //设置时间
csi_rtc_start(RTC);                          //RTC开始计时
```

```
csi_rtc_get_time(RTC,  &tRtcTimeRdbk);  //回读当前时间
......
csi_rtc_change_fmt(RTC, RTC_12FMT);   //修改时间模式为12小时制
csi_rtc_get_time(RTC,  &tRtcTimeRdbk);  //回读当前时间
```

详见rtc_set_time_demo函数。

### 6.11.2 rtc_alarm_demo

1、 函数位置

rtc_demo.c

2、 功能描述

设置rtc时间的示例代码：包括时钟源、时间模式、时间设置，当前时间回读函数。

3、参考用法

```
......
//初始化tRtcConfig，选择时钟源，选择时间模式
......
csi_rtc_init(RTC, &tRtcConfig);          //初始化RTC参数配置
......
csi_rtc_set_time(RTC, &tRtcTime);      //设置时间
csi_rtc_start(RTC);                    //RTC开始计时
csi_rtc_set_alarm(RTC, RTC_ALMA, tAlmA.byAlmMode, &tAlmTime);  //设置闹钟A
......
//如果闹钟时间没有到，每秒打印一次当前时间和距离闹钟的时间
......
csi_rtc_cancel_alarm(RTC, RTC_ALMA);  //取消闹钟，保持原闹钟时间
```

详见rtc_alarm_demo函数。

### 6.11.3 rtc_timer_demo

1、 函数位置

rtc_demo.c

2、 功能描述

如何将RTC当做一个简单timer来使用

3、参考用法

```
......
//初始化tRtcConfig，选择时钟源，选择时间模式
csi_rtc_init(RTC, &tRtcConfig);                    //初始化RTC参数配置
ccsi_rtc_start_as_timer(RTC, RTC_TIMER_1MIN); //每1s进一次中断
csi_rtc_start(RTC);                                //RTC开始计时
```

详见rtc_timer_demo函数。

### 6.11.4 rtc_trgev_demo

1、 函数位置

rtc_demo.c

2、 功能描述

RTC通过ETCB触发demo，可根据需求，配置相应的触发目标

3、参考用法

```
......
//初始化tRtcConfig，选择时钟源，选择时间模式
......
csi_rtc_init(RTC, &tRtcConfig);          //初始化RTC参数配置
......
csi_rtc_start_as_timer(RTC, RTC_TIMER_1S);     //每1s进一次中断
csi_rtc_int_enable(RTC, RTC_INT_CPRD , DISABLE);   //不需要中断的话，可以关掉
csi_rtc_start(RTC);                                //RTC开始计时
csi_rtc_set_evtrg(RTC, 0, RTC_TRGOUT_CPRD, 2);     //RTC TRGEV0  每两秒钟输出一次trigger event
```

详见rtc_trgev_demo函数。

## 6.12 ADC

### 6.12.1 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(PC13, PC13_ADC_INA0); // PC13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。

### 6.12.2 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(PC13, PC13_ADC_INA0); // PC13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。

### 6.12.3 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(PC13, PC13_ADC_INA0); // PC13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。

### 6.12.4 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(PC13, PC13_ADC_INA0); // PC13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。

## 6.13 FVR

### 6.13.1 fvr_output_demo

1、 函数位置

fvr_demo.c

2、 功能描述

配置FVR电平，并通过管脚输出3、参考用法

```
csi_pin_set_mux(PB7, PB7_FVROUT); //  设置PB7为FVR输出
......
//配置FVR时钟
......
csi_fvr_start(FVR);
```

详见fvr_output_demo函数。

### 6.13.2 fvr_buf_demo

1、 函数位置

fvr_demo.c

2、 功能描述

配置BUF(intervref)电平，并通过管脚输出。

3、参考用法

```
 soc_clk_enable(FVR_SYS_CLK); //  配置FVR时钟
csi_pin_set_mux(PD4, PD4_INPUT); //  设置BUF输入管脚PD4
csi_pin_set_mux(PB8, PB8_BUF); //  设置BUF输出管脚PB8
csi_fvr_buf_init(FVR,BUFLVL_INPUT); //  设置buf的输入源
csi_fvr_start(FVR);
```

详见fvr_buf_demo函数。

## 6.14 DAC

### 6.14.1 dac_demo

1、 函数位置

dac_demo.c

2、 功能描述

配置DAC的初始电平码值，设置时钟分频，配置DAC中断及触发模式

```
......
csi_dac_init(DAC0, &tDacConfig); // 配置DAC 分频、参考电平选择等
csi_dac_en(DAC0);
......
// 使能中断
// 开启DAC触发
csi_dac_start(DAC0);
```

详见dac_demo函数。

## 6.15 CRC

### 6.15.1 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。

## 6.16 ETCB

### 6.16.1 etcb_one_trg_one_demo

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LPT SYNC PORT0，即LPT启动。

3、参考用法

```
//PA1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_TRGIN_SYNCEN0, BT_TRG_ONCE, BT_TRG_AUTOAREARM);
//外部触发BT0启动(SYNCIN0),使用PEND事件REARM
csi_bt_set_evtrg(BT0, BT_TRGSRC_PEND, ENABLE); //BT0 PEND事件触发输出
csi_lpt_start_sync(LPT,LPT_CLK_PCLK_DIV4,50);
csi_lpt_set_sync(LPT, LPT_TRG_SYNCIN0, LPT_SYNC_ONCE, ENABLE); //外部触发LPT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(LPT SYNCIN0)：BT0 PEND -> ETCB -> LPT SYNCIN0
......
csi_etb_init();
......
```

详见etcb_one_trg_one_demo函数。

### 6.16.2 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。外部中断EXI事件通过ETCB模块触发BT0 SYNC PORT0、BT1 SYNC PORT0、BT2 SYNC PORT0，即BT0、BT1、BT2启动；

3、参考用法

```
//PA1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, 200);
```

```
csi_bt_set_sync(BT0, BT_TRGIN_SYNCEN0, BT_TRG_ONCE, BT_TRG_AUTOAREARM);
//外部触发BT0启动(SYNCIN0),使用PEND事件REARM
csi_bt_start_sync(BT1, 200);
csi_bt_set_sync(BT1, BT_TRGIN_SYNCEN0, BT_TRG_ONCE, BT_TRG_AUTOAREARM);
//外部触发BT0启动(SYNCIN0),使用PEND事件REARM
csi_bt_start_sync(BT2, 200);
csi_bt_set_sync(BT2, BT_TRGIN_SYNCEN0, BT_TRG_ONCE, BT_TRG_AUTOAREARM);
//外部触发BT0启动(SYNCIN0),使用PEND事件REARM


......
csi_etb_init();
......
```

详见etcb_one_trg_more_demo函数。

## 6.17 UART

### 6.17.1 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。

### 6.17.2 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。

### 6.17.3 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。

### 6.17.4 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

### 6.17.5 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。

#### 6.17.6 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。

### 6.18 USART

#### 6.18.1 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。

### 6.18.2 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。

### 6.18.3 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。

### 6.18.4 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。

### 6.18.5 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。

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。

### 6.18.6 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。

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。

## 6.19 CAN

### 6.19.1 can_send_demo

1、函数位置

can_demo.c

2、功能描述

can 发送demo，开启状态中断，禁止报文通道中断，可通过TXOK状态中断判断发送是否完成。

3、参考用法

```
//初始化tCanConfig，设置can基本参数：波特率500k，时钟选择，状态中断全部使能，报文通道中断,Chnl1~32
......
csi_can_init(CAN0, &tCanConfig);            //初始化can
......
csi_can_tx_config(CAN0, i, &tCanTxCfg);   //CAN TX 配置
csi_can_open(CAN0);
csi_can_chnl_send(CAN0, CAN_CH1, 6);   //通道1发送
csi_can_chnl_send(CAN0, CAN_CH2, 7); //通道2发送
//更新报文中数据(DAR/DBR)
......
csi_can_set_data_a(CAN0, CAN_CHX, byData); //updata 通道1或2 DAR 寄存器
......
csi_can_set_ir(CAN0, CAN_CH1, &tCanTxCfg.tIr);   //配置报文识别符
csi_can_chnl_send(CAN0, CAN_CHX, 8);            //通道1或2发送
......
```

详见can_send_demo函数。

### 6.19.2 can_send_int_demo

1、函数位置

can_demo.c

2、功能描述

can 发送demo，开启状态中断和报文通道中断，可通过具体的报文通道发送中断判断发送完成。

3、参考用法

```
//初始化tCanConfig，设置can基本参数：波特率500k，时钟选择，状态中断全部使能，报文通道中断,Chnl1~32
......
csi_can_init(CAN0, &tCanConfig);            //初始化can
......
csi_can_tx_config(CAN0, i, &tCanTxCfg);   //CAN TX 配置
csi_can_open(CAN0);
```

```
csi_can_chnl_send(CAN0, CAN_CHX 7); //通道1或2发送
//更新报文中数据(DAR/DBR)
......
csi_can_set_data_a(CAN0, CAN_CHX, byData); //updata 通道1或2 DAR 寄存器
......
csi_can_set_ir(CAN0, CAN_CH1, &tCanTxCfg.tIr);   //配置报文识别符
csi_can_chnl_send(CAN0, CAN_CHX, 8);              //通道1或2发送
......
//可通过查询发送通道中断、状态中断的消息,除通道中断消息 ，清除状态中断消息
```

详见can_send_int_demo函数。

### 6.19.3 can_recv_demo

1、函数位置

can_demo.c

2、功能描述

can 接收demo，接收报文通道不配置为FIFO缓存的报文接收，接收过滤掩码不使能(ID不过滤全匹配)；接收报文根据报文匹配的ID接收到对应的报文通道中。

3、参考用法

```
//初始化tCanConfig，设置can基本参数：波特率500k，时钟选择，状态中断全部使能，报文通道中断,Chnl1~32
......
csi_can_init(CAN0, &tCanConfig);              //初始化can
......
csi_can_rx_config(CAN0, i, &tCanRxCfg);   //CAN RX 配置
csi_can_recv_init(&tCanRecv[0], CAN_CH1, 3); //接收数据结构体
csi_can_open(CAN0);
......
csi_can_chnl_read(CAN0,byRxBuf, i);         //通道1或2读取
```

详见can_recv_demo函数。

### 6.19.4 can_recv_fifo_demo

1、函数位置

can_demo.c

2、功能描述

can 接收demo，接收报文通道配置为FIFO缓存的报文接收，接收过滤掩码不使能(ID不过滤全匹配)；接收报文根据报文匹配的ID接收到对应的报文通道中。

3、参考用法

```
//初始化tCanConfig，设置can基本参数：波特率500k，时钟选择，状态中断全部使能，报文通道中断,Chnl1~32
......
csi_can_init(CAN0, &tCanConfig);              //初始化can
```

```
.....
csi_can_rx_config(CAN0, i, &tCanRxCfg);   //CAN RX 配置
csi_can_recv_init(&tCanRecv[0], CAN_CH1, 3); //接收数据结构体
csi_can_open(CAN0);
.....
csi_can_chnl_read(CAN0,byRxBuf, i);          //通道1或2读取
```

详见can_recv_fifo_demo函数。

## 6.20 SPI

### 6.20.1 spi_master_send_demo

1、函数位置

spi_demo.c

2、功能描述

Spi 主机发送一串数据,TX使用轮询。

3、参考用法

```
//端口配置
.....
//初始化tSpiConfig，主机模式、clk空闲电平、帧数据长度、通讯速率1Mbps、初始配置无中断、发送轮询模式
.....
csi_spi_init(SPI0,&tSpiConfig);                //初始化spi
csi_spi_send(SPI0,byData,8);                   //发送数据
```

详见spi_master_send_demo函数。

### 6.20.2 spi_master_send_int_demo

1、函数位置

spi_demo.c

2、功能描述

spi 主机发送一串数据,TX使用中断。

3、参考用法

```
//端口配置
.....
//初始化tSpiConfig，主机模式、clk空闲电平、帧数据长度、通讯速率1Mbps、发送使用中断、发送使用中断模式
.....
csi_spi_init(SPI0,&tSpiConfig);                //初始化spi
csi_spi_send(SPI0,byData,8);                   //发送数据
.....                                          //等待发送完成状态
```

详见spi_master_send_int_demo函数。

### 6.20.3 spi_master_send_receive_demo

1、函数位置

spi_demo.c

2、功能描述

spi 主机收发一串数据，收发使用轮询。

3、参考用法

```
//端口配置
......
//初始化tSpiConfig，主机模式、clk空闲电平、帧数据长度、通讯速率1Mbps、初始配置无中断、接收轮询模式
......
csi_spi_init(SPI0,&tSpiConfig);                    //初始化spi
csi_spi_send_receive(SPI0, bySendData, byRecvData, 16);                    //接收数据
```

详见spi_master_send_receive_demo函数。

### 6.20.4 spi_slave_receive_int_demo

1、函数位置

spi_demo.c

2、功能描述

spi 从机接收一串数据，RX使用中断。

3、参考用法

```
//端口配置
......
//初始化tSpiConfig，作为从机、clk空闲电平、帧数据长度、作从机时，通讯速率取决于主机、初始配置接收中断、接收使用中断模式
......
csi_spi_init(SPI0,&tSpiConfig);                    //初始化spi
csi_spi_receive(SPI0,byRecvData,16);                    //接收数据
......                                                  //等待接收完成状态
```

详见spi_slave_receive_int_demo函数。

### 6.20.5 spi_slave_send_receive_demo

1、函数位置

spi_demo.c

2、功能描述

spi 从机收发一串数据，收发使用轮询。

3、参考用法

```
//端口配置
......
//初始化tSpiConfig，作为从机、clk空闲电平、帧数据长度、作从机时，通讯速率取决于主机、初始配置接收中断
......
csi_spi_init(SPI0,&tSpiConfig);                    //初始化spi
csi_spi_receive_slave(SPI0);                       //接收数据
csi_spi_send_slave(SPI0, byReceData);              //发送数据
```

详见spi_slave_send_receive_demo函数。

### 6.20.6 spi_etcb_dma_send_demo

1、函数位置

spi_demo.c

2、功能描述

spi DMA发送示例代码，使用DMA循环发送20字节数据。

3、参考用法

```
//初始化tDmaConfig，
.......
csi_dma_soft_rst(DMA0);
csi_dma_ch_init(DMA0, 0, &tDmaConfig);   //初始化DMA
//初始化tEtbConfig，
.......
csi_etb_init();
csi_etb_ch_config(ETB_CH20, &tEtbConfig);   //初始化ETCB DMA ETB CHANNEL > ETB_CH19_ID
//spi端口配置
......
//初始化tSpiConfig，主机模式、clk空闲电平、帧数据长度、通讯速率1Mbps、初始配置无中断
......
csi_spi_init(SPI0,&tSpiConfig);                  //初始化spi
csi_spi_send_dma(SPI0, (void *)bySdData, sizeof(bySdData), DMA0, 0);              //DMA发送数据
......                                             //等待直到dma传输完成
```

详见spi_etcb_dma_send_demo函数。

### 6.20.7 spi_etcb_dma_send_receive_demo

1、函数位置

spi_demo.c

2、功能描述

spi DMA发送接收示例代码，使用DMA循环发送20字节数据。

3、参考用法

```
//初始化tDmaConfig，
.......
```

```
csi_dma_soft_rst(DMA0);
csi_dma_ch_init(DMA0, byDMAChnlSend, &tDmaConfigSend); //初始化DMA
csi_dma_ch_init(DMA0, byDMAChnlRecv, &tDmaConfigRecv); //初始化DMA
//初始化tEtbConfig，
.......
csi_etb_init();
csi_etb_ch_config(ETB_CH20, &tEtbConfig);   //初始化ETCB DMA ETB CHANNEL > ETB_CH19_ID
csi_etb_ch_config(ETB_CH21, &tEtbConfig);   //初始化ETCB DMA ETB CHANNEL > ETB_CH19_ID

//spi端口配置
......
//初始化tSpiConfig，主机模式、clk空闲电平、帧数据长度、通讯速率1Mbps、初始配置无中断
......
csi_spi_init(SPI0,&tSpiConfig);                 //初始化spi
csi_spi_recv_dma(SPI0, (void *)byDesBuf, sizeof(byDesBuf), DMA0, 1);          //DMA接收数据
csi_spi_send_dma(SPI0, (void *)bySrcBuf, sizeof(bySrcBuf), DMA0, 0);          //DMA发送数据
......                                          //等待直到dma发送完成
......                                          //等待直到dma接收完成
```

详见spi_etcb_dma_send_receive_demo函数。

## 6.21 IIC

### 6.21.1 iic_master_eeprom_demo

1、函数位置

iic_demo.c

2、功能描述

iic主机读写EEPROM功能。

3、参考用法

```
//iic端口配置
......
//初始化g_tIicMasterCfg，设置主机地址模式 7/10 bit、使能重复起始位、设置主机速度模式、使能中断等
......
csi_iic_master_init(I2C0,&g_tIicMasterCfg);          //初始化iic主机
csi_iic_write_nbyte(I2C0,0xa0,0x0001,2,&data[0],9);      //写数据
csi_iic_read_nbyte(I2C0,0xa0,0x0001,2,&data1[0],9);      //读数据
```

详见iic_master_eeprom_demo函数。

### 6.21.2 iic_master_demo

1、函数位置

iic_demo.c

2、功能描述

iic主机读功能。

3、参考用法

```
//iic端口配置
......
//初始化g_tlicMasterCfg，设置主机地址模式 7/10 bit、使能重复起始位、设置主机速度模式、使能中断等
......
csi_iic_master_init(I2C0,&g_tlicMasterCfg);              //初始化iic主机
csi_iic_read_nbyte(I2C0,0xa0,0x01,1,&data1[0],19);       //读数据
```

详见iic_master_demo函数。

### 6.21.3 iic_slave_demo

1、函数位置

iic_demo.c

2、功能描述

Iic从机功能。作为从机时需要在IIC中断里调用 i2c_irqhandler(I2C0) 函数

3、参考用法

```
//iic端口配置
......
//初始化g_tlicSlaveCfg，设置从机地址模式 7/10 bit、使能重复起始位、设置主机速度模式、使能中断等
......
csi_iic_set_slave_buffer(g_byWriteBuffer,32,g_bySendBuffer,32) //从机就是数组和发送数组设置
csi_iic_slave_init(I2C0,&g_tlicSlaveCfg);                //初始化从机
```

详见iic_slave_demo函数。

### 6.21.4 iic_multi_slave_address_demo

1、函数位置

iic_demo.c

2、功能描述

iic 从机功能。作为从机时需要在IIC中断里调用 i2c_irqhandler(I2C0) 函数

3、参考用法

```
//iic端口配置
......
//初始化g_tlicSlaveCfg，设置从机地址模式 7/10 bit、使能重复起始位、设置主机速度模式、使能中断等
......
csi_iic_set_slave_buffer(g_byWriteBuffer,32,g_bySendBuffer,32) //从机就是数组和发送数组设置
csi_iic_slave_init(I2C0,&g_tlicSlaveCfg);                //初始化从机
csi_iic_qualmode_set(I2C0,I2C_QUAL_EXTEND);   //QUALMODE=1,地址扩展模式
```

```
csi_iic_slvqual_set(I2C0,maskaddr);                    //配置SLVQUAL
```

详见iic_multi_slave_address_demo函数。

### 6.21.5 iic_dma_tx_demo

1、函数位置

iic_demo.c

2、功能描述

iic 主机 DMA发送功能。

3、参考用法

```
//iic端口配置
......
csi_etb_init();                //初始化从机
......
//初始化tEtbConfig
csi_etb_ch_config(ETB_CH21, &tEtbConfig);          //初始化ETB，DMA ETB CHANNEL > ETB_CH19_ID
......
//初始化tDmaConfig
csi_dma_ch_init(DMA0, DMA_CH0, &tDmaConfig);  //初始化DMA
csi_dma_ch_start(DMA0, DMA_CH0, (void *)g_wTxBuff, (void *)0x400A0010,9,1)
//初始化g_tIicMasterCfg，设置主机地址模式 7/10 bit、使能重复起始位、设置主机速度模式、使能中断等
......
csi_iic_master_init(I2C0,&g_tIicMasterCfg);                    //主机初始化
......
csi_iic_enable(I2C0);
```

详见iic_dma_tx_demo函数。

### 6.21.6 iic_dma_rx_demo

1、函数位置

iic_demo.c

2、功能描述

iic 主机 DMA接收功能。

3、参考用法

```
//iic端口配置
......
csi_etb_init();                //初始化从机
......
//初始化tEtbConfig
csi_etb_ch_config(ETB_CH21, &tEtbConfig);          //初始化ETB，DMA ETB CHANNEL > ETB_CH19_ID
......
//初始化tDmaConfig
```

```
csi_dma_ch_init(DMA0, DMA_CH0, &tDmaConfig);  //初始化DMA
csi_dma_ch_start(DMA0, DMA_CH0, (void *)g_wTxBuff, (void *)0x400A0010,9,1)
//初始化g_tIicMasterCfg，设置主机地址模式 7/10 bit、使能重复起始位、设置主机速度模式、使能中断等
......
csi_iic_master_init(I2C0,&g_tIicMasterCfg);                    //主机初始化
......
csi_iic_read_nbyte_dma(I2C0,0xa0,0x0001,2,&g_bRxBuff[0],9);
```

详见iic_dma_rx_demo函数。

## 6.22 LED

### 6.22.1 led_demo

1、函数位置

led_demo.c

2、功能描述

LED示例代码。

3、参考用法

```
//led端口配置
......
//初始化ptLedCfg，COM0~3打开、LED显示亮度50%、显示周期时间、Non-Overlap时间
......
csi_led_init(LED, &ptLedCfg);                    //初始化led
//根据byDisplayStatus进行led控制
//display status 0:正常控制，依次显示1，2，3，4
//display status 1:闪烁控制，关闭COM0，依次显示2，3，4
//display status 2:闪烁控制，关闭COM0/1，依次显示3，4
//display status 3:闪烁控制，关闭COM0/1/2，显示4
```

详见led_demo函数。

## 6.23 SIO

### 6.23.1 sio_led_rgb_demo

1、函数位置

sio_demo.c

2、功能描述

sio 驱动RGB LED(ws2812), RGB DATA = 24bit; 驱动数据输出排列方式:GRB。

3、参考用法

```
//sio端口配置
......
```

```
//初始化tSioTxCfg，Dx 对象序列长度、发送数据缓存长度、一次发送总的数据长度、SIO空闲时刻IO管脚输出电平等
......
csi_sio_tx_init(SIO0, &tSioTxCfg);              //初始化sio
led_rgb_display(byDipData, 8);                  //驱动led显示
```

详见sio_led_rgb_demo函数。

### 6.23.2 sio_led_rgb_int_demo

1、函数位置

sio_demo.c

2、功能描述

sio 驱动RGB LED(ws2812), RGB DATA = 24bit; 驱动数据输出排列方式:GRB。

3、参考用法

```
//sio端口配置
......
//初始化tSioTxCfg，Dx 对象序列长度、发送数据缓存长度、一次发送总的数据长度、SIO空闲时刻IO管脚输出电平等
......
csi_sio_tx_init(SIO0, &tSioTxCfg);              //初始化sio
set_led_rgb_store(byDipDataEnd,byCount);
csi_sio_send(SIO0, byDipDataEnd, 24);                   //驱动led显示
```

详见sio_led_rgb_int_demo函数。

### 6.23.3 sio_led_rgb_send_dma_demo

1、函数位置

sio_demo.c

2、功能描述

sio 驱动RGB LED(ws2812), RGB DATA = 24bit; 驱动数据输出排列方式:GRB。

3、参考用法

```
//sio端口配置
......
//初始化tEtbConfig、初始化tDmaConfig
.......
csi_etb_init();                 //初始化etcb
csi_etb_ch_config(ETB_CH21, &tEtbConfig);                   //配置etcb
csi_dma_ch_init(DMA0, DMA_CH1, &tDmaConfig);                //初始化DMA
//初始化tSioTxCfg，Dx 对象序列长度、发送数据缓存长度、一次发送总的数据长度、SIO空闲时刻IO管脚输出电平等
......
csi_sio_tx_init(SIO0, &tSioTxCfg);              //初始化sio
set_led_rgb_store(byDipDataEnd,byCount);
```

```
csi_sio_send_dma(SIO0, DMA0, DMA_CH1, hwDmaSendData, 24);                    //dma发送
```

详见sio_led_rgb_send_dma_demo函数。

### 6.23.4 sio_led_rgb_recv_dma_demo

1、函数位置

sio_demo.c

2、功能描述

sio 驱动RGB LED(ws2812), RGB DATA = 24bit; 驱动数据输出排列方式:GRB。

3、参考用法

```
//sio端口配置
......
//初始化tEtbConfig、初始化tDmaConfig
.......
csi_etb_init();                  //初始化etcb
csi_etb_ch_config(ETB_CH21, &tEtbConfig);                        //配置etcb
csi_dma_ch_init(DMA0, DMA_CH1, &tDmaConfig);                     //初始化DMA
//初始化tSioTxCfg，Dx 对象序列长度、发送数据缓存长度、一次发送总的数据长度、SIO空闲时刻IO管脚输出电平等
......
csi_sio_rx_init(SIO0, &tSioRxCfg);               //初始化sio
set_led_rgb_store(byDipDataEnd,byCount);
csi_sio_recv_dma(SIO0, DMA0, DMA_CH2, (void*)byLedRxBuf, 24);                    //dma接收数据
```

详见sio_led_rgb_recv_dma_demo函数。

### 6.23.5 sio_led_rgb_recv_rxfull_demo

1、函数位置

sio_demo.c

2、功能描述

sio 接收RGB LED驱动数据，采用RXFULL中断模式；每收到byRxBufLen个bit，产生中断。

3、参考用法

```
//sio端口配置
......
//初始化tSioTxCfg，Dx 对象序列长度、发送数据缓存长度、一次发送总的数据长度、SIO空闲时刻IO管脚输出电平等
......
csi_sio_rx_init(SIO0, &tSioRxCfg);               //初始化sio
csi_sio_timeout_rst(SIO0, 20, ENABLE);  //接收超时复位, timeout cnt > bySpBitLen
csi_sio_set_buffer(g_wSioRxBuf, 24);        //设置接收数据buf和buf长度，将接收到的数据存放于用户定义的buffer中
csi_sio_receive(SIO0, wLedRxBuf, 24);       //接收数据
```

详见sio_led_rgb_recv_rxfull_demo函数。

### 6.23.6 sio_led_rgb_recv_rxdone_demo

1、函数位置

sio_demo.c

2、功能描述

sio 接收RGB LED驱动数据，采用RXDNE中断模式；每收到byRxCnt个bit，产生中断；此中断效率大于RXBUFFULL中断。

3、参考用法

```
//sio端口配置
......
//初始化tSioTxCfg，Dx 对象序列长度、发送数据缓存长度、一次发送总的数据长度、SIO空闲时刻IO管脚输出电平等
......
csi_sio_rx_init(SIO0, &tSioRxCfg);              //初始化SIO接收参数
csi_sio_timeout_rst(SIO0, 20, ENABLE);   //接收超时复位, timeout cnt > bySpBitLen
csi_sio_set_buffer(g_wSioRxBuf, 8);          //设置接收数据buf和buf长度，将接收到的数据存放于用户定义的buffer中
csi_sio_receive(SIO0, wRxBuf, 8);       //接收数据
```

详见sio_led_rgb_recv_rxdone_demo函数。

### 6.23.7 sio_hdq_send_demo

1、函数位置

sio_demo.c

2、功能描述

sio 实现TI HDQ单线通讯协议，主机发送数据；数据传输方式LSB，低7位是地址，最高位是R/W(0/1)控制位；一次传输两个字节。

3、参考用法

```
//sio端口配置
......
//初始化tSioTxCfg，Dx 对象序列长度、发送数据缓存长度、一次发送总的数据长度、SIO空闲时刻IO管脚输出电平等
......
csi_sio_tx_init(SIO0, &tHdqTxCfg);                          //初始化sio
wSendBuf[0] = sio_hdq_addr_conver(byHdqData[0] | HDQ_WR_CMD)   //第一个字节：地址+命令
wSendBuf[1] = sio_data_conver(byHdqData[1]                                  // 第二个字节：数据字节
csi_sio_send(SIO0, wSendBuf, 2);       //发送数据
```

详见sio_hdq_send_demo函数。

### 6.23.8 sio_hdq_recv_wrcmd_demo

1、函数位置

sio_demo.c

2、功能描述

sio 实现TI HDQ单线通讯协议，接收主机写命令数据；数据传输方式LSB，低7位是地址，最高位是R/W(0/1)控制位；一次传输两个字节。

3、参考用法

```
//sio端口配置
......
//初始化tHdqRxCfg，Dx 对象序列长度、发送数据缓存长度、一次发送总的数据长度、SIO空闲时刻IO管脚输出电平等
......
csi_sio_rx_init(SIO0, &tHdqRxCfg);                    //初始化sio
csi_sio_break_rst(SIO0, SIO_BKLEV_LOW, 19, ENABLE);   //检测接收break
csi_sio_set_buffer(g_wSioRxBuf, 1);          //设置接收数据buf和buf长度，将接收到的数据存放于用户定义的buffer中
csi_sio_receive(SIO0, wHdqRxBuf, 1);        //接收数据
```

详见sio_hdq_recv_wrcmd_demo函数。

### 6.23.9 sio_hdq_send_recv_demo

1、函数位置

sio_demo.c

2、功能描述

sio 实现TI HDQ单线通讯协议，主机读取数据；数据传输方式LSB，低7位是地址，最高位是R/W(0/1)控制位；一次传输两个字节。

3、参考用法

```
//sio端口配置
......
//初始化tHdqRxCfg，Dx 对象序列长度、发送数据缓存长度、一次发送总的数据长度、SIO空闲时刻IO管脚输出电平等
......
csi_sio_rx_init(SIO0, &tHdqRxCfg);                    //初始化sio
csi_sio_break_rst(SIO0, SIO_BKLEV_LOW, 19, ENABLE);   //检测接收break
csi_sio_set_buffer(g_wSioRxBuf, 1);          //设置接收数据buf和buf长度，将接收到的数据存放于用户定义的buffer中
......
//初始化tHdqTxCfg，Dx 对象序列长度、发送数据缓存长度、一次发送总的数据长度、SIO空闲时刻IO管脚输出电平等
......
csi_sio_tx_init(SIO0, &tHdqTxCfg);
sio_hdq_addr_conver(byHdqData[0]);           //第一个字节：地址+命令(低7位是地址，第8位是读写(0/1)控制位)
csi_sio_send(SIO0, wSendBuf, 1);             //发送读命
......
csi_sio_receive(SIO0, wHdqRx1Buf, 1);        //接收到需要数据
```

详见sio_hdq_send_recv_demo函数。

### 6.23.10 sio_hdq_recv_rdcmd_demo

1、函数位置

sio_demo.c

2、功能描述

sio 实现TI HDQ单线通讯协议，接收主机读命令数据；数据传输方式LSB，低7位是地址，最高位是R/W(0/1)控制位；一次传输两个字节。

3、参考用法

```
//sio端口配置
......
//初始化tHdqTxCfg，Dx 对象序列长度、发送数据缓存长度、一次发送总的数据长度、SIO空闲时刻IO管脚输出电平等
......
csi_sio_tx_init(SIO0, &tHdqTxCfg);
//初始化tHdqRxCfg，Dx 对象序列长度、发送数据缓存长度、一次发送总的数据长度、SIO空闲时刻IO管脚输出电平等
......
csi_sio_rx_init(SIO0, &tHdqRxCfg);                    //初始化sio
csi_sio_break_rst(SIO0, SIO_BKLEV_LOW, 19, ENABLE);   //检测接收break
csi_sio_set_buffer(g_wSioRxBuf, 1);          //设置接收数据buf和buf长度，将接收到的数据存放于用户定义的buffer中
......
csi_sio_receive(SIO0, wRxBuf, 1);              //接收到需要数据
.......
csi_sio_send(SIO0, &wTxData, 1);               //发送数据
```

详见sio_hdq_recv_rdcmd_demo函数。

## 6.24 CMP

### 6.24.1 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。

4、 参考用法

```
csi_pin_set_mux(PA9,PA9_CPIN1N);    //设置PA9为比较器的N级输入
csi_pin_set_mux(PA8,PA8_CPIN1P);    //设置PA8为比较器的P级输入
csi_pin_set_mux(PB2,PB2_CP0_OUT); //设置PB3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(CMP0,&tCmpCfg);
csi_cmp_start(CMP0);
```

详见cmp_base_demo函数。

### 6.24.2 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。

5、 参考用法

```
csi_pin_set_mux(PA9,PA9_CPIN1N);     //设置PA9为比较器的N级输入
csi_pin_set_mux(PA8,PA8_CPIN1P);     //设置PA8为比较器的P级输入
csi_pin_set_mux(PB2,PB2_CP0_OUT); //设置PB3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(CMP0,&tCmpCfg);
//初始化tCmpFltcrCfg，设置滤波器的时钟及滤波周期
......
csi_cmp_dflt1_config(CMP0,ENABLE,&tCmpDflt1Cfg);     //数字滤波1
......
csi_cmp_dflt2_config(CMP0,DISABLE,&tCmpDflt2Cfg);   //数字滤波2
csi_cmp_start(CMP,CMP_IDX1);
```

详见cmp_dfcr_demo函数。

### 6.24.3 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(PA9,PA9_CPIN1N);      //设置PA9为比较器的N级输入
csi_pin_set_mux(PA8,PA8_CPIN1P);     //设置PA8为比较器的P级输入
csi_pin_set_mux(PB2,PB2_CP0_OUT); //设置PB3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(CMP0,&tCmpCfg);
//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(CMP0,&tCmpWfcrCfg);
csi_cmp_start(CMP0);
csi_bt_timer_init(BT0, 2000);                    //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(ETB_CMP0_SYNCIN)：BT0 PEND -> ETCB -> CMP0 SYNCIN
......
csi_etb_init();
```

详见cmp_wfcr_demo函数。

## 6.25 OPA

### 6.25.1 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OPA0/1/2/3的输入输出引脚
......
//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_opa_init(OPA2,&ptOpaConfig_t);
csi_opa_start(OPA2);
csi_opa_init(OPA3,&ptOpaConfig_t);
csi_opa_start(OPA3);
```

详见opa_internal_gain_mode_test函数。

### 6.25.2 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OPA0/1/2/3的输入输出引脚
......
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_opa_init(OPA2,&ptOpaConfig_t);
csi_opa_start(OPA2);
csi_opa_init(OPA3,&ptOpaConfig_t);
csi_opa_start(OPA3);
```

详见opa_external_gain_mode_test函数。

## 6.26 WIZARD

### 6.26.1 lvr_wizard_demo

1、 函数位置

wizard_demo.c

2、 功能描述

Wizard图形化配置LVR功能的方法,及复位源查询方式

3、参考用法

```
// 用户配置格式化控制串启动文本，只有输入该文本之后，图形配置功能才会启动
// <<< Use Configuration Wizard in Context Menu >>>
// 符号图形化配置控制串
// <q> LVR Level (in Bytes)<2>[LVR_19//LVR_22//LVR_25//LVR_28//LVR_31//LVR_34//LVR_37//LVR_40]
// <i> Config LVR Level for the application
#define          LVR_LEVEL                LVR_31

csi_lvr_enable(LVR_LEVEL);
//VDD掉电到设定LVR电压，芯片复位用户配置格式化控制串启动文本，只有输入该文本之后，图形配置功能才会启动
 ......
```

详见lvr_wizard_demo函数。

### 6.26.2 cqcr_wizard_demo

1、 函数位置

wizard_demo.c

2、 功能描述

Wizard图形化配置CQCR功能的方法

3、参考用法

```
// 用户配置格式化控制串启动文本，只有输入该文本之后，图形配置功能才会启动
// <<< Use Configuration Wizard in Context Menu >>>
// 符号图形化配置控制串
// <q> SRC Freq (in Bytes)<2>[CQCR_SRCSEL_IM//CQCR_SRCSEL_ES//CQCR_SRCSEL_IS//CQCR_SRCSEL_HF]
// <i> Config SRC Freq for the application
#define          SRC_FREQ                CQCR_SRCSEL_IM
// <q> REF Freq(in Bytes)<2>[CQCR_REFSEL_EM//CQCR_REFSEL_ES]
// <i> Config REF Freq for the application
#define    REF_FREQ          CQCR_REFSEL_ES
// 数字图形化配置控制串
// <o> REF Value (in Bytes) <0x0-0x3FFFF:2>
// <i> Config REF Value for the application
#define          REF_VALUE               0x3f2

csi_set_cqcr(REF_FREQ,SRC_FREQ,REF_VALUE); //参考时钟选择 EM,  源时钟选择控 IM
 ......
```

详见cqcr_wizard_demo函数。