文档版本	V1.0.0
发布日期	20221026

APT32F110x 基于 CSI 库 BT 应用指南

目录

1 概述

本文介绍了在APT32F110x中使用BT的应用范例。

2. 适用的硬件

该例程使用于 APT32F110X 系列学习板

3. 应用方案代码说明

基于 APT32F110x 完整的 CSI 库文件系统，可以对 BT 进行配置。

3.1 BT 模块介绍

基本型计数器（Basic Timer）是一个 16 位计数器。Timer 工作在递增模式下，并支持自动重载功能。Basic Timer 可提供基础定时/计数功能和简单的 PWM 波形输出。

- **主要特性：**

16位可编程递增计数器。

16位预设计数器时钟分频器(支持On-the-fly修改配置)。

一个比较值寄存器，支持PWM波形输出。

支持通过ETCB进行硬件自动同步触发和外部计数。

- **管脚描述：**

管脚名称	功能描述
BT_OUT	PWM 波形输出

图 3.1.1 BT 相关管脚描述

- **模块框图：**

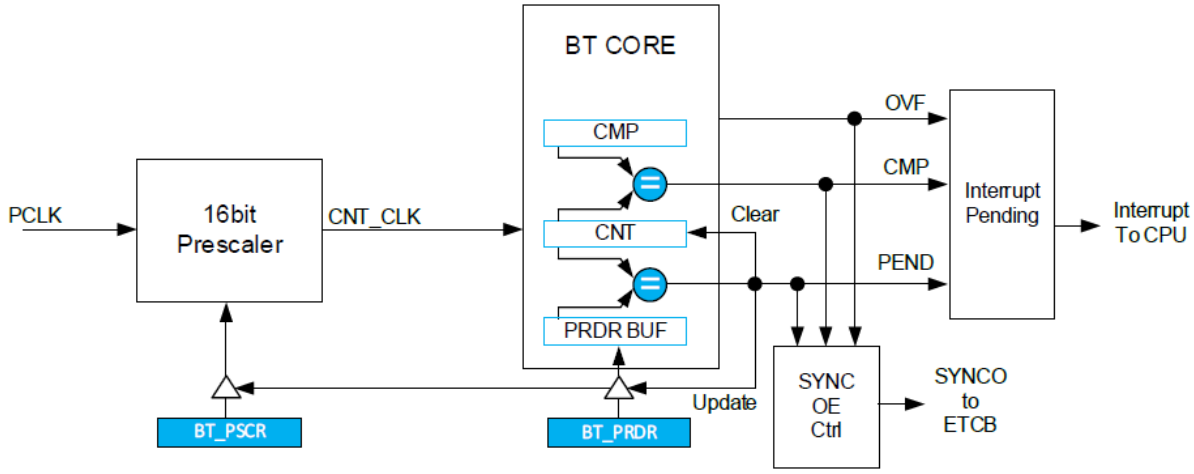

图 3.1.2 模块框图

3.2 定时中断配置

系统时钟选择内部 48MHz，利用 BT 定时使 GPIOA0.4 输出周期为 20ms，占空比为 50%的波形。(可在 user_demo.c 文件中的 bt_timer_demo ()进行配置)

```
int bt_timer_demo(void)
{
    int iRet = 0;
    csi_bt_timer_init(BT0, 10000);
    csi_bt_start(BT0);
    return iRet;
}


int pin_output_demo(void)
{
    int iRet = 0;
    csi_pin_set_mux(PA04,PA04_OUTPUT);
    csi_pin_set_high(PA04);
    mdelay(100);
    return iRet;
}


__attribute__((weak)) void bt_irqhandler(csp_bt_t *ptBtBase)
{
    // ISR content ...
```

```
    volatile uint32_t wMisr = csp_bt_get_isr(ptBtBase);

    if(wMisr & BT_PEND_INT)                         //PEND interrupt
    {
        csp_bt_clr_isr(ptBtBase, BT_PEND_INT);

        csi_pin_toggle(PA04);
    }

    if(wMisr & BT_CMP_INT)                          //CMP interrupt
    {
        csp_bt_clr_isr(ptBtBase, BT_CMP_INT);
    }

    if(wMisr & BT_OVF_INT)                          //OVF interrupt
    {
        csp_bt_clr_isr(ptBtBase, BT_OVF_INT);
    }

    if(wMisr & BT_EVTRG_INT)                    //EVTRG interrupt
    {
        csp_bt_clr_isr(ptBtBase, BT_EVTRG_INT);
    }
}


int main()
{
    system_init();
    board_init();

    user_demo();                                    //demo

    while(1)
    {
    }

    return 0;
}
```

- 代码说明:

csi_bt_start(); ------ 启动 BT。

csi_bt_timer_init(); ------ 定时功能初始化，默认使用计数器周期结束中断(PEND)。

- **函数参数说明：**

csi_bt_start(csp_bt_t *ptBtBase);

ptBtBase：BT 寄存器结构体指针，指向 BT 基地址。

csi_bt_timer_init(csp_bt_t *ptBtBase, uint32_t wTimeOut)；

ptBtBase：BT寄存器结构体指针，指向BT基地址。

wTimeOut：计数器溢出时间，即定时时间，单位 us。

- **测试波形：**

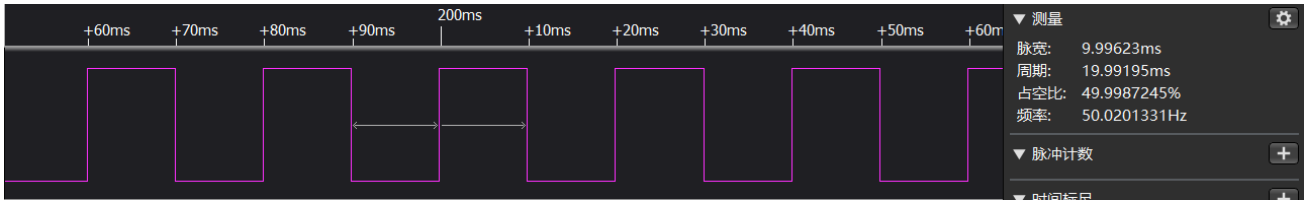

图 3.2.1 定时输出波形

3.3 PWM 输出配置

系统时钟选择内部 48MHz，启用 BT 的 PWM 模式，通过 PA1.9，输出周期为 1ms，占空比为 20%。(可在 user_demo.c 文件中的 bt_pwm_demo ()进行配置)

```
int bt_pwm_demo(void)
{
    int iRet = 0;
    csi_bt_pwm_config_t tPwmCfg;

    csi_pin_set_mux(PA19, PA19_BT0_OUT);

    tPwmCfg.byIdleLevel = BT_PWM_IDLE_HIGH;
    tPwmCfg.byStartLevel= BT_PWM_START_HIGH;
    tPwmCfg.byDutyCycle = 50;
    tPwmCfg.wFreq               = 1;
    tPwmCfg.byInt               = BT_INTSRC_NONE;

    csi_bt_pwm_init(BT0, &tPwmCfg);
    csi_bt_start(BT0);

    csi_bt_pwm_updata(BT0, 1000, 20);
```

```
    return iRet;
}

int main()
{
    system_init();
    board_init();

    user_demo();            //demo

    while(1)
    {

    }

    return 0;
}
```

- **代码说明：**

  csi_bt_pwm_init(); ------ PWM 输出初始化。

  csi_bt_pwm_updata(); ------ 更新 PWM 输出频率和占空比。

  csi_pin_set_mux(); ------ 设置 PIN 脚功能。

- **函数参数说明：**

  csi_bt_pwm_init(csp_bt_t *ptBtBase, csi_bt_pwm_config_t *ptBtPwmCfg);

  ptBtBase：BT寄存器结构体指针，指向BT基地址。

  ptBtPwmCfg： PWM 输出配置结构体指针；

  ptBtPwmCfg->byIdleLevel: PWM 输出空闲电平；

  ptBtPwmCfg->byStartLevel: PWM 输出起始电平；

  ptBtPwmCfg->byDutyCycle: PWM 输出占空比；

  ptBtPwmCfg->wFreq: PWM 输出频率；

  ptBtPwmCfg->byInt: PWM 中断配置。

csi_bt_pwm_updata(csp_bt_t *ptBtBase, uint32_t wFreq, uint8_t byDutyCycle);

ptBtBase：BT寄存器结构体指针，指向BT基地址。

wFreq：PWM输出频率，单位Hz。

byDutyCycle：PWM 输出占空比(0 < byDutyCycle < 100)。

csi_pin_set_mux(pin_name_e ePinName, pin_func_e ePinFunc)；

ePinName：PIN脚名字。

ePinFunc：PIN 脚功能。

- **测试波形：**

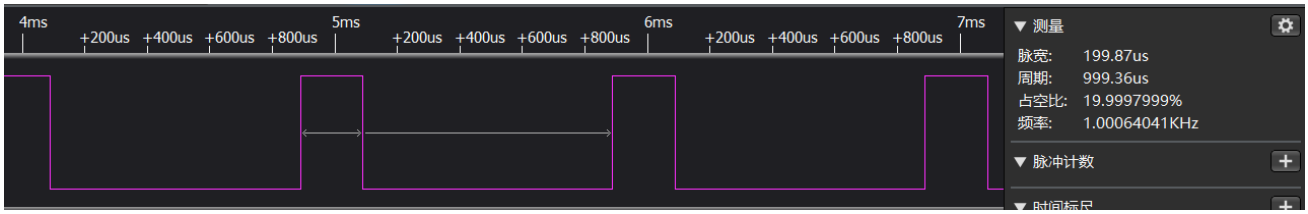

图 3.3.1 PWM 输出

4. 程序下载和运行

1．将目标板与仿真器连接，分别为 VDD、SCLK、SWIO、GND。

2. 程序编译后仿真运行。

3. 通过示波器或逻辑分析仪查看输出波形，如上图 3.2.1、图 3.3.1 所示。