

	V1.0
	202012

# APT32F102

## CSI



IP

CDKV2.6

©

A decorative blue wavy line graphic that starts from the left side of the page and curves upwards towards the right side.

**Revision History**

V1.0	2020-12		

5.1 ..... 4

5.2 ..... 4

5.3 ..... 4

7.1 Q1    CSI ..... 7

7.2 Q2    CSI        API ..... 8

7.3 Q3    CSI            CDK ..... 8

8.1 ..... 8

8.2 ..... 8

1.

APT32F102x CSI Chip Standard Interface

2.

CSI sys csp API csp CSI sys

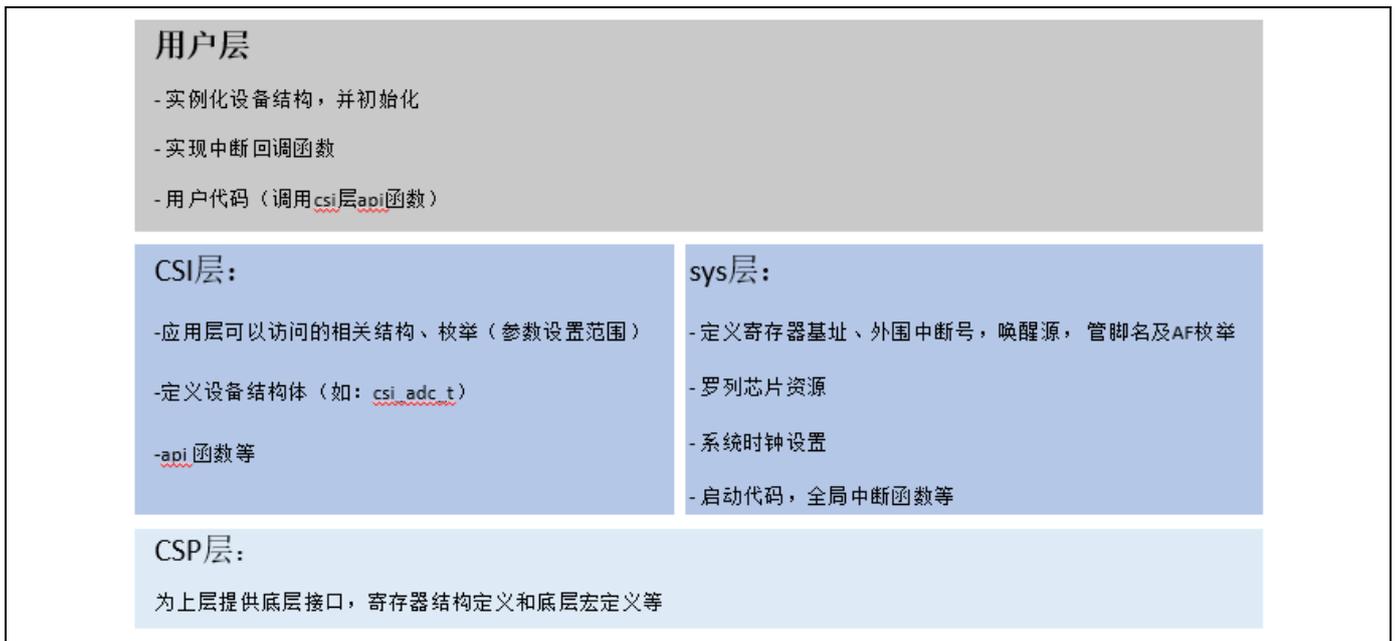


Figure 1

CDK

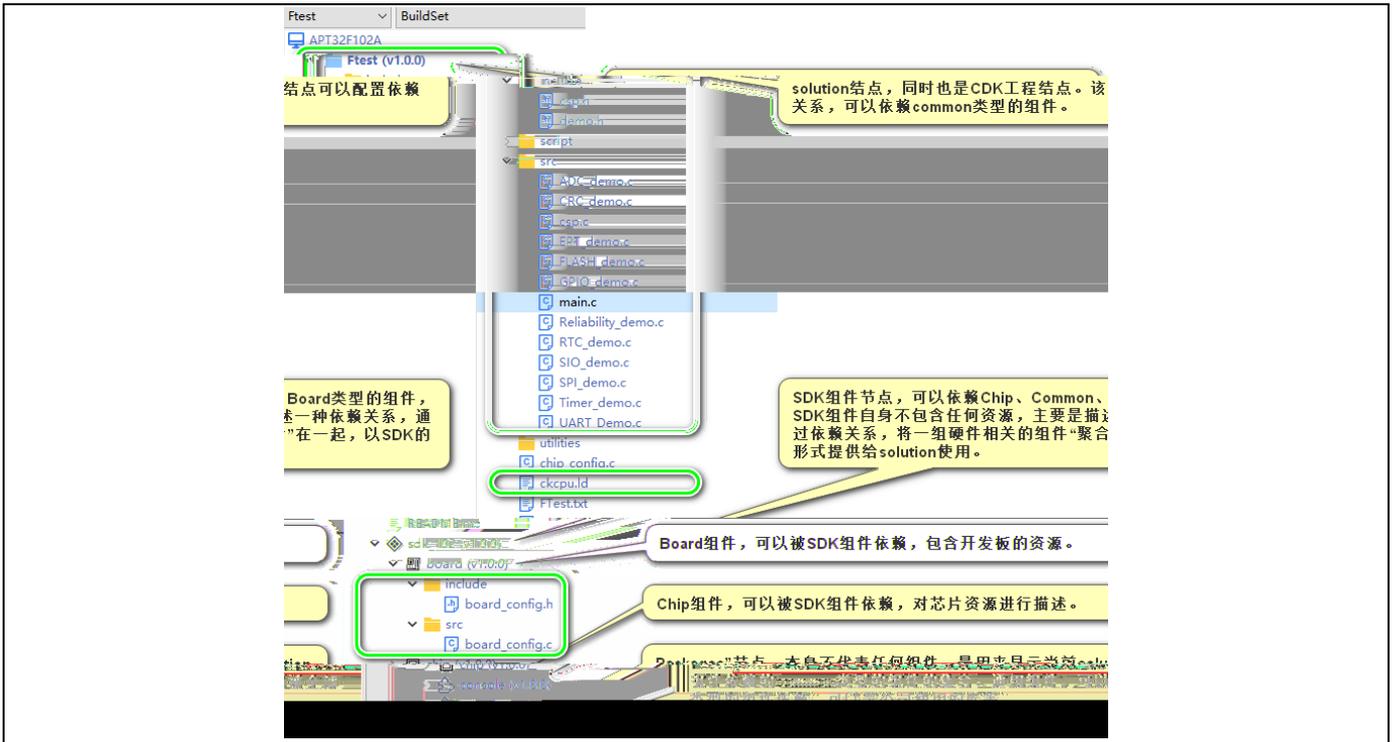


Figure 2

1

2

.pdf

3.

CDK2.6

CDK V2.6

IP

sdk\_102 -> board -> include -> board\_config.h

\board\include

```

/* example pin manager */

#define  CONSOLE_IDX                0
#define  CONSOLE_TXD                PA00
#define  CONSOLE_RXD                PA01
#define  CONSOLE_TXD_FUNC           PA01_UART0_TX
#define  CONSOLE_RXD_FUNC           PA00_UART0_RX

#define  EXI_PIN                    PA09
#define  EXI_PIN_FUNC               PIN_FUNC_INPUT

#define  SPI_IDX                    0
#define  SPI_MOSI_PIN               PA014
#define  SPI_MISO_PIN               PA015
#define  SPI_NSS_PIN                PB05
#define  SPI_SCK_PIN                PB04
#define  SPI_MOSI_PIN_FUNC           PA014_SPI_MOSI
#define  SPI_MISO_PIN_FUNC           PA015_SPI_MISO
#define  SPI_NSS_PIN_FUNC            PB05_SPI_NSS
#define  SPI_SCK_PIN_FUNC            PB04_SPI_SCK
    
```

Figure 3

xxx\_IDX      IP                      0      0      UART0                      g\_soc\_info[]  
 sdk\_102 -> chip -> sys > devices.c

```

const csi_perip_info_t g_soc_info[] = {
    {CK801_ADDR_BASE,            CORET_IRQn,            0,    DEV_CORET_TAG},
    {APB_SYS_BASE,              SYSCON_IRQn,          0,    DEV_SYSCON_TAG},
    {APB_IFC_BASE,              IFC_IRQn,            0,    DEV_IFC_TAG},
    {APB_ADC0_BASE,             ADC_IRQn,            0,    DEV_ADC_TAG},
    {APB_EPT0_BASE,             EPT0_IRQn,           0,    DEV_EPT_TAG},
}
    
```

Figure 4

4.

sdk\_102 -> board -> src -> board\_config.c

\board\src

```

/// system clock configuration parameters to define source, source freq(if selectable), sdiv and pdiv
const system_clk_config_t g_tSystemClkConfig[1] = {
    {SRC_HFOSC, HFOSC_48M_VALUE, SCLK_DIV2, PCLK_DIV1}
    //{SRC_EMOSC, 20000000, SCLK_DIV1, PCLK_DIV2}
    //{SRC_IMOSC, IMOSC_5M_VALUE, SCLK_DIV1, PCLK_DIV1}
    //{SRC_HFOSC, HFOSC_48M_VALUE, SCLK_DIV2, PCLK_DIV1}
    //{SRC_IMOSC, IMOSC_5M_VALUE, SCLK_DIV1, PCLK_DIV1}
};
    
```

Figure 5

g\_tSystemClkConfig      4                      SCLK      PCLK                      sys\_clk.h  
 sdk\_102 -> chip -> sys

5.

5.1

\_\_main() SRAM main

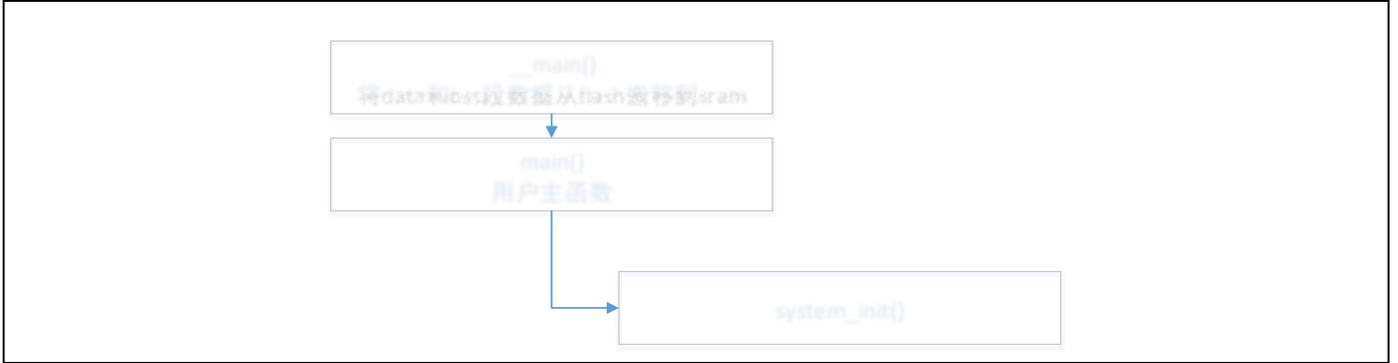


Figure 6

main system\_init()

```

void system_init(void)
{
    CK_CPU_DISALLNORMALIRQ;
    csi_reliability_init();
    csi_wdt_init(&tIwdt, 0);
    csi_wdt_stop(&tIwdt);
    soc_sysclk_config();
    soc_nvic_config();
    soc_gpio_config();
    csi_clk_init();
    CK_CPU_ENALLNORMALIRQ;
}
  
```

调试或要更改IWDT默认配置市

用delay函数时

必须有

Figure 7 system\_init()

5.2

board\_config.h

```
csi_pin_set_mux(ADC_PIN, ADC_PIN_FUNC);
```

Figure 8

5.3

5.3.1

CSI

ADC

-

```

typedef struct csi_adc csi_adc_t;
struct csi_adc {
    csi_dev_t dev; //< Hw-device info
    void (*callback)(csi_adc_t *adc, csi_adc_event_t event, void *arg); //< User callback signaled by driver event
    void *arg; //< 用户代码在attach callback时可以传入的中断相关的设置值
    uint32_t start; //< Start function
    uint32_t stop; //< Stop function
    csi_error_t error; //< Error returned by driver
    void state; //< state
    void priv; //< priv
};
    
```

设备硬件信息, 寄存器基址, 中断号等

中断回调函数

用户代码在attach callback时可以传入的中断相关的设置值

ADC采样频率

设备状态

使用的ADC channel数

Figure 9 ADC

● CSI\_xxx\_init

```

csi_error_t csi_adc_init(csi_adc_t *adc, uint32_t idx)
{
    CSI_PARAM_CHK(adc, CSI_ERROR);
    CSI_PARAM_CHK(CSI_ICR);
    adc->priv = 0U; //clr adc seq num
    if (target_get(DEV_ADC_TAG, idx, &adc->dev) != CSI_OK)
        ret = CSI_ERROR;
    else
    {
        adc_base = (csp_adc_t *)HANDLE_REG_BASE(adc);
        adc->state.writeable = 1U;
        adc->state.readable = 1U;
        adc->state.error = 0U;
        adc->callback = NULL;
        adc->arg = NULL;
        adc->data = NULL;
        adc->start = NULL;
        adc->stop = NULL;
        csi_clk_enable(&adc->dev); //adc p
        csp_adc_def_init(adc_base); //reset
        csp_adc_set_clk(adc_base, ENABLE); //ADC C
        csp_adc_set_bit_num(adc_base, ADC12_12BIT); //12BIT
        csp_adc_set_vref(adc_base, WERF_VDD_VSS); //ADC V
        csp_adc_en(adc_base); //enabl
    }
    return ret;
}
    
```

idx 是第几个ADC的意思。  
 这三个芯片中有多个一样的ID时具有意义的。  
 如A1220的ID为1220，A1220的ID为1220，A1220的ID为1220。

Peripheral clk en  
 all registers  
 LK ENABLE  
 AD  
 REF  
 e adc

Figure 10 ADC

5.3.2

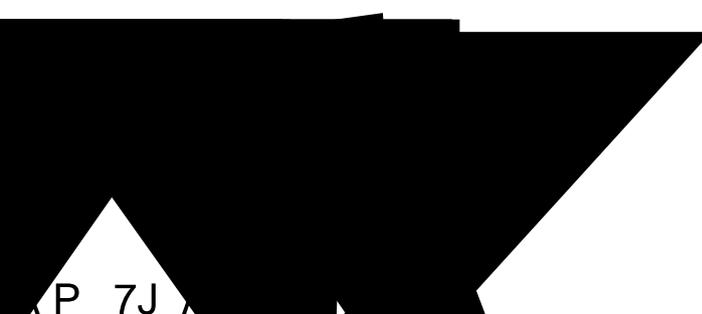
```

csi_adc_t g_tAdc;
csi_adc_init(&g_tAdc, ADC_IDX);
    
```

Figure 11 ADC

APT32F102                      ADC                      IP                      g\_soc\_info[]  
 sdk\_102 -> chip -> sys > devices.c                      0

6.



```

static void apt_adc_irqhandler(void *args)
{
    csi_adc_t *adc      = (csi_adc_t *)args;
    csp_adc_t *adc_base = (csp_adc_t *)HANDLE_REG_BASE(adc);

    uint8_t i;
    uint32_t wChnlNUM = (uint32_t)adc->priv;

    if(adc->data != NULL)
    {
        if(adc->num > 0)
        {
            for(i = 0; i < wChnlNUM; i++)
            {
                if(csp_adc_get_status(adc_base, ADC12_SEQ(i)))
                {
                    *(adc->data + i*s_byBufLen + adc->num- 1) = csp_adc_get_data(adc_base, i);
                    csp_adc_clr_status(adc_base, ADC12_SEQ(i));
                }
            }
            adc->num -- ;
        }

        if(adc->num == 0)
        {
            if (adc->callback)
            {
                adc->callback(adc, ADC_EVENT_CONVERT_COMPLETE, adc->arg);
                adc->state.readable = 10;
            }
        }
    }
}

```

Figure 13 adc\_irqhandler

### 6.1.2

#### 1. attach\_callback

```
CSI_adc_attach_callback(&g_tAdc, user_adc_event, (void *)arg);
```

#### 2.

```
void user_adc_event(CSI_adc_t *adc, CSI_adc_event_t event, void *arg)
```

arg

.c

## 7. Q & A

### 7.1 Q1 如果CSI的代码架构不能满足系统对实时性的要求怎么办

A1

1) ETCB

ETCB

2)

