文档版本	V1.0
发布日期	20191111

APT32F172 Lib 用户手册

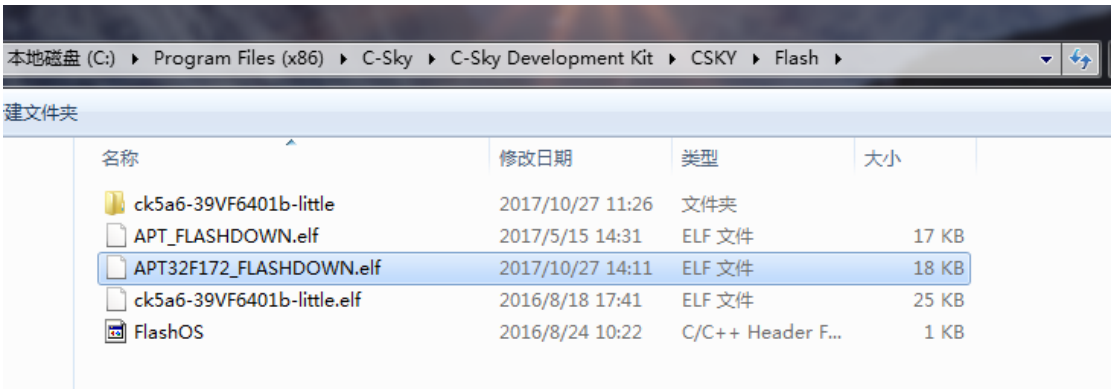
目录

1 前言

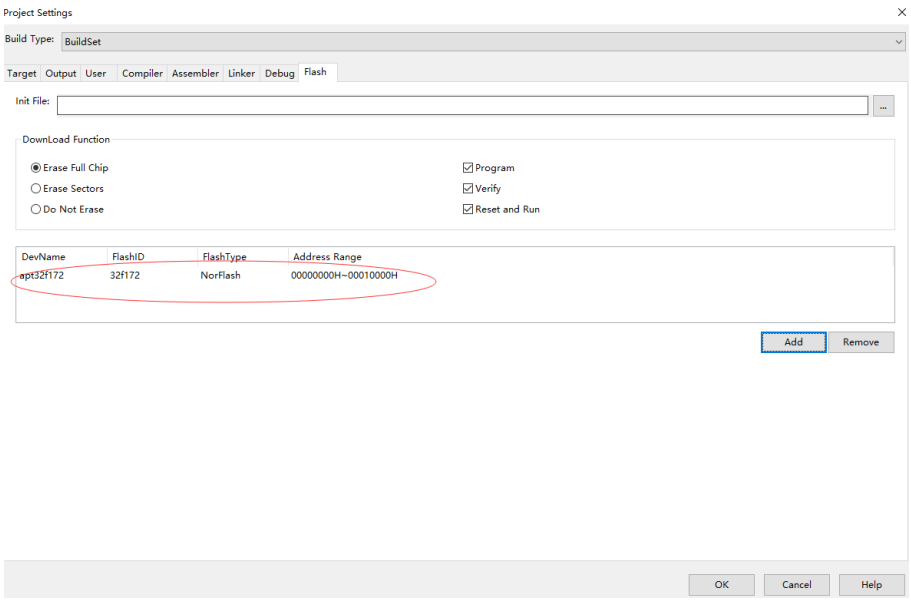
本文主要介绍如何使用 APT32F172 Lib 库文件

2 CDK 使用介绍

用户电脑第一次使用 CDK 及 APT 库文件工程时，需做如下步骤:
需要将 APT32F172_FLASHDOWN.elf 文件，复制到 CSKY 安装目录\CSKY\FLASH 下覆盖同名文件。



然后进入 Project settings->flash->add APT32F172_FLASHDOWN.elf 文件，添加完成后 Projec settings 界面会显示一下信息:

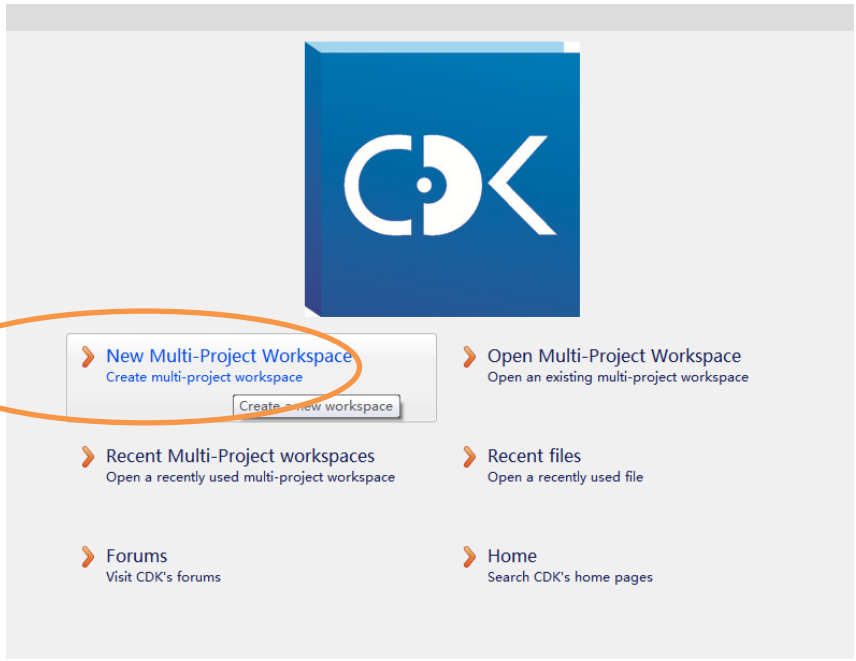


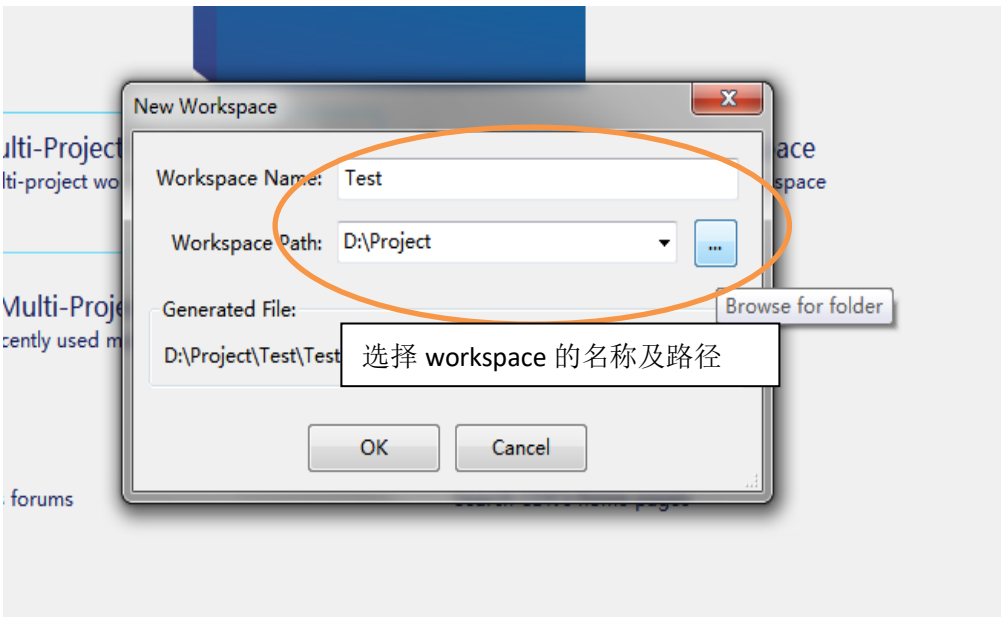
2.1 打开 workspace

直接双击库文件 workspace 文件夹的*.cdkws 开启工程，亦可通过 CDK 中的 Open Multi-Project Workspace 打开工程项目

2.2 新建 workspace

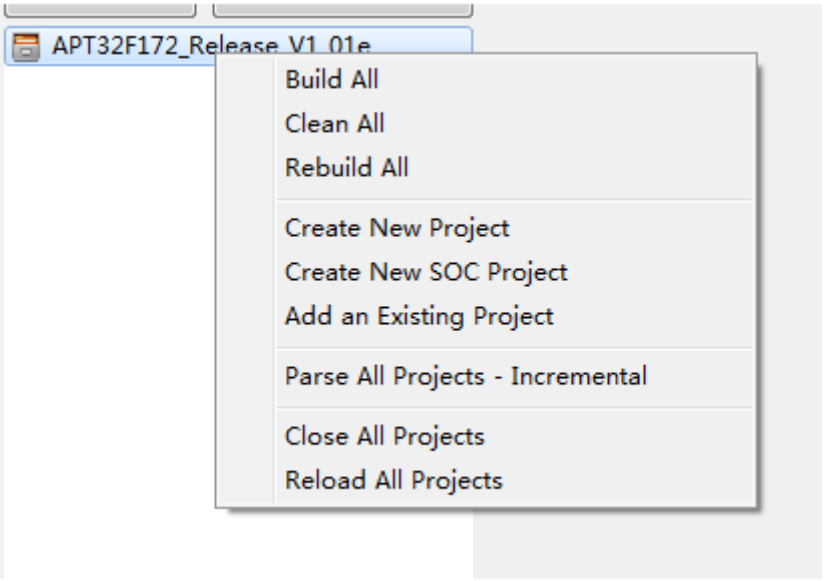
使用新的工程项目时，请按以下步骤（新建工程文件夹名字不能是中文）

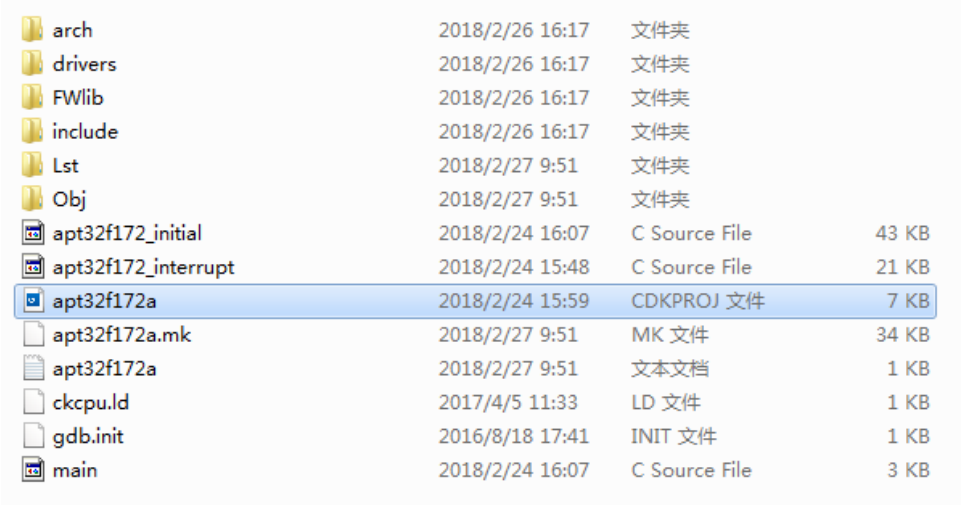




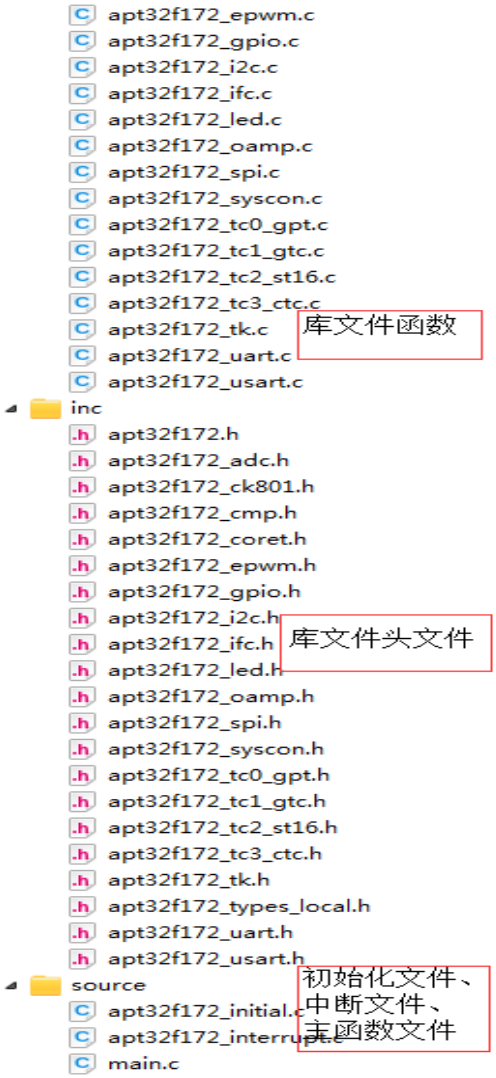
选择好路径后，将库文件中的 Source 文件夹拷贝到新的工程目录下

最后在 CDK 中将 Source 中的 apt32f172a.cdkproj 添加进来，即可。





2.3 库文件结构介绍



初始化函数编写在“apt32f172_intial.c”文档中。
若使用某个模块，在“APT32F172_init(void);”。
函数中添加对应模块初始化。
具体位置如下：

```
void APT32F172_init(void)
{
//-------------------------------------------------------------/
//Peripheral clock enable and disable
//EntryParameter:NONE
//ReturnValue:NONE
//-----------------------------|-------------------------------/
    SYSCON_WDT_CMD(DISABLE);                                    //Disable Watchdog
    SYSCON->PCER0=0xFFFFFFF;                                    //PCLK Enable  0x00410071
    SYSCON->PCER1=0xFFFFFFF;                                    //PCLK Enable
    while(!(SYSCON->PCSR0&0x1));                                //Wait PCLK enabled
//-------------------------------------------------------------/
//ISOSC/IMOSC/EMOSC/SYSCLK/IWDT/LVD/EM_CMFAIL/EM_CMRCV/CMD_ERR OSC stable interrupt
//EntryParameter:NONE
//ReturnValue:NONE
//-------------------------------------------------------------/
    SYSCON_Int_Enable();                                         //INT Vector
    SYSCON->IECR=ISOSC_ST|IMOSC_ST|EMOSC_ST|SYSCLK_ST;           //enable ISOSC Statues INT,IMOSC
    CK_CPU_EnAllNormalIrq();                                     //enable all IRQ
    SYSCON_CONFIG();                                             //syscon  initial
//-------------------------------------------------------------/
//Other IP config
//-------------------------------------------------------------/
   //GPIO_CONFIG();                                              //GPIO initial
    //LED_CONFIG();                                              //LED initial
    //CORET_CONFIG();                                            //CORET initial
    //GPT_CONFIG();                                              //GPT initial
    //GTC_CONFIG();                                              //GTC initial
    //STC16_CONFIG();                                            //STC16 initial
    //CTC_CONFIG();                                              //CTC initial
    //CMP_CONFIG();                                              //CMP initial
    //OPAMP_CONFIG();                                            //OPAMP initial
    //ADC_CONFIG();                                              //ADC initial
    //EPWM_CONFIG();                                             //EPWM initial
    //I2C_MASTER_CONFIG();                                       //I2C harware master initial
    //I2C_SLAVE_CONFIG();                                        //I2C harware slave initial
    //SPI_MASTER_CONFIG();                                       //SPI harware master initial
    //SPI_SLAVE_CONFIG();                                        //SPI harware slave initial
    //UART_CONFIG();                                             //UART initial
    //USART_CONFIG();                                            //USART initial
    //TK_CONFIG();                                               //TK initial
}
```

若使用GPIO功能，添加该函数即可

主函数编写在“main.c”文档中。具体位置如下：

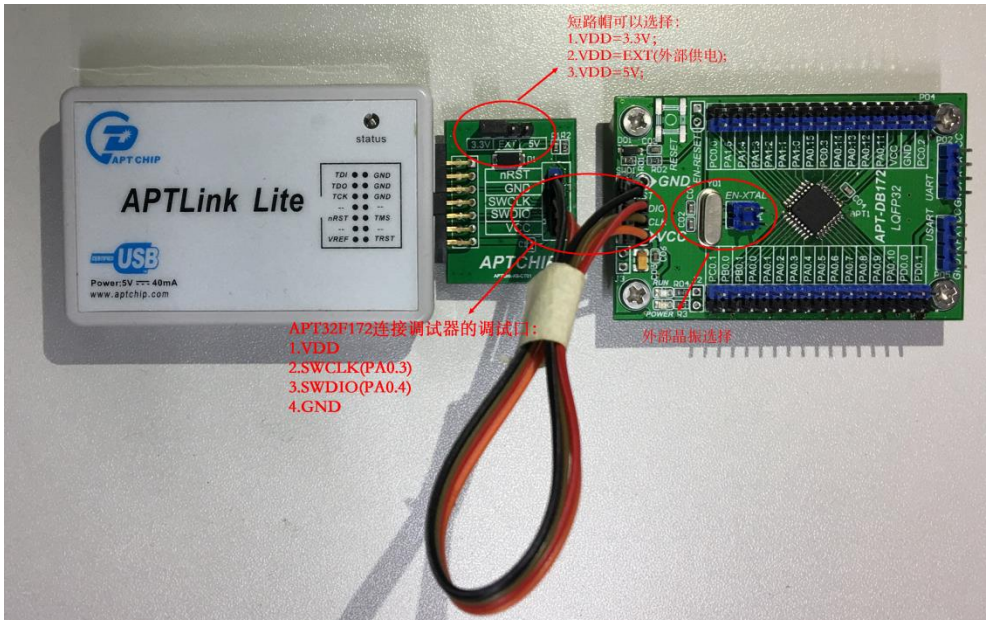
```
int main(void)
{
    APT32F172_init();
    while(1)
    {
        SYSCON_IWDCNT_Reload();

    }
}
```

编写代码处

中断编写在“apt32f172_interrupt.c”文档中。

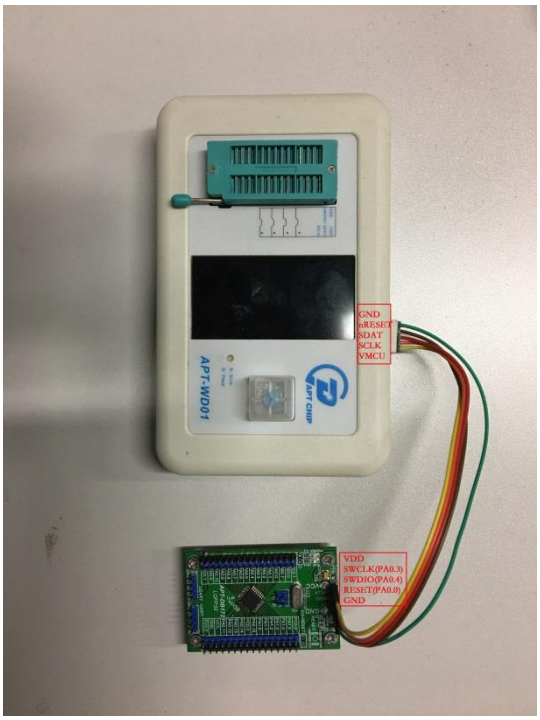
3 开发开发工具使用介绍

3.1 硬件连接



APTLink Lite 仅支持在线仿真和烧录使用，前提条件是 SWCLK 和 SWDIO 没有用作其他 IO 功能。若调试口用做其他 IO 功能，则不能用 APTLink Lite 调试和烧录，只能用 APT-WD01 擦除 APT32F172 后才能连接 APTLinkLite。

3.2 烧录连接



烧录脚位：VDD、SWCLK(PA0.3)、SWDIO(PA0.4)、RESET(PA0.0)、GND

3.3 APT-WD01 操作步骤

1、双击打开“APT MCU ISPV2”上位机软件。
2、进入软件后显示下面窗口：

3、确认是否连接成功，然后点击“擦除”按钮，进度条显示完成，表示擦除成功。

4 APT32F172 模块应用范例

4.1 SYSCON 系统模块

4.1.1 内部主频 20M 做系统时钟

功能说明:

开启内部主频 20MHz,并作为系统时钟。
WDT 使能，溢出时间 1s。
HCLK 分频设置为 1，PCLK 分频设置为 1。

操作步骤:

1. 关闭 WDT
2. 使能所有 IP 模块
3. 打开内部副频、内部主频、外部晶振、CPU 内部时钟中断
4. 打开全局中断向量
5. Syscon 函数配置

C 范例:

```
/*************************************************************/
//syscon Functions
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
void SYSCON_CONFIG(void)
{
    SYSCON_RST_VALUE();                         //SYSCON 所有寄存器复位赋值
    SYSCON_General_CMD(ENABLE,ENDIS_ISOSC);     //使能内部副频
    SYSCON_General_CMD(ENABLE,ENDIS_IMOSC);     //使能内部主频
    SYSCON_IMOSC_SETECTE(IMOSC_SETECTE_20M);    //选择内部主频为 20M
    SystemCLK_HCLKDIV_PCLKDIV_Config(SYSCLK_IMOSC,HCLK_DIV_1,PCLK_DIV_1);
    //内部主振作为系统时钟，HCLK 1 分频，PCLK 1 分频

    SYSCON_IWDCNT_Config(IWDT_TIME_1S,IWDT_INTW_DIV_1);
    //WDT 溢出时间 1s;WDT TEIM:1S*(1-(8-1)/8)=0.75S
```

```
    SYSCON_WDT_CMD(DISABLE);                    //禁止 WDT 功能
    SYSCON_IWDCNT_Reload();                     //清狗

    LVD_Disable();                              //禁止 LVR
}

/*****************************************************************************/
//APT32F172_init
//EntryParameter:NONE
//ReturnValue:NONE
/*****************************************************************************/
void APT32F172_init(void)
{
    SYSCON_WDT_CMD(DISABLE);                    //关闭 WDT
    SYSCON->PCER0=0xFFFFFFFF;                   //使能 IP
    SYSCON->PCER1=0xFFFFFFFF;                   //使能 IP
    while(!(SYSCON->PCSR0&0x1));                //判断 IP 是否使能

    SYSCON_Int_Enable();                        //使能 SYSCON 中断向量
    SYSCON->IECR=ISOSC_ST|IMOSC_ST|EMOSC_ST|SYSCLK_ST;
    //使能 ISOSC 时钟稳定中断,使能 IMOSC 时钟稳定中断,使能 EMOSC 时钟稳定中断
    CK_CPU_EnAllNormalIrq();                    //打开全局中断
    SYSCON_CONFIG();                            //syscon 参数  初始化
}

/*****************************************************************/
//main
/*****************************************************************/
int main(void)
{
    APT32F172_init();
    while(1)
    {
        SYSCON_IWDCNT_Reload();   //reload WDT
    }
}
```

4.1.2、内部主频 40M 做系统时钟

功能说明:

开启内部主频 40MHz,并作为系统时钟。

WDT 使能，溢出时间 1s。
HCLK 分频设置为 1，PCLK 分频设置为 1。

操作步骤:

1. 关闭 WDT
2. 使能所有 IP 模块
3. 打开内部副频、内部主频、外部晶振、CPU 内部时钟中断
4. 打开全局中断向量
5. Syscon 函数配置

C 范例:

```
/*************************************************************/
//syscon Functions
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
void SYSCON_CONFIG(void)
{
    SYSCON_RST_VALUE();                         //SYSCON 所有寄存器复位赋值
    SYSCON_General_CMD(ENABLE,ENDIS_ISOSC);     //使能内部副频
    SYSCON_General_CMD(ENABLE,ENDIS_IMOSC);     //使能内部主频
    SYSCON_IMOSC_SETECTE(IMOSC_SETECTE_40M);    //选择内部主频为 40M
    SystemCLK_HCLKDIV_PCLKDIV_Config(SYSCLK_IMOSC,HCLK_DIV_1,PCLK_DIV_1);
    //内部主振作为系统时钟，HCLK 1 分频，PCLK 1 分频

    SYSCON_IWDCNT_Config(IWDT_TIME_1S,IWDT_INTW_DIV_1);
    //WDT 溢出时间 1s;WDT TEIM:1S*(1-(8-1)/8)=0.75S
    SYSCON_WDT_CMD(DISABLE);                    //禁止 WDT 功能
    SYSCON_IWDCNT_Reload();                     //清狗

    LVD_Disable();                              //禁止 LVR
}

/*************************************************************************************/
//APT32F172_init                                                    /
//EntryParameter:NONE                                              /
//ReturnValue:NONE                                                  /
/*************************************************************************************/
void APT32F172_init(void)
{
```

```
    SYSCON_WDT_CMD(DISABLE);                                    //关闭 WDT
    SYSCON->PCER0=0xFFFFFFF;                                    //使能 IP
    SYSCON->PCER1=0xFFFFFFF;                                    //使能 IP
    while(!(SYSCON->PCSR0&0x1));                                //判断 IP 是否使能

    SYSCON_Int_Enable();                                        //使能 SYSCON 中断向量
    SYSCON->IECR=ISOSC_ST|IMOSC_ST|EMOSC_ST|SYSCLK_ST;
    //使能 ISOSC 时钟稳定中断,使能 IMOSC 时钟稳定中断,使能 EMOSC 时钟稳定中断
    CK_CPU_EnAllNormalIrq();                                    //打开全局中断
    SYSCON_CONFIG();                                            //syscon 参数  初始化
}

/*************************************************************/
//main
/*************************************************************/
int main(void)
{
    APT32F172_init();
    while(1)
    {
        SYSCON_IWDCNT_Reload();   //reload WDT
    }
}
```

4.2 GPIO 模块

4.2.1 GPIO 输入输出及外部中断唤醒配置

功能说明:

开启内部主频 20M 作系统时钟 HCLK 1 分频 PCLK 1 分频,关闭 WDT,关闭 LVR。
PA0.0 作为外部中断唤醒口，使能 PA0.0 内部上拉。
PA0.12 默认输出高,唤醒后 PA0.12 输出低。
PA0.1 漏极输出，PB0.0 强驱动使能

操作步骤:

1.SYSCON_CONFIG();函数配置
2.GPIO_CONFIG();函数配置
3.主循环代码

C 范例:

```
/*************************************************************/
//MCU goto SLEEP mode
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
void PCLK_goto_deepsleep_mode(void)
{
    SYSCON -> PWRCR = (SYSCON -> PWRCR & 0xFFFFCFFC) | (0xA66Aul<<16) | (0x3<<12) | 0x3 ;
    SYSCON->SCLKCR= (SYSCON -> SCLKCR & 0xFFFFF0FF) | SYSCLK_KEY | HCLK_DIV_2;
    //睡眠前 HCLK 分频必须>=2
    asm ("stop");                                   // Enter sleep mode
    SYSCON->SCLKCR= (SYSCON -> SCLKCR & 0xFFFFF0FF) | SYSCLK_KEY | HCLK_DIV_1;
}

/*************************************************************/
//GPIO Functions
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
void GPIO_CONFIG(void)
{
    GPIO_Init(GPIOA0,12,0);             //PA0.12 输出模式
    GPIO_Set_Value(GPIOA0,12,1);        //PA0.12 输出高
    GPIO_DriveStrength_EN(GPIOB0,0);    //PB0.0 强驱动使能
    GPIO_OpenDrain_EN(GPIOA0,1);        //PA0.1 漏极开路输出
    //----------  EXI FUNTION  ----------/
    //EXI0_INT= EXI0,EXI1_INT= EXI1, EXI2_INT=EXI2~EXI3, EXI3_INT=EXI4~EXI9, EXI4_INT=EXI10~EXI15
    GPIO_Init(GPIOA0,0,1);                  //PA0.00 输入模式, 下面配置为下降沿外部中断方式

    GPIO_IntGroup_Set(PA0);                 //外部中断选 GPIOA 组

    EXTI_trigger_CMD(ENABLE,EXI_PIN0,_EXIFT);   //PA0.0 下降沿中断
    EXTI_interrupt_CMD(ENABLE,EXI_PIN0);        //使能 EXI0 中断

    GPIO_EXTI_interrupt(GPIOA0,EXI_PIN0);       //使能 PA0.0 做外部中断

    EXI0_Int_Enable();                      //使能 EXI0 中断向量
    EXI0_WakeUp_Enable();                   //使能 EXI0 中断唤醒
}

/*************************************************************************************/
```

```
//APT32F172_init
//EntryParameter:NONE
//ReturnValue:NONE
/*******************************************************************************/
void APT32F172_init(void)
{
    SYSCON_WDT_CMD(DISABLE);                    //关闭 WDT
    SYSCON->PCER0=0xFFFFFFF;                    //使能 IP
    SYSCON->PCER1=0xFFFFFFF;                    //使能 IP
    while(!(SYSCON->PCSR0&0x1));                //判断 IP 是否使能

    SYSCON_Int_Enable();                        //使能 SYSCON 中断向量
    SYSCON->IECR=ISOSC_ST|IMOSC_ST|EMOSC_ST|SYSCLK_ST;
    //使能 ISOSC 时钟稳定中断,使能 IMOSC 时钟稳定中断,使能 EMOSC 时钟稳定中断
    CK_CPU_EnAllNormalIrq();                    //打开全局中断

    SYSCON_CONFIG();                            //syscon 参数 初始化

    GPIO_CONFIG();                              //GPIO 初始化
}

/*************************************************************/
//main
/*************************************************************/
int main(void)
{
    APT32F172_init();
    while(1)
    {
        SYSCON_IWDCNT_Reload();                 //清狗
        GPIO_Write_High(GPIOA0,12);             //GPIOA0.12 输出高
        PCLK_goto_deepsleep_mode();             //进入深度睡眠
        GPIO_Write_Low(GPIOA0,12);              //GPIOA0.12 输出低
    }
}
```

4.3 CORET 定时器模块

4.3.1 CORET 定时功能

功能说明:

开启内部主频 20MHz,并作为系统时钟。
PA0.12 输出占空比为 50%,周期为 2s 方波。

操作步骤:

1.SYSCON_CONFIG();函数配置
2.GPIO_CONFIG();函数配置
3.CORET_CONFIG();函数配置
4.PA0.12 输出翻转写在“CORETHandler();”中断函数中

C 范例:

```
volatile U32_T f_GPIO_Toggle;

/************************************************************/
//CORET Functions
//EntryParameter:NONE
//ReturnValue:NONE
/************************************************************/
void CORET_CONFIG(void)
{
    CORET_DeInit();                                    //Coret 所有寄存器复位赋值
    SYSCON_General_CMD(ENABLE,ENDIS_SYSTICK);
    //使能 STCLK 时钟 CK801->CORET_RVR=0x26259f;
    //CORETCLK=sysclock/8=20M/8=0.4us
    // e.g:1s=(CORET_RVR+1)*0.4us; CORET_RVR=0x2625a0-1=0x26259f
    CORET_reload();                                    // Coret CVR 清除

    CORET_CLKSOURCE_EX();                              //使用时钟源为 sysclk/8
    CORET_TICKINT_Enable();                            //使能计数器清零中断

    CORET_start();                                     //Coret 计时开始
    CORET_Int_Enable();                                //使能计数器清零中断向量
```

```
}

/*************************************************************/
//GPIO Functions
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
void GPIO_CONFIG(void)
{
    GPIO_Init(GPIOA0,12,0);              //PA0.12 输出模式

    GPIO_Set_Value(GPIOA0,12,1);         //PA0.12 输出高
}

/*****************************************************************************/
//APT32F172_init
//EntryParameter:NONE
//ReturnValue:NONE
/*****************************************************************************/
void APT32F172_init(void)
{
    SYSCON_WDT_CMD(DISABLE);                          //关闭 WDT


    SYSCON->PCER0=0xFFFFFFF;                          //使能 IP
    SYSCON->PCER1=0xFFFFFFF;                          //使能 IP
    while(!(SYSCON->PCSR0&0x1));                      //判断 IP 是否使能

    SYSCON_Int_Enable();                              //使能 SYSCON 中断向量
    SYSCON->IECR=ISOSC_ST|IMOSC_ST|EMOSC_ST|SYSCLK_ST;
    //使能 ISOSC 时钟稳定中断,使能 IMOSC 时钟稳定中断,使能 EMOSC 时钟稳定中断
    CK_CPU_EnAllNormalIrq();                          //打开全局中断

    SYSCON_CONFIG();                                      //syscon 参数 初始化

    GPIO_CONFIG();                                        //GPIO 初始化
    CORET_CONFIG();                                   //CORET 初始化
}

/*************************************************************/
//CORET Interrupt
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
```

```
void CORETHandler(void)
{
    CK801->CORET_CVR = 0;                   // Coret CVR 清除
    if(!f_GPIO_Toggle)
    {
        GPIO_Write_Low(GPIOA0,12);              //PA0.12 输出低
        f_GPIO_Toggle=1;
    }
    else
    {
        GPIO_Write_High(GPIOA0,12);     //PA0.12 输出高
        f_GPIO_Toggle=0;
    }
}
```

4.4 硬件 LED 驱动模块

4.4.1 硬件定时功能

功能说明:

开启内部主频 20MHz,并作为系统时钟。
使用硬件 8COM*6SEG 驱驱动 LED。
使能 8COM 对应 IO 强驱动模式。

操作步骤:

1. SYSCON_CONFIG();函数配置
2. GPIO_CONFIG();函数配置
3. LED_CONFIG();函数配置

C 范例:

```
/*************************************************************/
//GPIO Functions
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
void GPIO_CONFIG(void)
```

```
{
    GPIO_DriveStrength_EN(GPIOA1,3);        //PA1.3 强驱动使能 COM0
    GPIO_DriveStrength_EN(GPIOA1,4);        //PA1.4 强驱动使能 COM1
    GPIO_DriveStrength_EN(GPIOA1,5);        //PA1.5 强驱动使能 COM2
    GPIO_DriveStrength_EN(GPIOC0,0);        //PC0.0 强驱动使能 COM3
    GPIO_DriveStrength_EN(GPIOC0,1);        //PC0.1 强驱动使能 COM4
    GPIO_DriveStrength_EN(GPIOB0,0);            //PB0.0 强驱动使能 COM5
    GPIO_DriveStrength_EN(GPIOB0,1);            //PB0.1 强驱动使能 COM6
    GPIO_DriveStrength_EN(GPIOA0,0);        //PA0.0 强驱动使能 COM7
}

/*****************************************************************/
//LED Init
//EntryParameter:NONE
//ReturnValue:NONE
/*****************************************************************/
void LED_CONFIG(void)
{

    LED_RESET_VALUE();                                  //LED 所有寄存器复位赋值
    LED_seg_io_initial(LED_SEG_0,0);                    //led seg0 io 初始化
    LED_seg_io_initial(LED_SEG_1,0);                    //led seg1 io 初始化
    LED_seg_io_initial(LED_SEG_4,0);                    //led seg4 io 初始化
    LED_seg_io_initial(LED_SEG_5,0);                    //led seg5 io 初始化
    LED_seg_io_initial(LED_SEG_6,0);                    //led seg6 io 初始化
    LED_seg_io_initial(LED_SEG_7,0);                    //led seg7 io 初始化
    LED_Control_Config(LEDCLK_DIV32,0xff,LED_Bright_100,80,1);
    //F_LedClk=32/20M;COM0~COM7 使能;亮度 100%;COM 显示周期长度=(8*32/20M)*(80+7)=1.1MS;相邻 COM 互不交叠
    区时间时间=2 个时钟周期;
    LED_SEGDATX_data_write(SEGDAT_NUM0 , 0xaa);     //SegDat0=0xaa
    LED_SEGDATX_data_write(SEGDAT_NUM1 , 0xbb);     //SegDat1=0xbb
    LED_SEGDATX_data_write(SEGDAT_NUM2 , 0xcc);         //SegDat2=0xcc
    LED_SEGDATX_data_write(SEGDAT_NUM3 , 0xdd);     //SegDat3=0xdd
    LED_SEGDATX_data_write(SEGDAT_NUM4 , 0xee);         //SegDat4=0xee
    LED_SEGDATX_data_write(SEGDAT_NUM5 , 0xff);         //SegDat5=0xff
    LED_SEGDATX_data_write(SEGDAT_NUM6 , 0x99);         //SegDat6=0x99
    LED_SEGDATX_data_write(SEGDAT_NUM7 , 0x88);         //SegDat7=0x88
    LED_SCAN_ENABLE(ENABLE);                            //LED Scan enable
}

/*****************************************************************************/
//APT32F172_init                                                        /
//EntryParameter:NONE                                                 /
//ReturnValue:NONE                                                      /
```

```
/*******************************************************************************/
void APT32F172_init(void)
{
    SYSCON_WDT_CMD(DISABLE);                    //关闭 WDT

    SYSCON->PCER0=0xFFFFFFF;                    //使能 IP
    SYSCON->PCER1=0xFFFFFFF;                    //使能 IP
    while(!(SYSCON->PCSR0&0x1));                //判断 IP 是否使能

    SYSCON_Int_Enable();                        //使能 SYSCON 中断向量
    SYSCON->IECR=ISOSC_ST|IMOSC_ST|EMOSC_ST|SYSCLK_ST;
    //使能 ISOSC 时钟稳定中断,使能 IMOSC 时钟稳定中断,使能 EMOSC 时钟稳定中断
    CK_CPU_EnAllNormalIrq();                    //打开全局中断

    SYSCON_CONFIG();                            //syscon 参数 初始化

    GPIO_CONFIG();                                  //GPIO 初始化
    LED_CONFIG ();                                  //LED 初始化
}
```

4.5 ADC 数模转换模块

4.5.1 ADC 单次转换模式

功能说明:

开启内部主频 20MHz,并作为系统时钟。
使能 ADCIN10、ADCIN11 通道,12BIT ADC,参考电压选择内部 2.048V,单次转换模式，PRLVAL=2，ADC 采样周期=3。
ADC 转换周期：F_ANA=PCLK/(2*PRLVAL)-> F_ANA=20M/(2*2)=0.2us
ADC 转换时间：ADC 采样周期+1 转换周期*12bit（或 10bit）+3 个处理结果周期=18 个转换周期=18*0.2us=277KSPS
注意：
1.12bitADC 转换速率不能超过 500KSPS;10bitADC 转换速率不能超过 1MSPS。
2.选择内部参考电压 Vref 需要接 104 电容到 GND。

操作步骤:

1. SYSCON_CONFIG();函数配置
2. GPIO_CONFIG();函数配置
3. ADC_CONFIG();函数配置
4. 主函数读 ADC 数据

C 范例:

```
/*************************************************************/
//ADC Functions
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
void ADC_CONFIG(void)
{
    ADC12_RESET_VALUE();                                    //ADC 所有寄存器复位赋值
    ADC12_CLK_CMD(ADC_CLK_CR , ENABLE);                     //使能 ADC CLK
    ADC12_Software_Reset();                                 //ADC 软件复位
    ADC12_Configure_Mode(ADC12_12BIT , One_shot_mode ,0, 2 , 2);
    //选择 12BIT ADC; 单次模式; 转换优先序列寄存器为 0; ADC_CLK=PCLK/2*2=0.2us; 转换序列个数为 2
    ADC12_Configure_VREF_FVR(ADC12_FVR_2_048V);             //使用内部为参考电压 2.048mV
    ADC12_ConversionChannel_Config(ADC12_ADCIN10,ADC12_3CYCLES,ADC12_CV_RepeatNum1,ADC12_AVGDIS,0);
    //转换序列 0,选择 ADCIN10 通道, 3 个转换周期, 连续重复采样次数为 1, 平均值计算禁止
    ADC12_ConversionChannel_Config(ADC12_ADCIN11,ADC12_3CYCLES,ADC12_CV_RepeatNum1,ADC12_AVGDIS,1);
    //转换序列 1,选择 ADCIN11 通道, 3 个转换周期, 连续重复采样次数为 1, 平均值计算禁止
    ADC12_CMD(ENABLE);                                      //使能 ADC 模块
    ADC12_ready_wait();                                     //等待 ADC 模块配置完成
    ADC12_Control(ADC12_START);                             //ADC 模块启动
}

/*************************************************************************************/
//APT32F172_init
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************************************/
void APT32F172_init(void)
{
    SYSCON_WDT_CMD(DISABLE);                                //关闭 WDT

    SYSCON->PCER0=0xFFFFFFF;                                //使能 IP
    SYSCON->PCER1=0xFFFFFFF;                                //使能 IP
```

```
    while(!(SYSCON->PCSR0&0x1));                    //判断 IP 是否使能

    SYSCON_Int_Enable();                            //使能 SYSCON 中断向量
    SYSCON->IECR=ISOSC_ST|IMOSC_ST|EMOSC_ST|SYSCLK_ST;
    //使能 ISOSC 时钟稳定中断,使能 IMOSC 时钟稳定中断,使能 EMOSC 时钟稳定中断

    CK_CPU_EnAllNormalIrq();                        //打开全局中断
    SYSCON_CONFIG();                                //syscon 参数 初始化

    GPIO_CONFIG();                                  //GPIO 初始化
    ADC_CONFIG ();                                  //ADC 初始化
}

/*************************************************************/
//main
/*************************************************************/
volatile U32_T R_ADC_Buf1, R_ADC_Buf2;
int main(void)
{
    APT32F172_init();
    while(1)
    {
        SYSCON_IWDCNT_Reload();                     //清狗
        ADC12_SEQEND_wait(0);                       //等待转换序列 0 转换完成
    R_ADC_Buf1= ADC0->DR[0];                        //转换结果保存

        ADC12_SEQEND_wait(1);                       //读取转换序列 1 数据
        R_ADC_Buf2= ADC0->DR[1];                    //转换结果保存
        ADC12_Control(ADC12_START);                 //ADC 模块启动
    }
}
```

4.5.2 ADC 连续转换模式

功能说明:

开启内部主频 20MHz,并作为系统时钟。
使能 ADCIN10、ADCIN11 通道,12BIT ADC,参考电压选择外部 Vref,连续转换模式，PRLVAL=2，ADC 采样周期=3。
ADC 转换周期：F_ANA=PCLK/(2*PRLVAL)-> F_ANA=20M/(2*2)=0.2us
ADC 转换时间：ADC 采样周期+1 转换周期*12bit（或 10bit）+3 个处理结果周期=

18 个转换周期=18*0.2us=277KSPS
注意：
1.12bitADC 转换速率不能超过 500KSPS;10bitADC 转换速率不能超过 1MSPS。
2.选择内部参考电压 Vref 需要接 104 电容到 GND。

操作步骤:

1.SYSCON_CONFIG();函数配置
2.GPIO_CONFIG();函数配置
3.ADC_CONFIG();函数配置
4.主函数读 ADC 数据

C 范例:

```
/*************************************************************/
//ADC Functions
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
void ADC_CONFIG(void)
{
    ADC12_RESET_VALUE();                                    //ADC 所有寄存器复位赋值
    ADC12_CLK_CMD(ADC_CLK_CR , ENABLE);                     //使能 ADC CLK
    ADC12_Software_Reset();                                 //ADC 软件复位
    ADC12_Configure_Mode(ADC12_12BIT , One_shot_mode ,0, 2 , 2);
    //选择 12BIT ADC; 单次模式; 转换优先序列寄存器为 0; ADC_CLK=PCLK/2*2=0.2us; 转换序列个数为 2
    ADC12_Configure_VREF_FVR(ADC12_FVR_2_048V);             //使用内部为参考电压 2.048mV
   ADC12_ConversionChannel_Config(ADC12_ADCIN10,ADC12_3CYCLES,ADC12_CV_RepeatNum1,ADC12_AVGDIS,0);
   //转换序列 0,选择 ADCIN10 通道,3 个转换周期, 连续重复采样次数为 1,平均值计算禁止

   ADC12_ConversionChannel_Config(ADC12_ADCIN11,ADC12_3CYCLES,ADC12_CV_RepeatNum1,ADC12_AVGDIS,1);
   //转换序列 1,选择 ADCIN11 通道,3 个转换周期, 连续重复采样次数为 1,平均值计算禁止

    ADC12_CMD(ENABLE);                                      //使能 ADC 模块
    ADC12_ready_wait();                                     //等待 ADC 模块配置完成
    ADC12_Control(ADC12_START);                             //ADC 模块启动
}

/*************************************************************************************/
//APT32F172_init
//EntryParameter:NONE
```

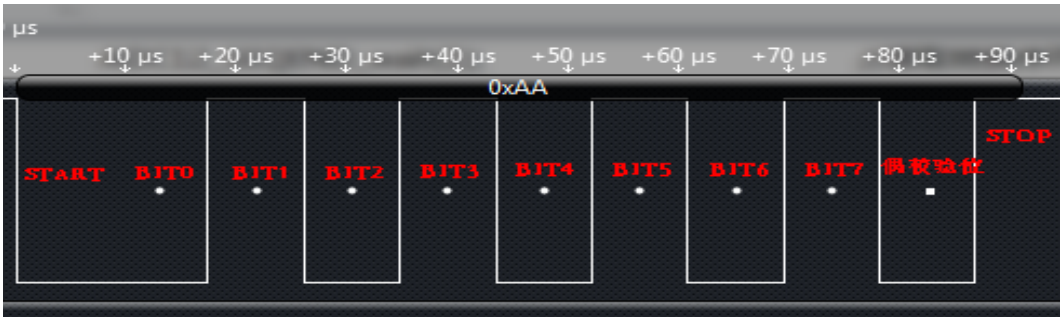
```
//ReturnValue:NONE
/*******************************************************************************/
void APT32F172_init(void)
{
    SYSCON_WDT_CMD(DISABLE);                    //关闭 WDT

    SYSCON->PCER0=0xFFFFFFF;                    //使能 IP
    SYSCON->PCER1=0xFFFFFFF;                    //使能 IP
    while(!(SYSCON->PCSR0&0x1));                //判断 IP 是否使能

    SYSCON_Int_Enable();                        //使能 SYSCON 中断向量
    SYSCON->IECR=ISOSC_ST|IMOSC_ST|EMOSC_ST|SYSCLK_ST;
    //使能 ISOSC 时钟稳定中断,使能 IMOSC 时钟稳定中断,使能 EMOSC 时钟稳定中断

    CK_CPU_EnAllNormalIrq();                    //打开全局中断
    SYSCON_CONFIG();                            //syscon 参数 初始化

    GPIO_CONFIG();                              //GPIO 初始化

    ADC_CONFIG ();                              //ADC 初始化
}

/*************************************************************/
//main
/*************************************************************/
volatile U32_T R_ADC_Buf1, R_ADC_Buf2;
int main(void)
{
    APT32F172_init();
    while(1)
    {
        SYSCON_IWDCNT_Reload();                 //清狗
        ADC12_SEQEND_wait(0);                   //等待转换序列 0 转换完成
        R_ADC_Buf1= ADC0->DR[0];                //转换结果保存

        ADC12_SEQEND_wait(1);                   //读取转换序列 1 数据
        R_ADC_Buf2= ADC0->DR[1];                //转换结果保存
    }
}
```

4.6 UART 通用异步收发器通讯模块

4.6.1 UART 发送接收

功能说明:

开启内部主频 20MHz,并作为系统时钟。
波特率：PCLK/DIV=20M/173=115200
使能偶数校验位。
发送顺序:bit0->bit7
UART 脚位选择：PA1.2->RXD1, PA1.1->TXD1
将 RXD 于 TXD 短接，发送 Uart 数据 0XAA，接收数据同样为 0XAA。
波形图如下：



操作步骤:

1. SYSCON_CONFIG();函数配置
2. GPIO_CONFIG();函数配置
3. UART_CONFIG();函数
4. 主函数发送 UART 数据
5. 中断函数中接收 UART 数据

C 范例:

```
/*************************************************************/
//UART Functions
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
void UART_CONFIG(void)
```

```
{
    UART_DeInit();                                  //UART 所有寄存器复位赋值
    UART_IO_Init(IO_PA1);                           //UART1 脚位选择 PA1.2->RXD1, PA1.1->TXD1
    UARTInitRxTxIntEn(UART1,173,UART1_PAR_EVEN);
    //使能 Uart 中断, baudrate=20000000/173=115200,  使能偶校验位
    UART1_Int_Enable();                             //uart1  中断向量使能
}
/*******************************************************************************/
//APT32F172_init                                                                 /
//EntryParameter:NONE                                                           /
//ReturnValue:NONE                                                               /
/*******************************************************************************/
void APT32F172_init(void)
{
    SYSCON_WDT_CMD(DISABLE);                        //关闭 WDT

    SYSCON->PCER0=0xFFFFFFF;                        //使能 IP
    SYSCON->PCER1=0xFFFFFFF;                        //使能 IP
    while(!(SYSCON->PCSR0&0x1));                    //判断 IP 是否使能

    SYSCON_Int_Enable();                            //使能 SYSCON 中断向量
    SYSCON->IECR=ISOSC_ST|IMOSC_ST|EMOSC_ST|SYSCLK_ST;
    //使能  ISOSC 时钟稳定中断,使能 IMOSC  时钟稳定中断,使能 EMOSC 时钟稳定中断

    CK_CPU_EnAllNormalIrq();                        //打开全局中断
    SYSCON_CONFIG();                                //syscon 参数  初始化

    GPIO_CONFIG();                                      //GPIO  初始化
    UART_CONFIG ();                                     //UART  初始化
}

/*****************************************************************/
//main
/*****************************************************************/
volatile U32_T R_Uart_RDBUF;
int main(void)
{
    APT32F172_init();
    while(1)
    {
        SYSCON_IWDCNT_Reload();                     //清狗
        UARTTxByte(UART1,0XAA);                     //发送 0xaa
        while(R_Uart_RDBUF!=0xAA);                  //判断是否接收到 0XAA
    }
```

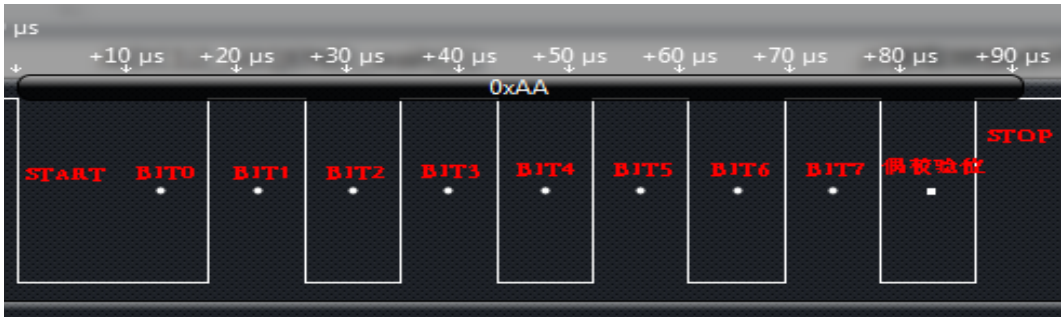
```
}

/*************************************************************/
//UART1 Interrupt
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
extern volatile U32_T R_Uart_RDBUF;
void UART1IntHandler(void)
{
    if ((UART1->ISR&UART_RX_INT_S)==UART_RX_INT_S)
    {
        UART1->ISR=UART_RX_INT_S;
        R_Uart_RDBUF=UART_ReturnRxByte(UART1);
    }
}
```

4.7 USART 通用同步异步收发器通讯模块

4.7.1 USART 发送接收

功能说明:

开启内部主频 20MHz,并作为系统时钟。
USART 作为普通模式。
波特率：38400 ， 字节长度 8bit ， 偶校验位 ， 1 个停止位 。
UART 脚位选择：PB0.0->RXD0, PB0.1->TXD0
将 RXD 于 TXD 短接，发送 Uart 数据 0XAA，接收数据同样为 0XAA。
波形图如下：



操作步骤:

1. SYSCON_CONFIG();函数配置
2. GPIO_CONFIG();函数配置
3. USART_CONFIG();函数配置
4. 主函数发送 USART 数据
5. 中断函数中接收 USART 数据

C 范例:

```
/*************************************************************/
//USART Functions
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
void USART_CONFIG(void)
{
    USART_DeInit();                                    //USART 所有寄存器复位赋值
    USART_CLK_Enable();                                //USART CLK 使能
    USART_CTRL_Config(RSTRX,ENABLE);                   //复位接收模块
    USART_CTRL_Config(RSTTX,ENABLE);                   //复位发射模块
    USART_IO_Init(USART_PB0);                          //USART 使用 PB0.0->RXD0, PB0.1->TXD0
    USART_MODE_Config(SENDTIME0,PCLK,CHRL8,ASYNC,PAR_EVEN,NBSTOP1,CHMODE_NORMAL);
    //重发次数为 0, CLKs=PCLK, 字节长度 8bit, 异步模式, 偶校验位, 1 个停止位, 普通模式
    USART_Baudrate_Cal(38400,20000000,PCLK, ASYNC);
    //波特率=38400, 主频选择 20M, PCLK 不分频, 异步模式
    USART_CTRL_Config(TXEN,ENABLE);                    //USART 发送使能
    USART_CTRL_Config(RXEN,ENABLE);                    //USART 接收使能
    USART_INT_Config(RXRDY_INT,ENABLE);                //USART 接收中断使能
    USART_INT_Config(TXRDY_INT,ENABLE);                //USART 发射中断使能
    USART_Int_Enable();                                //USART 中断向量使能
}

/*************************************************************************/
//APT32F172_init                                                        /
//EntryParameter:NONE                                                  /
//ReturnValue:NONE                                                      /
/*************************************************************************/
void APT32F172_init(void)
{
    SYSCON_WDT_CMD(DISABLE);                           //关闭 WDT
```

```
    SYSCON->PCER0=0xFFFFFFF;                    //使能 IP
    SYSCON->PCER1=0xFFFFFFF;                    //使能 IP
    while(!(SYSCON->PCSR0&0x1));                //判断 IP 是否使能

    SYSCON_Int_Enable();                        //使能 SYSCON 中断向量
    SYSCON->IECR=ISOSC_ST|IMOSC_ST|EMOSC_ST|SYSCLK_ST;
    //使能 ISOSC 时钟稳定中断,使能 IMOSC 时钟稳定中断,使能 EMOSC 时钟稳定中断

    CK_CPU_EnAllNormalIrq();                    //打开全局中断
    SYSCON_CONFIG();                            //syscon 参数 初始化

    GPIO_CONFIG();                                  //GPIO 初始化
    UART_CONFIG ();                                 //UART 初始化
}

volatile U32_T R_Uart_RDBUF;
/*****************************************************************/
//main
/*****************************************************************/
int main(void)
{
    APT32F172_init();
    while(1)
    {
        SYSCON_IWDCNT_Reload();                 //清狗
        USART_TxByte(0XAA);                         //发送 0xAA
        while(R_Uart_RDBUF!=0xAA);              //判断是否接收到 0XAA
    }
}

/*****************************************************************/
//USART0 Interrupt
//EntryParameter:NONE
//ReturnValue:NONE
/*****************************************************************/
void USART0IntHandler(void)
{
    unsigned int status;
    status = USART0->MISR & USART0->IMSCR ;
    if( status & USART_RXRDY )
    {
        CSP_USART_SET_ICR(USART0, USART_RXRDY);
        R_Uart_RDBUF = CSP_USART_GET_RHR(USART0);
    }
```

```
}
```

4.8 TC0/GPT 定时器模块

4.8.1 定时配置

功能说明:

开启内部主频 20MHz,并作为系统时钟。
计数器单周期时间：CLKS= MCLK /1=20MHz
PB0.0 输出周期 100us,占空比 50us 方波

操作步骤:

1.SYSCON_CONFIG();函数配置
2.GPIO_CONFIG();函数配置
3.GPT_CONFIG();函数配置
4.PB0.0 在 GPT 周期开始中断中翻转

C 范例:

```
/*************************************************************/
//GPT Functions
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
void GPT_CONFIG(void)
{
    GPT_RESET_VALUE(GPTCH0);                                    //GPT0 所有寄存器复位赋值
    GPTCHX_Clk_CMD(GPTCH0,ENABLE);                              //GPT0 时钟使能
    GPTCHX_SoftwareReset(GPTCH0);                               //GPT0 软件复位
    GPTCHX_CLK_Configure(GPTCH0,GPT_Mclk_Selecte_Pclk,GptClks_MCLK_DIV1,GPTCHX_CLKI_0,GPTCHX_BURST_SET_None);

    //GTP0 选择 PCLK 作为 MCLK;CLKS=MCLK/1;CLK 上升沿计数;关闭群脉冲模式
    GPTCHX_COUNT_Configure(GPTCH0,CPC_Reload_ENABLE); //GPT0 RC 匹配重新计数
    GPTCHX_Set_RA_RB_RC(GPTCH0,0,0,1000);                       //GPT0 RA=0,RB=0,RC=1000
    GPTCHX_ConfigInterrupt_CMD(GPTCH0,GPTCHX_INT_CPCS,ENABLE);//使能 GPT0 比较寄存器 C 匹配中断
    GPTCHX_CountClk_CMD(GPTCH0,ENABLE);                         //使能 GPT0 计数时钟
    GPTCHX_SWTRG(GPTCH0);                                       //软件触发 GPT0
    GPTCH0_Int_Enable();                                        //使能 GPT0 中断向量
```

```
}

/*******************************************************************************/
//APT32F172_init                                                               /
//EntryParameter:NONE                                                         /
//ReturnValue:NONE                                                            /
/*******************************************************************************/
void APT32F172_init(void)
{
    SYSCON_WDT_CMD(DISABLE);                    //关闭 WDT

    SYSCON->PCER0=0xFFFFFFF;                    //使能 IP
    SYSCON->PCER1=0xFFFFFFF;                    //使能 IP
    while(!(SYSCON->PCSR0&0x1));                //判断 IP 是否使能

    SYSCON_Int_Enable();                        //使能 SYSCON 中断向量
    SYSCON->IECR=ISOSC_ST|IMOSC_ST|EMOSC_ST|SYSCLK_ST;
    //使能 ISOSC 时钟稳定中断,使能 IMOSC 时钟稳定中断,使能 EMOSC 时钟稳定中断

    CK_CPU_EnAllNormalIrq();                    //打开全局中断
    SYSCON_CONFIG();                            //syscon 参数 初始化

    GPIO_CONFIG();                                  //GPIO 初始化
    GPT_CONFIG ();                                  //UART 初始化
}

volatile U32_T f_io_toggle;
/*************************************************************/
//GPT_0 Interrupt
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
void GPT_0IntHandler(void)
{
    if((GPTCH0->SR&GPTCHX_INT_CPCS)==GPTCHX_INT_CPCS)
    {
        GPTCH0->CSR = GPTCHX_INT_CPCS;
        if(!f_io_toggle)
        {
            f_io_toggle=1;
            GPIO_Write_High(GPIOB0,0);
        }
        else
        {
```

```
                f_io_toggle=0;
                GPIO_Write_Low(GPIOB0,0);
            }
        }
    }
```

4.8.2 脉宽调制配置

功能说明:

开启内部主频 20MHz,并作为系统时钟。
计数器单周期时间：CLKS= MCLK /1=20MHz
PB0.1 输出周期 100us,占空比 50us 方波

操作步骤:

1.SYSCON_CONFIG();函数配置
2.GPT_CONFIG();函数配置

C 范例:

```
/*************************************************************/
//GPT Functions
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
void GPT_CONFIG(void)
{
    GPT_RESET_VALUE(GPTCH0);                    //GPT0 所有寄存器复位赋值
    GPT_IO_Init(GPT_IO_IO0A,0);                 //IO0A (PB0.1)做 PWM 输出口
    GPTCHX_Clk_CMD(GPTCH0,ENABLE);              //GPT0 时钟使能
    GPTCHX_SoftwareReset(GPTCH0);               //GPT0 软件复位
    GPTCHX_CLK_Configure(GPTCH0,GPT_Mclk_Selecte_Pclk,GptClks_MCLK_DIV1,GPTCHX_CLKI_0,GPTCHX_BURST_SET_None);

    //GTP0 选择 PCLK 作为 MCLK;CLKS=MCLK/1;CLK 上升沿计数;关闭群脉冲模式
    GPTCHX_COUNT_Configure(GPTCH0,CPC_Reload_ENABLE); //GPT0 RC 匹配重新计数
    GPTCHX_PWM_Configure(GPTCH0,CPC_STOP_DISABLE,CPC_DisCountClk_DISABLE,CPC_Reload_ENABLE,EEVT_Reload_DISA
    BLE,EEVT_XC0_NONE,TIOA_SWTRG_OutPut_High,TIOA_EEVT_OutPut_NoChange,TIOA_CPA_OutPut_Low,TIOA_CPC_OutPut_Hi
    gh,TIOB_SWTRG_OutPut_High,TIOB_EEVT_OutPut_NoChange,TIOB_CPB_OutPut_Low,TIOB_CPC_OutPut_High);
    //GPT0RC 匹配停止计数禁止;RC 匹配停止计数时钟禁止;RC 匹配重新计数禁止;外部事件触发重新计数禁止;外部事、
```

```
    //件 XC0 选择禁止;软件触发 TIOA 为高电平;外部事件触发 TIOA 不改变;RA 匹配 TIOA 输出低电平;RC 匹配 TIOA
    //输出高;软件触发 TIOB 为高电平;外部事件触发 TIOB 不改变;RB 匹配 TIOB 输出低电平;RC 匹配 TIOB 输出高
    GPTCHX_Set_RA_RB_RC(GPTCH0,1000,0,2000);          //GPT0 RA=1000,RB=0,RC=2000
    GPTCHX_CountClk_CMD(GPTCH0,ENABLE);               //使能 GPT0 计数时钟
    GPTCHX_SWTRG(GPTCH0);                             //软件触发 GPT0
}

/*******************************************************************************/
//APT32F172_init
//EntryParameter:NONE
//ReturnValue:NONE
/*******************************************************************************/
void APT32F172_init(void)
{
    SYSCON_WDT_CMD(DISABLE);                          //关闭 WDT

    SYSCON->PCER0=0xFFFFFFF;                          //使能 IP
    SYSCON->PCER1=0xFFFFFFF;                          //使能 IP
    while(!(SYSCON->PCSR0&0x1));                      //判断 IP 是否使能

    SYSCON_Int_Enable();                              //使能 SYSCON 中断向量
    SYSCON->IECR=ISOSC_ST|IMOSC_ST|EMOSC_ST|SYSCLK_ST;
    //使能 ISOSC 时钟稳定中断,使能 IMOSC 时钟稳定中断,使能 EMOSC 时钟稳定中断

    CK_CPU_EnAllNormalIrq();                          //打开全局中断
    SYSCON_CONFIG();                                  //syscon 参数 初始化

    GPT_CONFIG ();                                    //GPT 初始化
}
```

4.8.3 输入捕捉配置

功能说明:

开启内部主频 20MHz,并作为系统时钟。
计数器单周期时间：CLKS= MCLK /1=20MHz。
PA0.11 捕捉周期 100us，占空比为 50us 方波。
RA_Capture 存储高电平计数。
RB_Capture 存储周期计数。

操作步骤:

1. SYSCON_CONFIG();函数配置
2. GPT_CONFIG();函数配置
3. 中断中读取 RA_Capture 和 RB_Capture 计数值

C 范例:

```
/*************************************************************/
//GPT Functions
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
void GPT_CONFIG(void)
{
    GPT_RESET_VALUE(GPTCH1);                                    //GPT1 所有寄存器复位赋值
    GPT_IO_Init(GPT_IO_IO1A,0);                                 //IO1A 初始化
    GPTCHX_Clk_CMD(GPTCH1,ENABLE);                          //GPT1 时钟使能
    GPTCHX_SoftwareReset(GPTCH1);                           //GPT1 软件复位
   GPTCHX_CLK_Configure(GPTCH1,GPT_Mclk_Selecte_Pclk,GptClks_MCLK_DIV1,GPTCHX_CLKI_0,GPTCHX_BURST_SET_None);

   //GTP1 选择 PCLK 作为 MCLK;CLKS=MCLK/1;CLK 上升沿计数;关闭群脉冲模式
   GPTCHX_Capture_Configure(GPTCH1,LDB_STOP_ENABLE,LDB_DisCountClk_DISABLE,ABETRG_TIOA_Rise,CPC_Reload_DISABL
   E,LDRA_TIOA_Fall,LDRB_TIOA_Rise);
   //GPT1;RB 载入停止计数使能;RB 载入停止时钟禁止;TIOA 上升沿触发重启计数;RC 匹配重新计数使能;RA 在 TIOA 下降
   沿载入;RB 在 TIOA 上升沿载入
    GPTCHX_ConfigInterrupt_CMD(GPTCH1,GPTCHX_INT_LDRAS,ENABLE);//使能 GPT1 载入寄存器 A 中断
    GPTCHX_ConfigInterrupt_CMD(GPTCH1,GPTCHX_INT_LDRBS,ENABLE); //使能 GPT1 载入寄存器 B 中断
    GPTCHX_CountClk_CMD(GPTCH1,ENABLE);             //使能 GPT1 计数时钟
    GPTCHX_SWTRG(GPTCH1);                           //软件触发 GPT1
    GPTCH1_Int_Enable();                                    //使能 GPT1 中断向量
}
/*************************************************************************************/
//APT32F172_init                                                                    /
//EntryParameter:NONE                                                             /
//ReturnValue:NONE                                                                  /
/*************************************************************************************/
void APT32F172_init(void)
{
    SYSCON_WDT_CMD(DISABLE);                    //关闭 WDT

    SYSCON->PCER0=0xFFFFFFF;                    //使能 IP
```

```
    SYSCON->PCER1=0xFFFFFFF;                    //使能 IP
    while(!(SYSCON->PCSR0&0x1));                //判断 IP 是否使能

    SYSCON_Int_Enable();                        //使能 SYSCON 中断向量
    SYSCON->IECR=ISOSC_ST|IMOSC_ST|EMOSC_ST|SYSCLK_ST;
    //使能 ISOSC 时钟稳定中断,使能 IMOSC 时钟稳定中断,使能 EMOSC 时钟稳定中断

    CK_CPU_EnAllNormalIrq();                    //打开全局中断
    SYSCON_CONFIG();                            //syscon 参数 初始化

    GPT_CONFIG ();                                  //GPT 初始化
}

volatile U32_T RA_Capture,RB_Capture；
/*****************************************************************/
//GPT_0 Interrupt
//EntryParameter:NONE
//ReturnValue:NONE
/*****************************************************************/
void GPT_1IntHandler (void)
{
    if((GPTCH1->SR&GPTCHX_INT_LDRAS)==GPTCHX_INT_LDRAS)
    {
        GPTCH1->CSR = GPTCHX_INT_LDRAS;
        RA_Capture=GPTCH1->RA;
    }
    if((GPTCH1->SR&GPTCHX_INT_LDRBS)==GPTCHX_INT_LDRBS)
    {
        GPTCH1->CSR = GPTCHX_INT_LDRBS;
        RB_Capture=GPTCH1->RB;
    }
}
```

4.9 TC1/GTC 定时器模块

4.9.1 定时器配置

功能说明:

开启内部主频 20MHz,并作为系统时钟。
计数器单周期时间：TCCLK=sysclock/2^1/10=1us

PB0.0 输出周期 100us，占空比 50us 方波

操作步骤：

1. SYSCON_CONFIG()；函数配置
2. GPIO_CONFIG()；函数配置
3. GTC_CONFIG()；函数配置
4. PB0.0 在 GTC 周期开始中断中翻转

C 范例：

```
/*************************************************************/
//CTC Functions
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
void GTC_CONFIG(void)
{
    GTC_RESET_VALUE();                                          //GTC 所有寄存器复位赋值
    GTC_SoftwareReset();                                        //GTC 软件复位
    GTC_Configure(GTC_FIN_IMOSC , 1 , 9 , Counter_Size_32BIT  , 50 , 0);      //TCCLK=sysclock/2^1/10=1us,周期
100*1US
    GTC_ControlSet_Configure(GTC_ControlSet_REPEAT,ENABLE);
    //使能该模式后，计数器在连续计数模式溢出或者周期模式周期结束后，会自动重新开始计数
    GTC_ConfigInterrupt_CMD(GTC_PSTARTI);                       //GTC 周期开始中断使能
    GTC_Int_Enable();                                           //GTC 中断向量使能
    GTC_Start();                                                //start counter
}

/*************************************************************************/
//APT32F172_init                                                  /
//EntryParameter:NONE                                            /
//ReturnValue:NONE                                               /
/*************************************************************************/
void APT32F172_init(void)
{
    SYSCON_WDT_CMD(DISABLE);                //关闭 WDT

    SYSCON->PCER0=0xFFFFFFF;                //使能 IP
    SYSCON->PCER1=0xFFFFFFF;                //使能 IP
    while(!(SYSCON->PCSR0&0x1));            //判断 IP 是否使能
```

```
    SYSCON_Int_Enable();                        //使能 SYSCON 中断向量
    SYSCON->IECR=ISOSC_ST|IMOSC_ST|EMOSC_ST|SYSCLK_ST;
    //使能 ISOSC 时钟稳定中断,使能 IMOSC 时钟稳定中断,使能 EMOSC 时钟稳定中断

    CK_CPU_EnAllNormalIrq();                    //打开全局中断
    SYSCON_CONFIG();                            //syscon 参数 初始化

    GPIO_CONFIG();                                  //GPIO 初始化
    GTC_CONFIG ();                                  //GTC 初始化
}

volatile U32_T f_io_toggle;
/*****************************************************************/
//GTC Interrupt
//EntryParameter:NONE
//ReturnValue:NONE
/*****************************************************************/
void GTCIntHandler(void)
{
    if((GTC->MISR&GTC_PSTARTI)==GTC_PSTARTI)
    {
        GTC->ICR = GTC_PSTARTI;
        if(!f_io_toggle)
        {
            f_io_toggle=1;
            GPIO_Write_High(GPIOB0,0);
        }
        else
        {
            f_io_toggle=0;
            GPIO_Write_Low(GPIOB0,0);
        }
    }
}
```

4.9.2 脉宽调试配置

功能说明:

开启内部主频 20MHz,并作为系统时钟。

PB0.0 输出周期 100us，占空比 50us 方波

操作步骤：

1. SYSCON_CONFIG()；函数配置
2. GTC_CONFIG()；函数配置

C 范例：

```
/*************************************************************/
//gtc Functions
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
void GTC_CONFIG(void)
{
    GTC_RESET_VALUE();                                          //GTC 所有寄存器复位赋值
    GTC_SoftwareReset();                                        //GTC 软件复位
    GTC_IO_Init(GTC_IO_TXOUT , 0 );                             //PB0.0 做 PWM 输出口
    GTC_Configure(GTC_FIN_IMOSC , 1 , 9 , Counter_Size_32BIT  , 100 , 50);      //TCCLK=sysclock/2^1/10=1us,周期
100*1US
    GTC_ControlSet_Configure(GTC_ControlSet_REPEAT,ENABLE);
    //使能该模式后，计数器在连续计数模式溢出或者周期模式周期结束后，会自动重新开始计数
    /GTC_ControlSet_Configure(GTC_ControlSet_PWMEN,ENABLE);     //使能 PWM 模式
    GTC_Start();                                                //start counter
}

/*************************************************************************/
//APT32F172_init                                                /
//EntryParameter:NONE                                          /
//ReturnValue:NONE                                             /
/*************************************************************************/
void APT32F172_init(void)
{
    SYSCON_WDT_CMD(DISABLE);                    //关闭 WDT

    SYSCON->PCER0=0xFFFFFFF;                    //使能 IP
    SYSCON->PCER1=0xFFFFFFF;                    //使能 IP
    while(!(SYSCON->PCSR0&0x1));                //判断 IP 是否使能

    SYSCON_Int_Enable();                        //使能 SYSCON 中断向量
    SYSCON->IECR=ISOSC_ST|IMOSC_ST|EMOSC_ST|SYSCLK_ST;
```

```
    //使能 ISOSC 时钟稳定中断,使能 IMOSC 时钟稳定中断,使能 EMOSC 时钟稳定中断

    CK_CPU_EnAllNormalIrq();                    //打开全局中断
    SYSCON_CONFIG();                            //syscon 参数 初始化

    GTC_CONFIG ();                                  //GTC 初始化
}
```

4.9.3 输入捕捉配置

功能说明:

开启内部主频 20MHz,并作为系统时钟。
计数器单周期时间：TCCLK=sysclock/2^1/10=1us
PA1.1 捕捉高电平 50us，低电平 50us 方波。
R_Capture_buf1 存储周期时间。
R_Capture_buf2 存储低电平时间。
R_Capture_buf3 存储高电平时间。

操作步骤:

1.SYSCON_CONFIG();函数配置
2.GTC_CONFIG();函数配置
3.GTCIntHandler();中断函数配置

C 范例:

```
/*************************************************************/
//gtc Functions
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
void GTC_CONFIG(void)
{
    GTC_RESET_VALUE();                                      //GTC 所有寄存器复位赋值
    GTC_SoftwareReset();                                    //GTC 软件复位
    GTC_IO_Init(GTC_IO_TCAPX , 1 );                         //PA1.1 做捕捉输入口
    GTC_Configure(GTC_FIN_PCLK, 1 , 9 , Counter_Size_32BIT   , 0 , 0);      //TCCLK=sysclock/2^1/10=1us
    GTC_ControlSet_Configure(GTC_ControlSet_STOPCLEAR,ENABLE);      //计数器停止后清除计数值
    GTC_ControlSet_Configure(GTC_ControlSet_CNTM,ENABLE);
```

```
    //工作在连续计数模式，计数值自增直到溢出
    GTC_ControlSet_Configure(GTC_ControlSet_REPEAT,ENABLE);          //使能循环重复模式
    GTC_ControlSet_Configure(GTC_ControlSet_CAPT_F,ENABLE);          //下降沿捕捉使能
    GTC_ControlSet_Configure(GTC_ControlSet_CAPT_TCAP,ENABLE);       //捕捉输入使能
    GTC_ConfigInterrupt_CMD(GTC_CAPTI, ENABLE);                      //捕捉中断使能
    GTC_Start();                                                     //start counter
    GTC_Int_Enable();                                                //使能 GTC 中断向量
}

/*******************************************************************************/
//APT32F172_init                                                              /
//EntryParameter:NONE                                                        /
//ReturnValue:NONE                                                           /
/*******************************************************************************/
void APT32F172_init(void)
{
    SYSCON_WDT_CMD(DISABLE);                    //关闭 WDT

    SYSCON->PCER0=0xFFFFFFF;                    //使能 IP
    SYSCON->PCER1=0xFFFFFFF;                    //使能 IP
    while(!(SYSCON->PCSR0&0x1));                //判断 IP 是否使能

    SYSCON_Int_Enable();                        //使能 SYSCON 中断向量
    SYSCON->IECR=ISOSC_ST|IMOSC_ST|EMOSC_ST|SYSCLK_ST;
    //使能 ISOSC 时钟稳定中断,使能 IMOSC 时钟稳定中断,使能 EMOSC 时钟稳定中断

    CK_CPU_EnAllNormalIrq();                    //打开全局中断
    SYSCON_CONFIG();                            //syscon 参数 初始化

    GTC_CONFIG ();                                  //GTC 初始化
}

volatile U32_T    R_Capture_buf1,R_Capture_buf2,R_Capture_buf3,f_GTC_CaptTrigg;
/*****************************************************************/
//GTC Interrupt
//EntryParameter:NONE
//ReturnValue:NONE
/*****************************************************************/
void GTCIntHandler(void)
{
    if((GTC->MISR&GTC_CAPTI)==GTC_CAPTI)
    {
        GTC->ICR = GTC_CAPTI;
        if(!f_GTC_CaptTrigg)
```

```
        {
            f_GTC_CaptTrigg=1;
            R_Capture_buf1=GTC->CDCR;                                   //R_Capture_buf1 周期时间
            GTC_ControlSet_Configure(GTC_ControlSet_CAPT_R,ENABLE);     //capture down enable
            GTC_ControlSet_Configure(GTC_ControlSet_CAPT_F,DISABLE);
            GTC_Stop();
            GTC_Start();
        }
        else
        {
            f_GTC_CaptTrigg=0;
            R_Capture_buf2=GTC->CUCR;                                   // R_Capture_buf2 低电平时间
            GTC_ControlSet_Configure(GTC_ControlSet_CAPT_R,DISABLE);    //捕捉上升沿禁止
            GTC_ControlSet_Configure(GTC_ControlSet_CAPT_F,ENABLE);     //捕捉下降沿使能
        }
        if(R_Capture_buf1>R_Capture_buf2)                               //R_Capture_buf3 高电平时间
        {
            R_Capture_buf3=R_Capture_buf1-R_Capture_buf2;
        }
        else
        {
            R_Capture_buf3=R_Capture_buf2-R_Capture_buf1;
        }
    }
}
```

4.10 TC2/STC16 定时器模块

4.10.1 定时器模块

功能说明:

开启内部主频 20MHz,并作为系统时钟。
计数器单周期时间：Stc16_TimeClk=PCLK/2^0/20=1us
PB0.0 捕捉高电平 50us，低电平 50us 方波。

操作步骤:

1.SYSCON_CONFIG();函数配置
2.GPIO_CONFIG();函数配置
3. STC16_CONFIG ();函数配置
4.PB0.0 在 STC16 周期结束中断中翻转

C 范例:

```
/*************************************************************/
//stc16 Functions
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
void STC16_CONFIG(void)
{
    STC16_RESET_VALUE();                //STC16 所有寄存器复位赋值
    STC16_Softreset();                  //STC16 软件复位
    STC16_Clk_CMD(ENABLE);              //STC16 时钟使能
    STC16_Configure(STC16_Count_mode_Continue,STC16_Count_STOPTYPE_StopConmand,STC16_CM0_Mode_Capture,STC16_CM
    1_Mode_Match,19,0);
    //STC16 Configure;设置为连续计数模式;单次计数模式下,选择立即停止模式;通道 0 作为匹配模式;通道 1 作为捕捉模
    式;stc16_timeclk=pclk/(19+1)/2^0=1us
    STC16_CNR_CC0_CC1_Load(50,0,0);          //STC16 CNTR=50,CC0R=0,CC1R=0
    STC16_MINT_CMD(ST16_PENDI,ENABLE); //STC16 周期结束中断使能
    STC16_Start();                           //Start stc16
    STC16_Int_Enable();                      //ENABLE STC16 中断向量
}

/*************************************************************************************/
//APT32F172_init                                                              /
//EntryParameter:NONE                                                        /
//ReturnValue:NONE                                                            /
/*************************************************************************************/
void APT32F172_init(void)
{
    SYSCON_WDT_CMD(DISABLE);                //关闭 WDT

    SYSCON->PCER0=0xFFFFFFF;                //使能 IP
    SYSCON->PCER1=0xFFFFFFF;                //使能 IP
    while(!(SYSCON->PCSR0&0x1));            //判断 IP 是否使能
```

```
    SYSCON_Int_Enable();                        //使能 SYSCON 中断向量
    SYSCON->IECR=ISOSC_ST|IMOSC_ST|EMOSC_ST|SYSCLK_ST;
    //使能 ISOSC 时钟稳定中断,使能 IMOSC 时钟稳定中断,使能 EMOSC 时钟稳定中断

    CK_CPU_EnAllNormalIrq();                    //打开全局中断
    SYSCON_CONFIG();                            //syscon 参数 初始化

    GPIO_CONFIG();                                  //GPIO 初始化
    STC16_CONFIG ();                            //stc16 初始化
}
/*****************************************************************/
//STC16 Interrupt
//EntryParameter:NONE
//ReturnValue:NONE
/*****************************************************************/
void STC16IntHandler(void)
{
    if((ST16->MISR&ST16_PENDI)==ST16_PENDI)
    {
        ST16->ICR = ST16_PENDI;
        if(!f_io_toggle)
        {
            f_io_toggle=1;
            GPIO_Write_High(GPIOB0,0);
        }
        else
        {
            f_io_toggle=0;
            GPIO_Write_Low(GPIOB0,0);
        }
    }
}
```

4.10.2 输入捕捉配置

功能说明:

开启内部主频 20MHz,并作为系统时钟。
计数器单周期时间：Stc16_TimeClk=PCLK/2^0/20=1us
PB0.0 捕捉高电平 50us，低电平 50us 方波。
R_LowLevel_T 存储低电平时间。
R_HighLevel_T 存储高电平时间。

操作步骤:

1. SYSCON_CONFIG();函数配置
2. STC16_CONFIG();函数配置
3. STC16IntHandler();中断函数配置

C 范例:

```
/*************************************************************/
//stc16 Functions
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
void STC16_CONFIG(void)
{
    STC16_RESET_VALUE();                //STC16 所有寄存器复位赋值
    STC16_Softreset();                  //STC16 软件复位
    STC16_IO_Init(STC16_IO_CAP1,0);     //STC16 CAP1 初始化
    STC16_Clk_CMD(ENABLE);              //STC16 时钟使能
    STC16_Channel1_CMD(ENABLE);         //STC16 通道 1 使能
    STC16_Configure(STC16_Count_mode_Continue,STC16_Count_STOPTYPE_StopConmand,STC16_CM0_Mode_Match,STC16_CM1
    _Mode_Capture,19,0);
    //STC16 Configure;设置为连续计数模式;单次计数模式下,选择立即停止模式;通道 0 作为匹配模式;通道 1 作为捕捉模
    式;stc16_timeclk=pclk/(19+1)/2^0=1us
    STC16_CNR_CC0_CC1_Load(0xffff,0,0);     //STC16 CNTR=0xffff,CC0R=0,CC1R=0
    STC16_CINT_CMD(ST16_CC1RI,ENABLE);      //STC16 通道 1 上升沿中断使能
    STC16_CINT_CMD(ST16_CC1FI,ENABLE);      //STC16 通道 1 下降沿中断使能
    STC16_Start();                          //Start stc16
    STC16_Int_Enable();                     //ENABLE STC16 中断向量
}

/*************************************************************/
//syscon Functions
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
void SYSCON_CONFIG(void)
{
//------SYSTEM CLK AND PCLK FUNTION--------------------------/
    SYSCON_RST_VALUE();                                 //SYSCON 所有寄存器复位赋值
    SYSCON_General_CMD(ENABLE,ENDIS_IDLE_PCLK);         //SLEEP 模式下 PCLK 使能
    SYSCON_General_CMD(ENABLE,ENDIS_ISOSC);             //使能内部副频
```

```
    SYSCON_General_CMD(ENABLE,ENDIS_IMOSC);                    //使能内部主频
    SYSON_EMOSC_32k_EN();                                      //使能 外部晶振外接 32.768K
    SYSCON_IMOSC_SETECTE(IMOSC_SETECTE_20M);                   //选择内部主频为 20M
    SystemCLK_HCLKDIV_PCLKDIV_Config(SYSCLK_IMOSC,HCLK_DIV_1,PCLK_DIV_1);
    //内部主振作为系统时钟，HCLK 1 分频，PCLK 1 分频
//-----------   WDT FUNTION   -------------------------------/
    SYSCON_IWDCNT_Config(IWDT_TIME_1S,IWDT_INTW_DIV_1);
    //WDT 溢出时间 1s;WDT TEIM:1S*(1-(8-1)/8)=0.75S
    SYSCON_WDT_CMD(DISABLE);
//-----------   LVD FUNTION   -------------------------------/
    LVR_Disable();
}

/*****************************************************************************/
//APT32F172_init
//EntryParameter:NONE
//ReturnValue:NONE
/*****************************************************************************/
void APT32F172_init(void)
{
    SYSCON_WDT_CMD(DISABLE);                   //关闭 WDT

    SYSCON->PCER0=0xFFFFFFF;                   //使能 IP
    SYSCON->PCER1=0xFFFFFFF;                   //使能 IP
    while(!(SYSCON->PCSR0&0x1));               //判断 IP 是否使能

    SYSCON_Int_Enable();                       //使能 SYSCON 中断向量
    SYSCON->IECR=ISOSC_ST|IMOSC_ST|EMOSC_ST|SYSCLK_ST;
    //使能 ISOSC 时钟稳定中断,使能 IMOSC 时钟稳定中断,使能 EMOSC 时钟稳定中断

    CK_CPU_EnAllNormalIrq();                   //打开全局中断
    SYSCON_CONFIG();                           //syscon 参数 初始化

    STC16_CONFIG ();                           //stc16 初始化
}

volatile U32_T R_LowLevel_T,R_HighLevel_T;
/*****************************************************************/
//STC16 Interrupt
//EntryParameter:NONE
//ReturnValue:NONE
/*****************************************************************/
void STC16IntHandler(void)
{
```

```
    // ISR content ...
      if((ST16->CMISR&ST16_CC1RI)==ST16_CC1RI)
      {
            ST16->CICR = ST16_CC1RI;
            R_LowLevel_T=ST16->CC1R;
            STC16_stop();
            STC16_Channel1_Capture_LoadMode_set(STC16_C1SR_CaptureFall);
            //stc16 Channel0 下降沿捕捉
            STC16_Start();
      }
      if((ST16->CMISR&ST16_CC1FI)==ST16_CC1FI)
      {
            ST16->CICR = ST16_CC1FI;
            R_HighLevel_T=ST16->CC1R;
            STC16_stop();
            STC16_Channel1_Capture_LoadMode_set(STC16_C1SR_CaptureRise);
            //stc16 Channel0 下降沿捕捉
            STC16_Start();
      }
}
```

4.11 TC3/CTC 模块

4.11.1 Sleep 模式定时唤醒配置

功能说明:

开启内部主频 20MHz,并作为系统时钟。
使能外部晶振 32.768K
PB0.0 输出高电平 2s,低电平 2s 方波

操作步骤:

1.SYSCON_CONFIG();函数配置
2.GPIO_CONFIG();函数配置
3.STC16_CONFIG();函数配置
4.主循环程序

C 范例:

```
/*************************************************************/
//CTC Functions
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
//计算公式：Trct_ck * (2^32 - TIMDR)
void CTC_CONFIG(void)
{
    CTC_RESET_VALUE();                                          //CTC 所有寄存器复位赋值
    CTC_SoftReset();                                            //CTC 软件复位
    CTC_Clk_CMD(ENABLE);                                        //使能 CTC_CLK
    CTC_Config(CTC_CLK_Source_set_EMOSC,CTC_BUZZ_Freq_1kHz,CTC_Count_Period_PRDR);      //CTC 使用外部晶振振，
buzz 输出频率为 1Khz，计数周期 2s,若计数周期大于 2s,周期=PRDR*2S
    CTC->PRDR=0X02;
    CTC_INT_CMD(CTC_INT_PEND,ENABLE);                           //使能 CTC PEND 中断
    CTC_Start();                                                //CTC 开启
    CTC_Int_Enable();                                           //使能 CTC 中断向量
    CTC_Wakeup_Enable();                                        //使能 CTC 唤醒
}

/*************************************************************************************/
//APT32F172_init
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************************************/
void APT32F172_init(void)
{
    SYSCON_WDT_CMD(DISABLE);                    //关闭 WDT

    SYSCON->PCER0=0xFFFFFFFF;                   //使能 IP
    SYSCON->PCER1=0xFFFFFFFF;                   //使能 IP
    while(!(SYSCON->PCSR0&0x1));                //判断 IP 是否使能

    SYSCON_Int_Enable();                        //使能 SYSCON 中断向量
    SYSCON->IECR=ISOSC_ST|IMOSC_ST|EMOSC_ST|SYSCLK_ST;
    //使能 ISOSC 时钟稳定中断,使能 IMOSC 时钟稳定中断,使能 EMOSC 时钟稳定中断

    CK_CPU_EnAllNormalIrq();                    //打开全局中断
    SYSCON_CONFIG();                            //syscon 参数 初始化

    CTC_CONFIG ();                              //CTC 初始化
```

```
}

/*************************************************************/
//main
/*************************************************************/
int main(void)
{
        APT32F172_init();
        while(1)
        {
                SYSCON_IWDCNT_Reload();                 //清狗
                GPIO_Write_High(GPIOB0,0);
                PCLK_goto_idle_mode();
                GPIO_Write_Low(GPIOB0,0);
                PCLK_goto_idle_mode();
      }
}
```

4.12 EPWM 模块

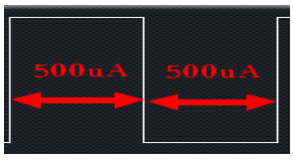
4.12.1 六路 PWM 独立输出

功能说明:

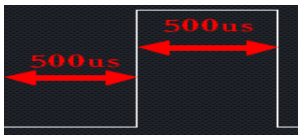
开启内部主频 20MHz,并作为系统时钟。

输出 PWM 波形如图:

1. PA0.9(PWM_X0)

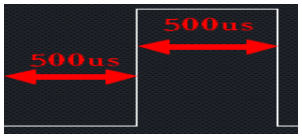


2. PA0.10 (PWM_Y0)

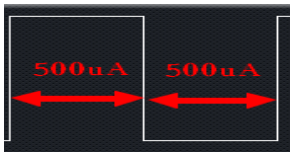


3. PA0.7(PWM_X1)

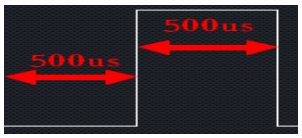
4. PA0.8(PWM_Y1)



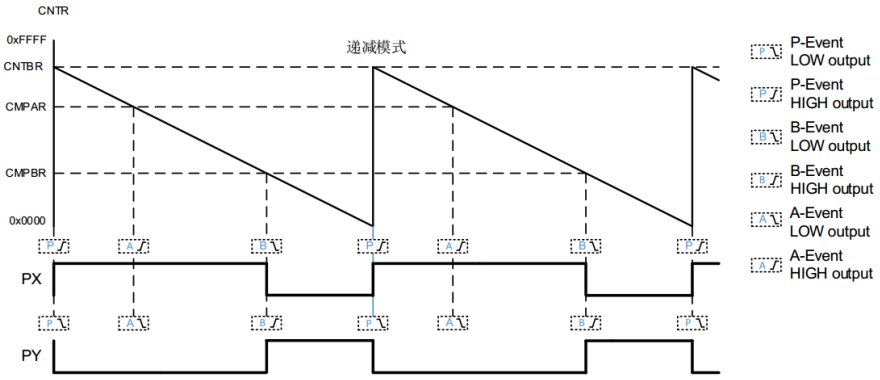
5. PA0.5(PWM_X2)



6. PA0.6(PWM_Y2)



波形生成示意图：



时间优先级列表如下：

优先级	触发事件	可以触发该事件的信号
1	S-Event	StartE（软件使能，或者比较器触发）/ StopE （软件停止，或者Soft Lock）
2	P-Event	PendE（周期结束）
3	C-Event	CENTERE （递增递减计数中间点）
4	B-Event	CMPBDME 和 CMPBUME（计数值与CMPB相等）
5	A-Event	CMPADME 和 CMPAUME（计数值与CMPA相等）

操作步骤：

1. SYSCON_CONFIG();函数配置
2. EPWM_CONFIG ();函数配置

C 范例:

```
/*************************************************************/
//EPWM Init
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
/**********************BLOCK VIEW*************************************/
/*                              - PX ---                                    -- PWM_X    */
/*          PWM Engine --                ---PWM output Control ---                */
/*                              -PY ---                                     --PWM_Y     */
/*************************************************************************/
void EPWM_CONFIG(void)
{
        EPWM_RESET_VALUE();                                                     //EPWM 所有寄存器复位赋值
        EPWM_software_reset();                                          //EPWM 软件复位
        EPWM_IO_Init(PWM_X0,0);                                         //PWM_X0 初始化
        EPWM_IO_Init(PWM_Y0,0);                                         //PWM_Y0 初始化
        EPWM_IO_Init(PWM_X1,0);                                         //PWM_X1 初始化
        EPWM_IO_Init(PWM_Y1,0);                                         //PWM_Y1 初始化
        EPWM_IO_Init(PWM_X2,0);                                         //PWM_X2 初始化
        EPWM_IO_Init(PWM_Y2,0);                                         //PWM_Y2 初始化
      EPWM_CONTER_Configure(EPWM_ContMode_decrease,EPWM_Conter_three,EMP_Overflow_Mode_Continue,1,9);
      //递减计数, EPMW_CLK=PCLK/(2^DIVN)/(DINM+1)=20M/2/(9+1)=1M=1US,  单次触发
      EPWM_PX_PY_Configure(EPWM_P0X,EPWM_StartStopEvent_OutHigh,EPWM_PendEvent_OutHigh,EPWM_CentralEvent_NoChange,
      EPWM_EqCMPAEvent_OutLow,EPWM_EqCMPBEvent_NoChange,1000,500,0);    //P0X CNTR=1000,CMPAR=500,CMPBR=0
      EPWM_PX_PY_Configure(EPWM_P0Y,EPWM_StartStopEvent_OutLow,EPWM_PendEvent_OutLow,EPWM_CentralEvent_NoChange,
      EPWM_EqCMPAEvent_OutHigh,EPWM_EqCMPBEvent_NoChange,1000,500,0);    //P0Y CNTR=1000,CMPAR=500,CMPBR=0
      EPWM_PX_PY_Configure(EPWM_P1X,EPWM_StartStopEvent_OutHigh,EPWM_PendEvent_OutHigh,EPWM_CentralEvent_NoChange,
      EPWM_EqCMPAEvent_OutLow,EPWM_EqCMPBEvent_NoChange,1000,500,0);    //P1X CNTR=1000,CMPAR=500,CMPBR=0
      EPWM_PX_PY_Configure(EPWM_P1Y,EPWM_StartStopEvent_OutLow,EPWM_PendEvent_OutLow,EPWM_CentralEvent_NoChange,
      EPWM_EqCMPAEvent_OutHigh,EPWM_EqCMPBEvent_NoChange,1000,500,0);    //P1Y CNTR=1000,CMPAR=500,CMPBR=0
      EPWM_PX_PY_Configure(EPWM_P2X,EPWM_StartStopEvent_OutHigh,EPWM_PendEvent_OutHigh,EPWM_CentralEvent_NoChange,
      EPWM_EqCMPAEvent_OutLow,EPWM_EqCMPBEvent_NoChange,1000,500,0);    //P2X CNTR=1000,CMPAR=500,CMPBR=0
      EPWM_PX_PY_Configure(EPWM_P2Y,EPWM_StartStopEvent_OutLow,EPWM_PendEvent_OutLow,EPWM_CentralEvent_NoChange,
      EPWM_EqCMPAEvent_OutHigh,EPWM_EqCMPBEvent_NoChange,1000,500,0);    //P2Y CNTR=1000,CMPAR=500,CMPBR=0

      EPWM_OUTPUT_Configure(EPWM_PWM_X0OrPWM_Y0,EPWM_OUTSE_PXPYOutputDirect,EPWM_X_POLARITY_NoChange,EPW
      M_Y_POLARITY_NoChange,EPWM_SRCSEL_PX,0x10,0x10);                  //PWM_X PWM_Y 直接输出模式,输出端电平保持不
      变;RED=EPMW_CLK*16=16us,FED=EPMW_CLK*16=16us
      EPWM_OUTPUT_Configure(EPWM_PWM_X1OrPWM_Y1,EPWM_OUTSE_PXPYOutputDirect,EPWM_X_POLARITY_NoChange,EPW
      M_Y_POLARITY_NoChange,EPWM_SRCSEL_PX,0x10,0x10);                  //PWM_X PWM_Y 直接输出模式,输出端电平保持不
      变;RED=EPMW_CLK*16=16us,FED=EPMW_CLK*16=16us
```

```
    EPWM_OUTPUT_Configure(EPWM_PWM_X2OrPWM_Y2,EPWM_OUTSE_PXPYOutputDirect,EPWM_X_POLARITY_NoChange,EPW
    M_Y_POLARITY_NoChange,EPWM_SRCSEL_PX,0x10,0x10);          //PWM_X PWM_Y 直接输出模式,输出端电平保持不
    变;RED=EPMW_CLK*16=16us,FED=EPMW_CLK*16=16us

    EPWM_Conter0_START();                                     //Count0 开启
     EPWM_Conter1_START();                                    //Count1 开启
     EPWM_Conter2_START();                                    //Count2 开启
}

/*****************************************************************************/
//APT32F172_init                                                      /
//EntryParameter:NONE                                               /
//ReturnValue:NONE                                                  /
/*****************************************************************************/
void APT32F172_init(void)
{
    SYSCON_WDT_CMD(DISABLE);                  //关闭 WDT

    SYSCON->PCER0=0xFFFFFFF;                  //使能 IP
    SYSCON->PCER1=0xFFFFFFF;                  //使能 IP
    while(!(SYSCON->PCSR0&0x1));              //判断 IP 是否使能

    SYSCON_Int_Enable();                      //使能 SYSCON 中断向量
    SYSCON->IECR=ISOSC_ST|IMOSC_ST|EMOSC_ST|SYSCLK_ST;
    //使能 ISOSC 时钟稳定中断,使能 IMOSC 时钟稳定中断,使能 EMOSC 时钟稳定中断

    CK_CPU_EnAllNormalIrq();                  //打开全局中断
    SYSCON_CONFIG();                          //syscon 参数 初始化

    EPWM _CONFIG ();                          //EPWM 初始化
}
```

4.12.2 两路 PWM 互补输出支持死区可调

功能说明:

开启内部主频 20MHz,并作为系统时钟。
输出 PWM 波形如图:

操作步骤:

1. SYSCON_CONFIG();函数配置
2. EPWM_CONFIG ();函数配置

C 范例:

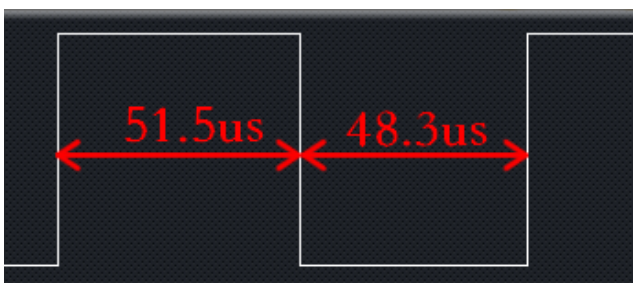
```
/*************************************************************/
//EPWM Init
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
/*********************BLOCK VIEW*************************************/
/*                          - PX ---                                -- PWM_X    */
/*        PWM Engine --              ---PWM output Control ---            */
/*                          -PY ---                                 --PWM_Y     */
/************************************************************************/
void EPWM_CONFIG(void)
{
    EPWM_RESET_VALUE();                                          //EPWM 所有寄存器复位赋值
    EPWM_software_reset();                                       //EPWM 软件复位
    EPWM_IO_Init(PWM_X0,0);                                      //PWM_X0 初始化
    EPWM_IO_Init(PWM_Y0,0);                                      //PWM_Y0 初始化

    EPWM_CONTER_Configure(EPWM_ContMode_decrease,EPWM_Conter_three,EMP_Overflow_Mode_Continue,1,9);
    //递减计数, EPMW_CLK=PCLK/(2^DIVN)/(DINM+1)=20M/2/(9+1)=1M=1US,  单次触发
    EPWM_PX_PY_Configure(EPWM_P0X,EPWM_StartStopEvent_OutHigh,EPWM_PendEvent_OutHigh,EPWM_CentralEvent_NoChange
    ,EPWM_EqCMPAEvent_OutLow,EPWM_EqCMPBEvent_NoChange,1000,500,0);    //P0X:CNTR=1000,CMPAR=500,CMPBR=0
    EPWM_OUTPUT_Configure(EPWM_PWM_X0OrPWM_Y0,EPWM_OUTSE_OutputComplementary,EPWM_X_POLARITY_NoChange,E
    PWM_Y_POLARITY_Negate,EPWM_SRCSEL_PX,0x10,0x10);
    //PX 为输入源互补输出,PWM_X 输出保持不变,PWM_Y 输出反向,
    //RED=EPMW_CLK*16=16us,FED=EPMW_CLK*16=16us

    EPWM_Conter0_START();                                        //Count0 开启
```

```
}

/*******************************************************************************/
//APT32F172_init
//EntryParameter:NONE
//ReturnValue:NONE
/*******************************************************************************/
void APT32F172_init(void)
{
    SYSCON_WDT_CMD(DISABLE);                    //关闭 WDT

    SYSCON->PCER0=0xFFFFFFF;                    //使能 IP
    SYSCON->PCER1=0xFFFFFFF;                    //使能 IP
    while(!(SYSCON->PCSR0&0x1));                //判断 IP 是否使能

    SYSCON_Int_Enable();                        //使能 SYSCON 中断向量
    SYSCON->IECR=ISOSC_ST|IMOSC_ST|EMOSC_ST|SYSCLK_ST;
    //使能 ISOSC 时钟稳定中断,使能 IMOSC 时钟稳定中断,使能 EMOSC 时钟稳定中断

    CK_CPU_EnAllNormalIrq();                    //打开全局中断
    SYSCON_CONFIG();                            //syscon 参数 初始化

    EPWM _CONFIG ();                            //EPWM 初始化
}
```

4.13 CMP 比较器模块

功能说明:

开启内部主频 20MHz,并作为系统时钟。
VDD 接 5V。
比较器负向使用 CPINN0(PB0.01)，输入周期为 100us，振幅为 5V 正玄波。
比较器正向使用 CPINP0(PB0.00)，输入 0.7V 固定电平。
输出波形如下：

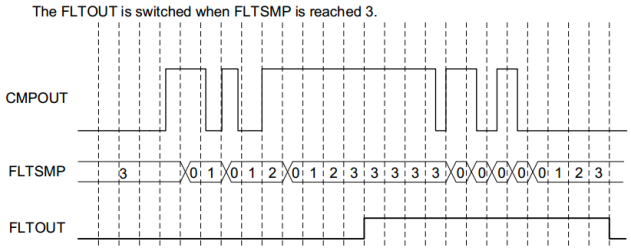


打开比较器滤波使能，选择经过滤波器后输出。

比较器输出状态和输入关系：

输入状态	极性设置	CPxOUT
CxVIN- > CxVIN+	0	0
CxVIN- < CxVIN+	0	1
CxVIN- > CxVIN+	1	1
CxVIN- < CxVIN+	1	0

比较器数字滤波示意图：



操作步骤：

1. SYSCON_CONFIG();函数配置
2. CMP_CONFIG();函数配置

C 范例：

```
/*************************************************************/
//CMP Init
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
void CMP_CONFIG(void)
{
    CMP_RESET_VALUE();                                  //CMP 所有寄存器复位赋值
    CMP_software_reset();                               //CMP 软件复位

    CMP_IO_Init(CPINP0,0);                              //CPINP0 初始化
    CMP_IO_Init(CPINN0,0);                          //CPINN0 初始化

    CMP_IO_Init(CP0_OUT,0);                         //CP0_OUT 初始化
    CMP_INPCRX_Config(CMP0_NUM,248,0);              /比较器负向:CPINN0   比较器正向:CPINP0
    CMP_CLK_CMD(CMP0_NUM,ENABLE);                   //比较器 0 时钟使能
    CMP_CR_Config(CMP0_NUM,NHYST_0mv,PHYST_0mv,POLARITY_0,EVE_SEL_fall_rise,EN_FLTEN,CPOS_1);
    //使能 CMP0 比较器，比较器正向和负向迟滞电压为 0mV，比较器输出不反向，事件触发为上升沿和下降沿，滤波器
    使能，选择经过滤波器后输出
```

```
    CMPX_FLTCR_Config(CMP0_NUM,CMPX_CLK_PCLK,4,199);    //FLT_CK = 20M/(199+1)/2^4=160us; 滤波次数固定为 3 次
    160us*3=480us
    CMP_Open(CMP0_NUM);                                 //比较器 0 打开
}

/*******************************************************************************/
//APT32F172_init                                                          /
//EntryParameter:NONE                                                   /
//ReturnValue:NONE                                                      /
/*******************************************************************************/
void APT32F172_init(void)
{
    SYSCON_WDT_CMD(DISABLE);                //关闭 WDT

    SYSCON->PCER0=0xFFFFFFF;                //使能 IP
    SYSCON->PCER1=0xFFFFFFF;                //使能 IP
    while(!(SYSCON->PCSR0&0x1));            //判断 IP 是否使能

    SYSCON_Int_Enable();                    //使能 SYSCON 中断向量
    SYSCON->IECR=ISOSC_ST|IMOSC_ST|EMOSC_ST|SYSCLK_ST;
    //使能 ISOSC 时钟稳定中断,使能 IMOSC 时钟稳定中断,使能 EMOSC 时钟稳定中断

    CK_CPU_EnAllNormalIrq();                //打开全局中断
    SYSCON_CONFIG();                        //syscon 参数 初始化

    CMP_CONFIG ();                          //CMP 初始化
}
```

4.14 OPAMP 运算放大器模块

功能说明:

开启内部主频 20MHz,并作为系统时钟。
VDD 接 5V。
使用内部增益正向放大，放大倍数 8。
OPAOP(PA1.4)接 0.2V 电压。
OPAOX(PA1.3)用万用表测量电压为 1.6V。

操作步骤:

1.SYSCON_CONFIG();函数配置

2.OPA_CONFIG();函数配置

C 范例:

```
/*************************************************************/
//OPAMP Init
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
void OPAMP_CONFIG(void)
{
    OPA_RESET_VALUE();                                          //OPAx 所有寄存器复位赋值
    OPA_IO_Init(OPA0P,0);                                       //OPA0P 初始化
    OPA_IO_Init(OPA0X,0);                                       //OPA0X 初始化
    OPA_EN_CMD(OPA0_NUM,ENABLE);                                //OPA0  ENABLE
    OPA_Config_Prg(OPA0_NUM,PGAEN_ENABLE,Op_ExtPinConnect_DIS,0,7,0);
    //OPA0,使能增益控制,负向输入口与 PIN 脚连通禁止,正向如输入口为 OPA0P,增益 x8,微调增益 0
}

/*************************************************************************************/
//APT32F172_init                                                                    /
//EntryParameter:NONE                                                              /
//ReturnValue:NONE                                                                  /
/*************************************************************************************/
void APT32F172_init(void)
{
    SYSCON_WDT_CMD(DISABLE);                    //关闭 WDT

    SYSCON->PCER0=0xFFFFFFF;                    //使能 IP
    SYSCON->PCER1=0xFFFFFFF;                    //使能 IP
    while(!(SYSCON->PCSR0&0x1));                //判断 IP 是否使能

    SYSCON_Int_Enable();                        //使能 SYSCON 中断向量
    SYSCON->IECR=ISOSC_ST|IMOSC_ST|EMOSC_ST|SYSCLK_ST;
    //使能 ISOSC 时钟稳定中断,使能 IMOSC 时钟稳定中断,使能 EMOSC 时钟稳定中断

    CK_CPU_EnAllNormalIrq();                    //打开全局中断
    SYSCON_CONFIG();                            //syscon 参数 初始化

    OPAMP _CONFIG ();                           //OPAMP 初始化
}
```

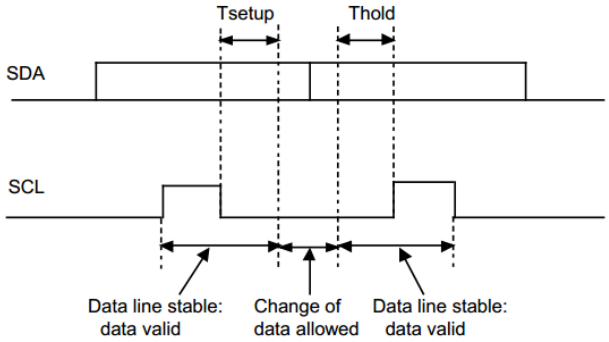
4.15 I2C 通讯模块

4.15.1 主机通讯配置

功能说明:

开启内部主频 20MHz,并作为系统时钟。
设置 SDA(PA0.9)， SCL(PA0.8)且使能漏极开路输出，使能外部上拉 4.7k。
I2C 做主机模式
I2C 速度=PCLK/(PRV+4)
e.g: 20Mhz/(0x040+4)=294kHz
I2C hold/setup time=Htime*PCLK
e.g:0x20* (1/20Mhz)=1.6us

hold/setup time 如下：



I2C 器件地址设置为 0X56
0X01 地址中写 0XAA
0X01 地址中回读 0XAA

操作步骤:

1.SYSCON_CONFIG();函数配置
2.GPIO_CONFIG();函数配置
3.I2C_MASTER_CONFIG();函数配置
4.主函数中对应 0X01 地址进行读写操作。

C 范例:

```
/**********************************************************/
```

```
//GPIO Functions
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
void GPIO_CONFIG(void)
{
    GPIO_OpenDrain_EN(GPIOA0,8);        //PA0.8 漏极开路输出
    GPIO_OpenDrain_EN(GPIOA0,9);        //PA0.9 漏极开路输出
    GPIO_PullHigh_Init(GPIOA0,8);       // PA0.8 上拉使能
    GPIO_PullHigh_Init(GPIOA0,9);       // PA0.9 上拉使能
}

/*************************************************************/
//I2C MASTER Initial
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
void I2C_MASTER_CONFIG(void)
{
   I2C_DeInit();                                    //所有寄存器复位赋值
   I2C_Master_Init(I2C_G0,FAST_MODE,0x040,0x20);    // 管脚配置 SDA(PA0.9)，SCL(PA0.8)，通讯速度配置
}

/*****************************************************************************/
//APT32F172_init                                                  /
//EntryParameter:NONE                                             /
//ReturnValue:NONE                                                /
/*****************************************************************************/
void APT32F172_init(void)
{sm
    SYSCON_WDT_CMD(DISABLE);                    //关闭 WDT

    SYSCON->PCER0=0xFFFFFFF;                    //使能 IP
    SYSCON->PCER1=0xFFFFFFF;                    //使能 IP
    while(!(SYSCON->PCSR0&0x1));                //判断 IP 是否使能

    SYSCON_Int_Enable();                        //使能 SYSCON 中断向量
    SYSCON->IECR=ISOSC_ST|IMOSC_ST|EMOSC_ST|SYSCLK_ST;
    //使能 ISOSC 时钟稳定中断,使能 IMOSC 时钟稳定中断,使能 EMOSC 时钟稳定中断

    CK_CPU_EnAllNormalIrq();                    //打开全局中断
    SYSCON_CONFIG();                            //syscon 参数 初始化
    GPIO_CONFIG();                                  //GPIO 初始化
    I2C_MASTER_CONFIG ();                       //I2C 主机初始化
```

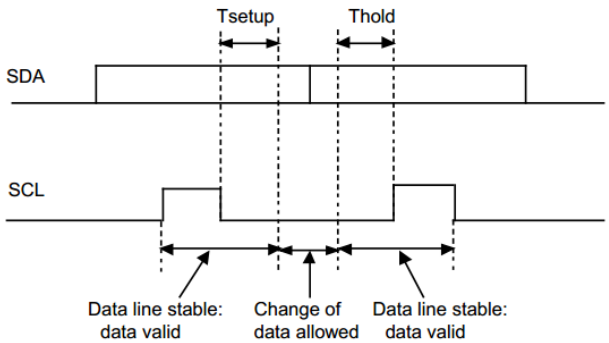
```
}

U32_T R_i2c_read_data;
/*************************************************************/
//main
/*************************************************************/
int main(void)
{
        APT32F172_init();
        while(1)
        {
                SYSCON_IWDCNT_Reload();                 //清狗
                I2C_WriteByte(0x01,0xaa);
                delay_nms(40);                                  //延时 1ms
                R_i2c_read_data= I2C_ReadByte(0x01);
     }
}
```

4.15.2 从机通讯配置

功能说明:

开启内部主频 20MHz,并作为系统时钟。
设置 SDA(PA1.1)， SCL(PA1.2)且使能漏极开路输出，使能上拉 4.7k。
I2C 速度=PCLK/(PRV+4)
e.g: 20Mhz/(0x04F+4)=254kHz
I2C hold/setup time=Htime*PCLK
e.g:0x40* (1/20Mhz)=3.2us

hold/setup time 如下:



I2C 器件地址设置为 0X56
0X01 地址中写 0XAA

0X01 地址中回读 0XAA

操作步骤:

1. SYSCON_CONFIG();函数配置
2. GPIO_CONFIG();函数配置
3. I2C_SLAVE_CONFIG ();函数配置
4. I2C 中断调用 I2C_Slave_Receive();函数。
5. 在主循环中对读取到的数据进行处理。

i. 由主机写入的地址及数据存储在 I2CRdBuffer[BUFSIZE]中，传送给主机的数据存储在 I2CWrBuffer[BUFSIZE]中；

ii. 主机写入数据时，所要操作的寄存器值存储在 I2CRdBuffer[0]中，由 I2CRdBuffer[1]开始为写入的数据值；
eg. 主机向从机寄存器 0x20 写入数据 0x55，操作完成后 I2CRdBuffer[0]=0x20；I2CRdBuffer[1]=0x55；

iii. 主机读取值时，读取值寄存器值与 I2CWrBuffer[BUFSIZE]一一对应；
eg. 读取 0x00 的值，及代表读取 I2CWrBuffer[0]的值。

C 范例:

```
/*************************************************************/
//GPIO Functions
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
void GPIO_CONFIG(void)
{
    GPIO_OpenDrain_EN(GPIOA1,1);        //PA1.1 漏极开路输出
    GPIO_OpenDrain_EN(GPIOA1,2);        //PA1.2 漏极开路输出
    GPIO_PullHigh_Init(GPIOA1,1);       // PA1.1 上拉使能
    GPIO_PullHigh_Init(GPIOA1,2);       // PA1.2 上拉使能
}

/*************************************************************/
//I2C SLAVE Initial
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
void I2C_SLAVE_CONFIG(void)
{
  I2C_DeInit();                                    //所有寄存器复位赋值
```

```
    I2C_Slave_Init(I2C_G2,FAST_MODE,0x4F,0x40,0xAC);
    //从机地址=0xac(8bit)，快速模式, sysclock=20M，通讯速度=sysclock/(0x4f+4)=240K
    I2C_Int_Enable();                                   //I2C 中断向量使能
}

/*****************************************************************/
//I2C Interrupt
//EntryParameter:NONE
//ReturnValue:NONE
/*****************************************************************/
void I2CIntHandler(void)
{
    I2C_Slave_Receive();
}

/*************************************************************************************/
//APT32F172_init                                                          /
//EntryParameter:NONE                                                   /
//ReturnValue:NONE                                                      /
/*************************************************************************************/
void APT32F172_init(void)
{
    SYSCON_WDT_CMD(DISABLE);                //关闭 WDT

    SYSCON->PCER0=0xFFFFFFF;                //使能 IP
    SYSCON->PCER1=0xFFFFFFF;                //使能 IP
    while(!(SYSCON->PCSR0&0x1));            //判断 IP 是否使能

    SYSCON_Int_Enable();                    //使能 SYSCON 中断向量
    SYSCON->IECR=ISOSC_ST|IMOSC_ST|EMOSC_ST|SYSCLK_ST;
    //使能 ISOSC 时钟稳定中断,使能 IMOSC 时钟稳定中断,使能 EMOSC 时钟稳定中断

    CK_CPU_EnAllNormalIrq();                //打开全局中断
    SYSCON_CONFIG();                        //syscon 参数 初始化
    GPIO_CONFIG();                          //GPIO 初始化
    I2C_ SLAVE _CONFIG ();                  //I2C 从机初始化
}

/*****************************************************************/
//main
/*****************************************************************/
int main(void)
{
    APT32F172_init();
```

```
        while(1)
        {
                SYSCON_IWDCNT_Reload();                    //清狗
                //I2CWrBuffer[1] 从机写数据
                //I2CRdBuffer[1] 从机读数据
        }
}
```

4.16 SPI 模块

4.16.1 主机通讯配置

功能说明:

开启内部主频 20MHz,并作为系统时钟。
功能管脚配置:
SPI_NSS=PA0.7; SPI_SCK=PA0.8; SPI_MISO=PA0.9; SPI_MOSI=PA0.10
通讯速度:
FSSPCLKOUT=20M/10=1M
发送接收模式:
发送数据大小为 8BIT; SCK 工作时为高电平; SCK 第一个时钟沿捕捉; 串行正常输出

操作步骤:

1.SYSCON_CONFIG();函数配置
2.SPI_MASTER_CONFIG();函数配置
3.主循环调用 SPI_WRITE_BYTE();发送数据

C 范例:

```
/*************************************************************/
//SPI MASTER Initial
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
void SPI_MASTER_CONFIG(void)
{
```

```
    SPI_DeInit();
    SPI_NSS_IO_Init(1);
    SPI_Master_Init(SPI_G1,SPI_DATA_SIZE_8BIT,SPI_SPO_0,SPI_SPH_0,SPI_LBM_0,SPI_RXIFLSEL_1_8,0,10);
    //选择SPI IO group1;  发送数据大小为8BIT; SCK 工作时为高电平; SCK 第一个时钟沿捕捉;  串行正常输出;  接收占用1/8
    FIFO 中断触发断点; FSSPCLKOUT=20M/10=1M
}

/*****************************************************************************/
//APT32F172_init                                                              /
//EntryParameter:NONE                                                        /
//ReturnValue:NONE                                                           /
/*****************************************************************************/
void APT32F172_init(void)
{
    SYSCON_WDT_CMD(DISABLE);                    //关闭 WDT

    SYSCON->PCER0=0xFFFFFFF;                    //使能 IP
    SYSCON->PCER1=0xFFFFFFF;                    //使能 IP
    while(!(SYSCON->PCSR0&0x1));                //判断 IP 是否使能

    SYSCON_Int_Enable();                        //使能 SYSCON 中断向量
    SYSCON->IECR=ISOSC_ST|IMOSC_ST|EMOSC_ST|SYSCLK_ST;
    //使能  ISOSC 时钟稳定中断,使能 IMOSC  时钟稳定中断,使能 EMOSC 时钟稳定中断

    CK_CPU_EnAllNormalIrq();                    //打开全局中断
    SYSCON_CONFIG();                            //syscon 参数  初始化
    GPIO_CONFIG();                                  //GPIO 初始化
    SPI_MASTER_CONFIG ();                       //SPI 主机初始化
}

U32_T R_i2c_read_data;
/*****************************************************************/
//main
/*****************************************************************/
int main(void)
{
    APT32F172_init();
    while(1)
    {
        SYSCON_IWDCNT_Reload();                 //清狗
        SPI_WRITE_BYTE (0x01);                  //发送 0x01
    }
}
```

4.16.2 从机通讯配置

功能说明:

开启内部主频 20MHz,并作为系统时钟。
功能管脚配置:
SPI_NSS=PA0.7; SPI_SCK=PA0.8; SPI_MISO=PA0.9; SPI_MOSI=PA0.10
通讯速度:
FSSPCLKOUT=20M/10=1M
使能接收中断

操作步骤:

1.SYSCON_CONFIG();函数配置
2.SPI_SLAVE_CONFIG ();函数配置
3.在 SPI 中断接收程序中接收数据。

C 范例:

```
/*****************************************************************/
//SPI SLAVE Initial
//EntryParameter:NONE
//ReturnValue:NONE
/*****************************************************************/
void SPI_SLAVE_CONFIG(void)
{
    SPI_DeInit();
    SPI_NSS_IO_Init(1);
    SPI_Slave_Init(SPI_G1,SPI_DATA_SIZE_8BIT,SPI_RXIFLSEL_1_8,0,12);
    //选择 SPI IO group1; 发送数据大小为 8BIT;接收占用 1/8 FIFO 中断触发断点 ;FSSPCLKOUT=20M/2=10M
    SPI_ConfigInterrupt_CMD(SPI_RXIM,ENABLE);                    //使能 FIFO 接收中断
    SPI_Int_Enable();                                                    //使能 SPI 中断向量
}

/*****************************************************************************/
//APT32F172_init                                                        /
//EntryParameter:NONE                                                 /
//ReturnValue:NONE                                                     /
/*****************************************************************************/
void APT32F172_init(void)
```

```
{
    SYSCON_WDT_CMD(DISABLE);                    //关闭 WDT

    SYSCON->PCER0=0xFFFFFFF;                    //使能 IP
    SYSCON->PCER1=0xFFFFFFF;                    //使能 IP
    while(!(SYSCON->PCSR0&0x1));                //判断 IP 是否使能

    SYSCON_Int_Enable();                        //使能 SYSCON 中断向量
    SYSCON->IECR=ISOSC_ST|IMOSC_ST|EMOSC_ST|SYSCLK_ST;
    //使能 ISOSC 时钟稳定中断,使能 IMOSC 时钟稳定中断,使能 EMOSC 时钟稳定中断

    CK_CPU_EnAllNormalIrq();                    //打开全局中断
    SYSCON_CONFIG();                            //syscon 参数 初始化
    GPIO_CONFIG();                                  //GPIO 初始化
    SPI_ SLAVE _CONFIG ();                      //SPI 主机初始化
}

volatile unsigned int   SPI_DATA[8];
/*****************************************************************/
//SPI Interrupt
//EntryParameter:NONE
//ReturnValue:NONE
/*****************************************************************/
void SPIIntHandler(void)
{
    if((SPI0->MISR&SPI_RXIM)==SPI_RXIM)             //接收 FIFO 中断,FIFO 占用 1/8,1/4,1/2 中断
    {
        SPI0->ICR = SPI_RXIM;
        /*SPI_DATA[0]=SPI0->DR;
        SPI_DATA[1]=SPI0->DR;
        SPI_DATA[2]=SPI0->DR;
        SPI_DATA[3]=SPI0->DR;
        SPI_DATA[4]=SPI0->DR;
        SPI_DATA[5]=SPI0->DR;
        SPI_DATA[6]=SPI0->DR;
        SPI_DATA[7]=SPI0->DR;
        nop;*/
    }
}
```

5 开发注意事项

5.2 EPWM 注意事项

1. EPWM 递增或递减递增模式下，计数开始前 SLPCNTx 的值需设置成跟 CNTR 一样。
2. EPWM 软锁止自增自减模式下，SLCON 的设置必须放在 SLPCMPxR 的后面。
3. EPWM 设置触发延时功能时，TRGTDL 范围在：0~14，若 TRGTDL=15,PWM 输出异常。
4. EPWM_CONTER_Configure （ ）；、 EPWM_PX_PY_Configure （ ）；、EPWM_OUTPUT_Configure（）；在主程序中不要配置 2 次以上,若要重新配置需要先设置 EPWM_software_reset();
5. EPWM 单次触发时，当 CMP 触发 PWM 时，此时 PWM 产生 STOP 事件，当次触发无效。解决办法：避免 CMP 触发周期小于 PWM 周期。
6. EPWM 单次触发时，当 CMP 触发 PWM 时，此时周期值比比较值小，当次触发无效。
7. 解决办法：在 PWM STOP 中断配置周期和比较寄存器值，且将 STOP 中断优先级设置最高。

5.2 GTC 注意事项

1. GTC（TC1）开始计数后，周期和占空比不能设置为 0，否则计数无法再启动。

5.3 ADC 注意事项

1. 读取 OP 输出 ADC 值时，需要在 OP 输出口接 10nF~100nF 电容，或保证 ADC 转速率不超过 200K，否则 ADC 采样不准。
2. ADC 在使用外部参考电压或内部 FVR 时，需要在外部参考电压脚接 104 电容，且该脚位必须设置为 VREF 功能。

5.4 CMP 注意事项

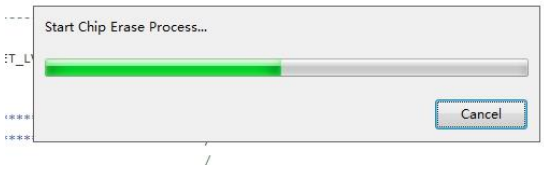
1. CMP 输入电压应小于 VDD-1.5V，否则比较器无法工作，输出默认值。

5.5 功耗注意事项

1. 工作电压 3V,20M 主频，RUN 功耗=4.1mA。
2. 3V 工作电压,40M 主频，RUN 功耗=6.7mA。
3. 3.3V 工作电压,deep sleep 功耗=0.7uA。
4. 3.3V 工作电压,deep sleep 功耗=37uA(使能 LVR)。
5. 3.3V 工作电压,deep sleep 功耗=30uA(使能 WDT)。
6. 工作电压,sleep 模式下 CTC 外挂 32.768K 唤醒 功耗=29uA(WDT LVR 禁止)。
7. 工作电压，sleep 模式下，ST16 唤醒模式，主频 20M DIV4 功耗=0.84mA(WDT LVR STCLK 关闭)。

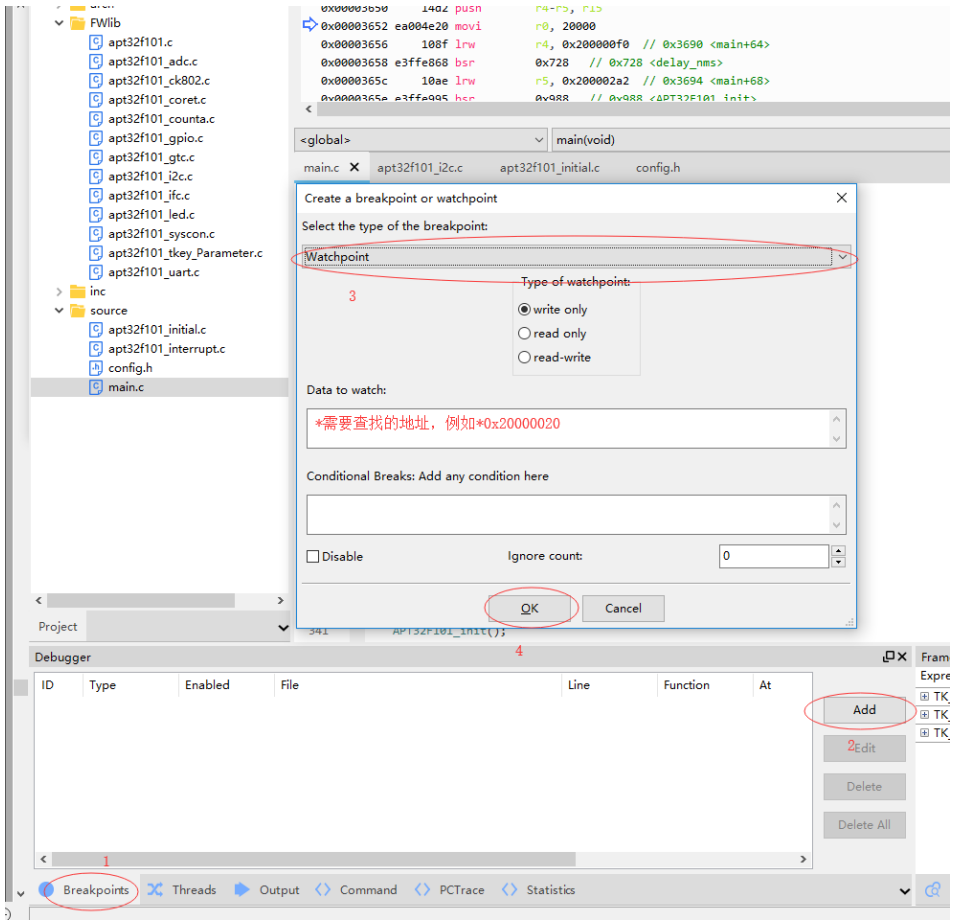
5.6 CDK 使用注意事项

1. 1.10.2 版本 debug 模式下，出现 debug 停留在下面窗口。



由于该版本默认仿真速度为 100KHz，需要调整为 1000KHz 后，debug 速度正常。步骤如下：
Project Setting->Debug->ICE Configration->ICE Clock 1000

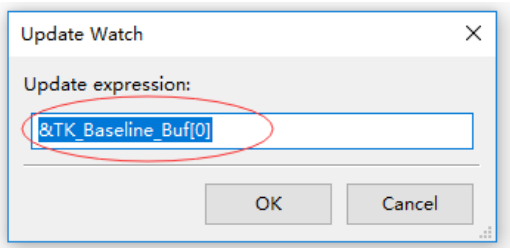
2. 添加 Watchpoint



添加完成后，CDK 仿真全速跑，当设置的地址里面数据发生改变，CDK 立刻停止，且停在改变变量的代码处。

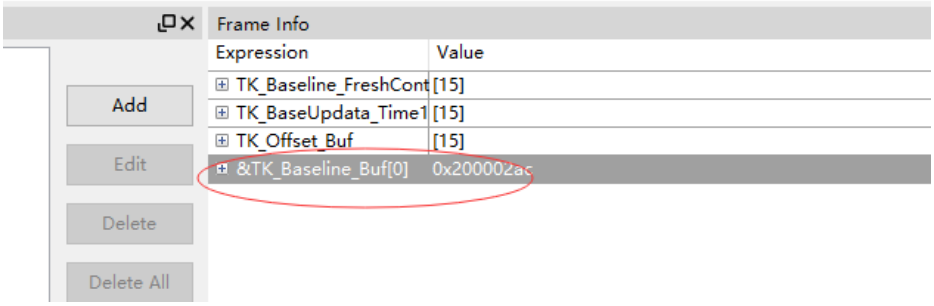
该功能主要用于代码中异常情况下，更改变量的 debug 方式。

3. 查找对应变量的 sram 地址，如下图：

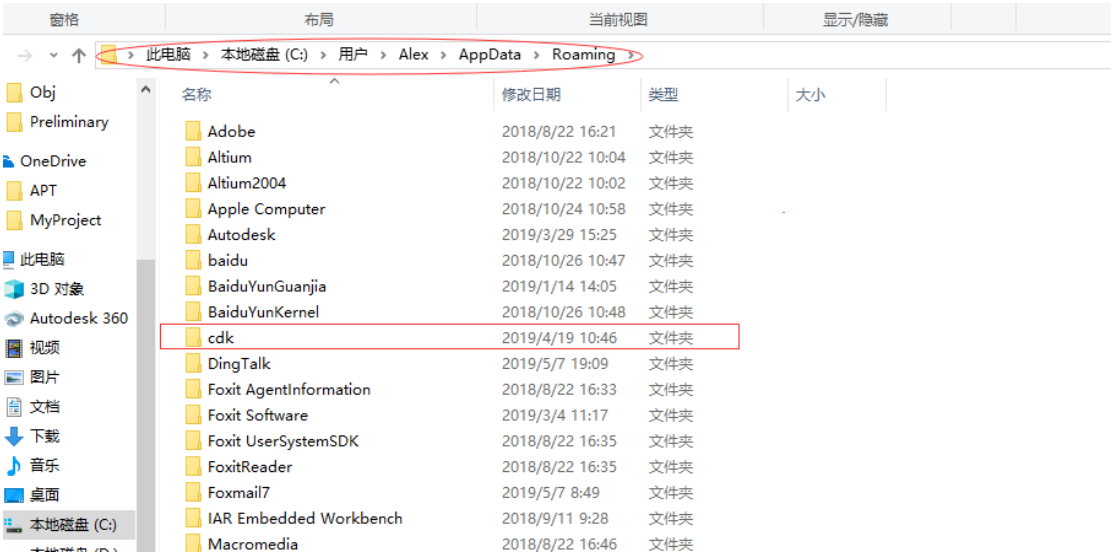


/

/



4. CDK 无法打开，重新安装也不能打开
   1. 打开文件夹隐藏功能。
   2. 修改 C 盘->用户->客户名->AppData->Roaming，修改 CDK 文件名。



6 改版历史

版本	修改日期	修改概要
V1.0	2019-11-11	初版