文档版本	V1.0.0
发布日期	20221026

APT32F110x 基于 CSI 库 EPT 应用指南

目录

1 概述 1

2. 适用的硬件 1

3. 应用方案代码说明 1

3.1 EPT 概述 1

3.2 PWM 互补输出带死区 2

3.3 输入捕获 9

3.4 四路独立 PWM 12

4. 程序下载和运行 15

1 概述

本文介绍了在APT32F110x中EPT应用

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 的库文件系统，进行配置 EPT。

3.1 EPT 概述

- **硬件配置：**

EPT 模块是一个增强型通用定时器。具有自动重载寄存器，有可编程的死区控制单元。支持捕获和波形发生器模式，有 7 个 TIMER 输出通道，支持 4 路独立输出或者 3 组互补输出。支持事件计数触发机制。

需要注意 EPT 中很多寄存器是由两个物理寄存器组成：活动寄存器（Active）和影子寄存器（Shadow），它们共享同一个物理访问地址。每个影子寄存器只有在特定条件满足时，才会更新到活动寄存器中。更新条件均可以独立设置。

- PWM 输出管脚：

管脚名称	突发计数模式	波形发生器： 单波形输出模式	波形发生器： 双波形输出模式
CHAX	时钟控制使能	输出波形	输出波形
CHAY	NA	输出波形	输出波形
CHBX	时钟控制使能	输出波形	输出波形
CHBY	NA	输出波形	输出波形
CHCX	NA	输出波形	输出波形
CHCY	NA	输出波形	输出波形
CHD	NA	输出波形	输出波形

图 3.1.1 输出管脚

- 模块框图：

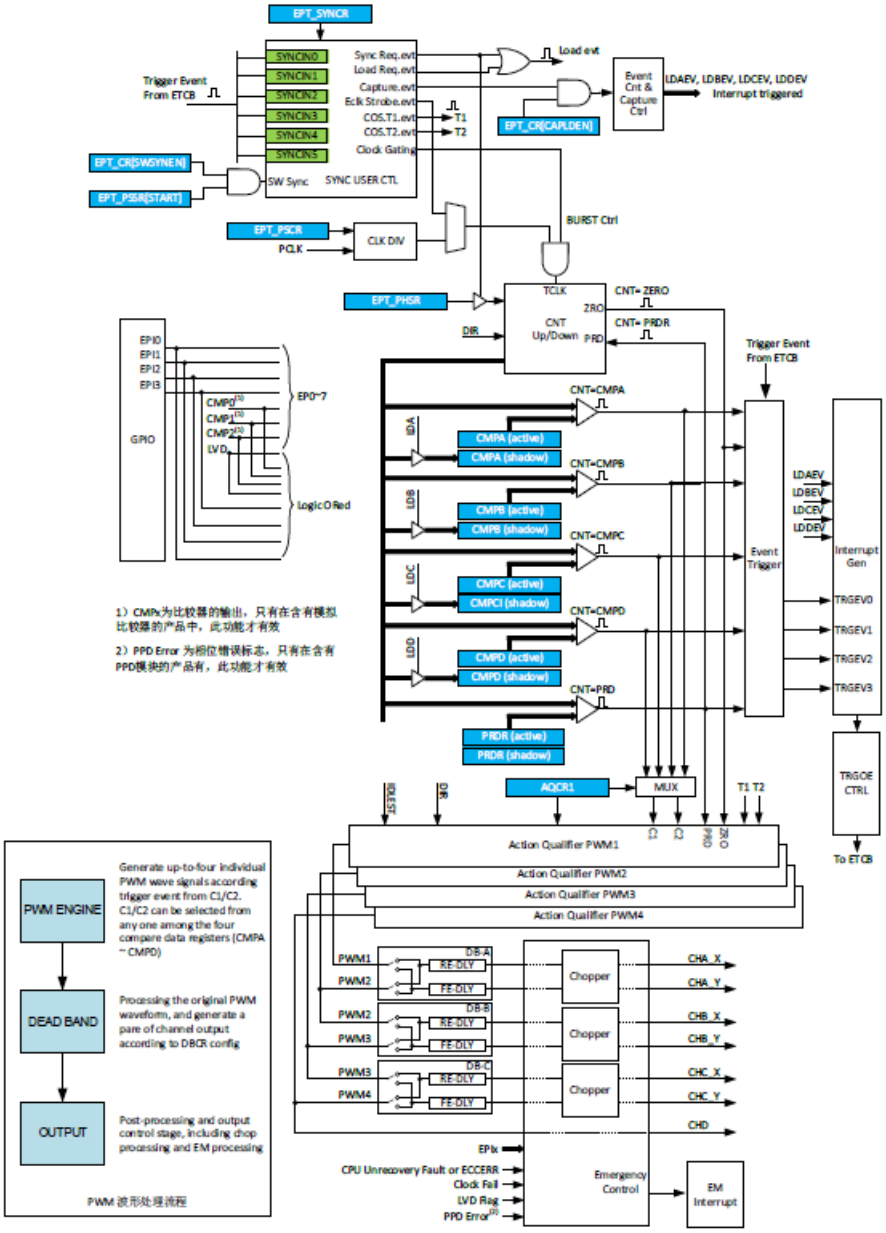

图 3.1.2 模块框图

3.2 PWM 互补输出带死区

系统时钟选择内部 48Mhz，输出周期为 100us，占空比为 50%,死区为 500ns。

PA1.3->CHAX 、PA1.0->CHAY

PA1.4->CHBX 、PA1.1->CHBY

PA1.5->CHCX 、PA1.2->CHCY

(可在 user_demo.c 文件中的 ept_pwm_dz_demo()进行配置)

```
int ept_pwm_dz_demo(void)
{
    int iRet = 0;

//-------------------------------------------------------------------------------------------------------------------
    csi_pin_set_mux(PA13, PA13_EPT_CHAX);                                   //
    csi_pin_set_mux(PA14, PA14_EPT_CHBX);                                   //
    csi_pin_set_mux(PA15, PA15_EPT_CHCX);                                   //
    csi_pin_set_mux(PA16, PA16_EPT_CHD);                                //

    csi_pin_set_mux(PA10, PA10_EPT_CHAY);                                   //
    csi_pin_set_mux(PA11, PA11_EPT_CHBY);                               //
    csi_pin_set_mux(PA12, PA12_EPT_CHCY);
//-------------------------------------------------------------------------------------------------------------------
    csi_ept_config_t tPwmCfg;
    tPwmCfg.byWorkmod          = EPT_WAVE;                  //WAVE or CAPTURE     //计数或捕获
    tPwmCfg.byCountingMode   = EPT_UPDNCNT;                 //CNYMD   //计数方向
    tPwmCfg.byOneshotMode     = EPT_OP_CONT;                 //OPM      //单次或连续(工作方式)
    tPwmCfg.byStartSrc         = EPT_SYNC_START;                    //软件使能同步触发使能控制（RSSR 中 START 控制位）//启动方式
    tPwmCfg.byPscld             = EPT_LDPSCR_ZRO;       //PSCR(分频)活动寄存器载入控制。活动寄存器在配置条件满足时，从影子寄存器载入更新值

    tPwmCfg.byDutyCycle        = 50;                                    //pwm ouput duty cycle//PWM 初始值
    tPwmCfg.wFreq                  = 10000;                                 //pwm ouput frequency
    tPwmCfg.wInt                   = EPT_INTSRC_TRGEV0;             //interrupt
    csi_ept_config_init(EPT0, &tPwmCfg);

//  csi_ept_set_evtrg(EPT0, EPT_TRG_OUT0, EPT_TRGSRC_PE1);     //EP1 用 trg0 输出，
//  csi_ept_set_sync (EPT0, EPT_TRG_SYNCEN2, EPT_TRG_CONTINU,EPT_AUTO_REARM_ZRO);
//  csi_ept_int_enable(EPT0, EPT_INTSRC_TRGEV0,true);
//-------------------------------------------------------------------------------------------------------------------
    csi_ept_pwmchannel_config_t   tEptchannelCfg;
    tEptchannelCfg.byActionZro     =    LO;
    tEptchannelCfg.byActionPrd     =    NA;
    tEptchannelCfg.byActionC1u     =    HI;
    tEptchannelCfg.byActionC1d     =    LO;
    tEptchannelCfg.byActionC2u     =    NA;
    tEptchannelCfg.byActionC2d     =    NA;
    tEptchannelCfg.byActionT1u     =    LO;
    tEptchannelCfg.byActionT1d     =    LO;
    tEptchannelCfg.byActionT2u     =    NA;
    tEptchannelCfg.byActionT2d     =    NA;
    tEptchannelCfg.byChoiceC1sel   =    EPT_CMPA;
```

```
    tEptchannelCfg.byChoiceC2sel   =    EPT_CMPA;
    csi_ept_channel_config(EPT0, &tEptchannelCfg,   EPT_CHANNEL_1);//channel
    csi_ept_channel_config(EPT0, &tEptchannelCfg,   EPT_CHANNEL_2);
    csi_ept_channel_config(EPT0, &tEptchannelCfg,   EPT_CHANNEL_3);
//  csi_ept_channel_config(EPT0, &tEptchannelCfg,   EPT_CHANNEL_4);
//csp_ept_set_aqtscr(EPT0,EPT_T1,EP1);//波形输出 T 事件选择
//-----------------------------------------------------------------------------------------------------------------
    csi_ept_deadzone_config_t   tEptDeadZoneTime;
    tEptDeadZoneTime.byDcksel                    = EPT_DB_DPSC;       //
    tEptDeadZoneTime.hwDpsc                      =   0;               //FDBCLK = FHCLK / (DPSC+1)
    tEptDeadZoneTime.hwRisingEdgereGister    = 500;                   //上升沿-ns
    tEptDeadZoneTime.hwFallingEdgereGister   = 500;                   //下降沿-ns
    tEptDeadZoneTime.byChaDedb                   = DB_AR_BF;          //不使用死区双沿
    tEptDeadZoneTime.byChbDedb                   = DB_AR_BF;
    tEptDeadZoneTime.byChcDedb                   = DB_AR_BF;
    csi_ept_dz_config(EPT0,&tEptDeadZoneTime);

    tEptDeadZoneTime.byChxOuselS1S0        = E_DBOUT_AR_BF;       //使能通道 A 的上升沿延时，使能通道 B 的下降沿延时
    tEptDeadZoneTime.byChxPolarityS3S2     = E_DB_POL_B;          //通道 A 和通道 B 延时输出电平是否反向
    tEptDeadZoneTime.byChxInselS5S4        = E_DBCHAIN_AR_AF;     //PWMA 作为上升沿和下降沿延时处理的输入信号
    tEptDeadZoneTime.byChxOutSwapS8S7      = E_CHOUTX_OUA_OUB;    //OUTA=通道 A 输出，OUTB=通道 B 输出
      csi_ept_channelmode_config(EPT0,&tEptDeadZoneTime,EPT_CHANNEL_1);
    csi_ept_channelmode_config(EPT0,&tEptDeadZoneTime,EPT_CHANNEL_2);
      csi_ept_channelmode_config(EPT0,&tEptDeadZoneTime,EPT_CHANNEL_3);
//-----------------------------------------------------------------------------------------------------------------
    csi_ept_start(EPT0);//start   timer

      while(1){
          }
    return iRet;
}
```

- **代码说明：**

1. **csi_ept_config_init():** ----- ept 基本配置
2. **csi_ept_channel_config():** ----- 设置通道 PWM1、PWM2、PWM3、PWM4 的波形
3. **csi_ept_dz_config():** ----- 死区功能基本配置
4. **csi_ept_channelmode_config():** ----- 死区通道配置，包含的通道有：

CHANNEL_A、CHANNEL_B、CHANNEL_C

5. **csi_ept_start():** ----- 启动定时器

- **函数参数说明：**

1. **csi_ept_config_init(csp_ept_t *ptEptBase, csi_ept_config_t *pteptPwmCfg);**

   ptEptBase: ept 模块基地址

   pteptPwmCfg: 定时器工作参数结构体指针

   pteptPwmCfg->byWorkmode: 工作模式，0：捕捉模式、1：波形输出模式

   pteptPwmCfg->byCountingMode: 计数方向，00b：递增方向、01b：递减方向、10b：递增递减方向

   pteptPwmCfg->byOneshotMode: 计数器触发工作模式选择，0：连续计数模式、1：单次触发模式

   pteptPwmCfg->byStartSrc: 软件使能同步触发使能控制，0：设置 SW START 控制只用于启动、1：设置 SW START 控制用于启动和以产生一次外部触发的方式重新启动

   pteptPwmCfg->byPscld: PSCR 活动寄存器载入控制。活动寄存器在配置条件满足时，从影子寄存器载入更新值

   pteptPwmCfg->byDutyCycle: PWM 初始占空比

   pteptPwmCfg->wFreq: PWM 频率

   pteptPwmCfg->wInt: 中断控制

2. **csi_ept_channel_config(csp_ept_t *ptEptBase, csi_ept_pwmchannel_config_t *tPwmCfg, csi_ept_channel_e channel);**

   ptEptBase: 解析同上

   tPwmCfg: 配置参数结构体指针

tPwmCfg-> byActionZro: 当 CNT 值等于 0 时, 在通道 PWM 上做出的波形动作定义。在递增递减模式时，当计数器值等于 0 时，计数方向为递增模式。0：不动作（过滤该处理事件）、1：清除输出（低电平）、2：置为输出（高电平）、3：反向（翻转），以下参数动作保持一致

tPwmCfg-> byActionPrd: 当 CNT 值等于 PRDR 时，在通道 PWM 上做出的波形输出动作定义。在递增递减模式时，当计数器值等于 PRDR 时，计数方向为递增模式

tPwmCfg-> byActionC1u: 当 CNT 值等于 C1 时，且此时计数方向为递增时，在通道 PWM 上做出的波形输出动作定义

tPwmCfg-> byActionC1d: 当 CNT 值等于 C1 时，且此时计数方向为递减时，在通道 PWM 上做出的波形输出动作定义

tPwmCfg-> byActionC2u: 当 CNT 值等于 C2 时，且此时计数方向为递增时，在通道 PWM 上做出的波形输出动作定义

tPwmCfg-> byActionC2d: 当 CNT 值等于 C2 时，且此时计数方向为递减时，在通道 PWM 上做出的波形输出动作定义

tPwmCfg-> byActionT1u: 当 T1 事件发生时, 且此时计数方向为递增时, 在通道 PWM 上做出的波形输出动作定义

tPwmCfg-> byActionT1d: 当 T1 事件发生时, 且此时计数方向为递减时, 在通道 PWM 上做出的波形输出动作定义

tPwmCfg-> byActionT2u: 当T2事件发生，且此时计数方向为递增时，在通道PWM上做出的波形输出动作定义

tPwmCfg-> byActionT2d: 当 T2 事件发生，且此时计数方向为递减时，在通道 PWM 上做出的波形输出动作定义

tPwmCfg-> byChoiceC1sel: C1 比较值的数据源选择。0：CMPA 寄存器作为 C1 的数据源、1：CMPB 寄存器作为 C1 的数据源、2：CMPC 寄存器作为 C1 的数据源、3：CMPD 寄存器作为 C1 的数据源

tPwmCfg-> byChoiceC2sel: C2 比较值数据源选择。0：CMPA 寄存器作为 C2 的数据源、1：CMPB 寄存器作为 C2 的数据源、2：CMPC 寄存器作为 C2 的数据源、3：CMPD 寄存器作为 C2 的数据源

channel: 通道选择

3. **csi_ept_dz_config(csp_ept_t *ptEptBase, csi_ept_deadzone_config_t *tCfg);**

ptEptBase: 解析同上

tCfg: 死区配置参数结构体指针

tCfg-> byDcksel: 半周期时钟使能控制。0:死区控制延时计数器以 TCLK 频率工作、1:死区控制延时计数器以 HCLK/(DPSC+1)工作

tCfg-> hwDpsc: 时钟分频控制

tCfg-> hwRisingEdgereGister: 上升沿延时数值

tCfg-> hwFallingEdgereGister: 下降沿延时数值

tCfg-> byChaDedb: 在 PWM1 DBCOUTY 上选择死区双沿模式。0：不使用死区双沿、1：使用死区双沿

tCfg-> byChbDedb: 在 PWM2 DBCOUTY 上选择死区双沿模式。0：不使用死区双沿、1：使用死区双沿

tCfg-> byChcDedb: 在 PWM3 DBCOUTY 上选择死区双沿模式。0：不使用死区双沿、1：使用死区双沿

tCfg-> byChxOuselS1S0: 死区输出配置（S1、S0 开关）。

0：bypass 死区控制，X 通道输出 PWM1，Y 通道输出 PWM2、

1：X 通道输出 PWM1，使能 Y 通道下降沿延时、

2：使能 X 通道的上升沿延时，Y 通道输出 PWM2、

3：使能 X 通道的上升沿延时，使能 Y 通道的下降沿延时。

tCfg-> byChxPolarityS3S2: 输出极性控制（S3、S2 开关）。0：X 通道和 Y 通道延时输出不反向、1：X 通道的延时输出反向、2：Y 通道的延时输出反向、3：X 通道和 Y 通道延时输出全部反向

tCfg-> byChxInselS5S4: 延时模块输入选择（S5、S4开关）。在经典死区控制模式下，上升沿延时和下降沿延时都选择同一个输入信号进行处理。

0：PWM1作为上升沿和下降沿延时处理的输入信号

1：PWM2作为上升沿延时输入，PWM1作为下降沿延时输入

2：PWM1作为上升沿延时输入，PWM2作为下降沿延时输入

3：PWM2 作为上升沿和下降沿延时处理的输入信号

tCfg-> byChxOutSwapS8S7: 死区输出交换控制（S7、S8 开关）。

0：OUTX=X通道输出，OUTY=Y通道输出

1：OUTX=Y通道输出，OUTY=Y通道输出

2：OUTX=X通道输出，OUTY=X通道输出

3：OUTX=Y 通道输出，OUTY=X 通道输出

4. **csi_ept_channelmode_config(csp_ept_t *ptEptBase,csi_ept_deadzone_config_t *tCfg,csi_ept_channel_e byCh);**

ptEptBase: 解析同上

tCfg: 解析同上

byCh: 通道选择

5. **csi_ept_start(csp_ept_t *pteptBase);**

   pteptBase： 解析同上

- **输出波形：**

图 3.2.1 PWM 互补波形

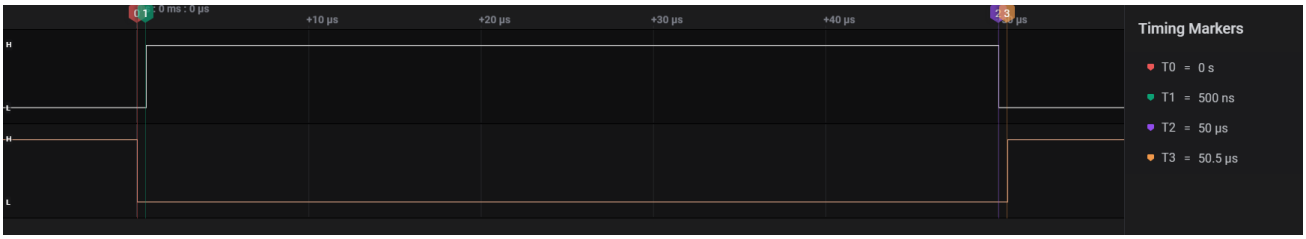

图 3.2.2 死区输出波形

3.3 输入捕获

系统时钟选择内部 48Mhz，设置 PA0.1 的下降沿外部中断事件，通过 ETCB 触发 TIMER 的捕获操作。(可在 user_demo.c 文件中的 ept_capture_demo ()进行配置)

- **编程要点：**

  1. 配置 GPIO 的外部触发

  2. 设置 ETCB 触发

  3. 配置 EPT 捕获模式

4. 配置 EPT 中断

5. 配置 SYSCON 中的事件触发选择

```
int ept_capture_demo(void)
{
    int iRet = 0;

    volatile uint8_t ch;
    csi_pin_set_mux(PA01,PA01_INPUT);
    csi_pin_pull_mode(PA01, GPIO_PULLUP);                                   //PA01 上拉
    csi_pin_irq_mode(PA01,EXI_GRP1, GPIO_IRQ_FALLING_EDGE);                 //PA01 下降沿产生中断
    csi_exi_set_evtrg(EXI_TRGOUT1, TRGSRC_EXI1, 1);
//-------------------------------------------------------------------------------------------------------------------
    csi_etb_config_t tEtbConfig;                        //ETB 参数配置结构体
    tEtbConfig.byChType   = ETB_ONE_TRG_ONE;            //单个源触发单个目标
    tEtbConfig.bySrcIp    = ETB_EXI_TRGOUT1 ;    //...作为触发源
    tEtbConfig.byDstIp    =   ETB_EPT0_SYNCIN2;         //EPT0 同步输入 2 作为目标事件
    tEtbConfig.byTrgMode = ETB_HARDWARE_TRG;
    csi_etb_init();
    ch = csi_etb_ch_alloc(tEtbConfig.byChType);       //自动获取空闲通道号,ch >= 0 获取成功
//  if(ch < 0)return -1;                                //ch < 0,则获取通道号失败
    iRet = csi_etb_ch_config(ch, &tEtbConfig);
//-------------------------------------------------------------------------------------------------------------------
    csi_ept_captureconfig_t tPwmCfg;
    tPwmCfg.byWorkmod          = EPT_CAPTURE;                   //WAVE or CAPTURE     //计数或捕获
    tPwmCfg.byCountingMode   = EPT_UPCNT;                      //CNYMD   //计数方向
    tPwmCfg.byOneshotMode    = EPT_OP_CONT;                    //OPM     //单次或连续(工作方式)
    tPwmCfg.byStartSrc         = EPT_SYNC_START;                   //软件使能同步触发使能控制（RSSR 中 START 控制位）//启动方式
    tPwmCfg.byPscId             = EPT_LDPSCR_ZRO;                 //PSCR(分频)活动寄存器载入控制。活动寄存器在配置条件满足时，从影子寄存器载入更新值
    tPwmCfg.byCaptureCapmd   = EPT_CAPMD_CONT;                  //0:连续捕捉模式     1h：一次性捕捉模式
    tPwmCfg.byCaptureStopWrap=4-1;                             //Capture 模式下，捕获事件计数器周期设置值
    tPwmCfg.byCaptureLdaret   =0;                              //CMPA 捕捉载入后，计数器值计数状态控制位(1h：CMPA 触发后，计数器值进行重置;0h：CMPA 触发后，计数器值不进行重置)
    tPwmCfg.byCaptureLdbret   =0;
    tPwmCfg.byCaptureLdcret   =0;
    tPwmCfg.byCaptureLddret   =0;
    tPwmCfg.wInt                =EPT_INTSRC_CAPLD3;            //interrupt

    csi_ept_capture_init(EPT0, &tPwmCfg);
//-------------------------------------------------------------------------------------------------------------------
```

```
    csi_ept_set_sync (EPT0, EPT_TRG_SYNCEN2, EPT_TRG_CONTINU,EPT_AUTO_REARM_ZRO);
//-------------------------------------------------------------------------------------------------------------------

    csi_ept_start(EPT0);//start   timer
      while(1){

                mdelay(200);
//
                mdelay(200);
    }
    return iRet;
};
```

- **代码说明：**

  csi_ept_capture_init(): **-----** 捕捉模式下定时器基本参数设定

  csi_ept_set_sync(): **-----** ept 同步事件使能

- **函数参数说明：**

  csi_ept_capture_init(csp_ept_t *ptEptBase, csi_ept_captureconfig_t *pteptPwmCfg);

  ptEptBase: 解析同上

  pteptPwmCfg: 参数结构体指针

  pteptPwmCfg-> byWorkmod: 工作模式

  pteptPwmCfg-> byCountingMode: 计数方向

  pteptPwmCfg-> byOneshotMode: 计数器触发工作模式选择，0：连续计数模式、1：单次触发模式

  pteptPwmCfg-> byStartSrc: 软件使能同步触发控制

  pteptPwmCfg-> byPscld: PSCR 活动寄存器载入控制。活动寄存器在配置条件满足时，从影子寄存器载入更新值

  pteptPwmCfg-> byCaptureCapmd: 0：连续捕捉模式、1：一次性捕捉模式

  pteptPwmCfg-> byCaptureStopWrap: Capture 模式下，捕获事件计数器周期设置值

  pteptPwmCfg-> byCaptureLdaret: CMPA 捕捉载入后，计数器值计数状态控制位(1h：

CMPA 触发后，计数器值进行重置;0h：CMPA 触发后，计数器值不进行重置)

pteptPwmCfg-> byCaptureLdbret: 同 CMPA

pteptPwmCfg-> byCaptureLdcret: 同 CMPA

pteptPwmCfg-> byCaptureLddret: 同 CMPA

pteptPwmCfg-> wInt: 中断使能

- **PA01 口实际输入波形：**

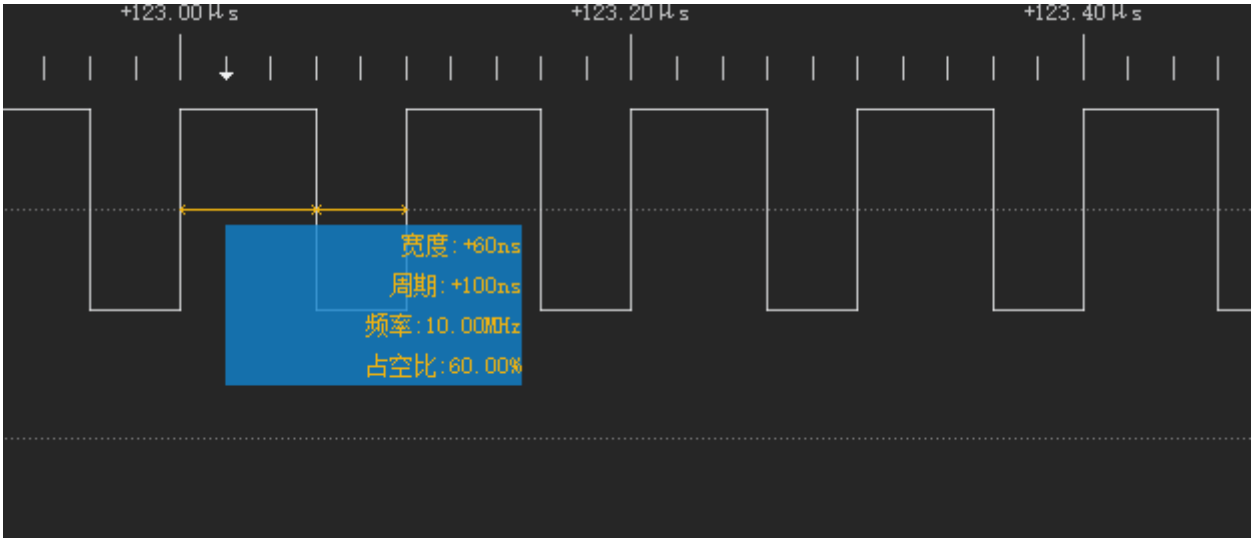

图 3.3.1 PA0.1 触发波形

- **计数器单周期时间：**

T = (val_)BUFF[1] – val_BUFF[0])*(1/48Mhz) = 104ns

val_BUFF	[4]
0	193
1	198
2	203
3	207

图 3.3.2 CDK 中触发

3.4 四路独立 PWM

系统时钟选择内部 48Mhz，输出周期为 100us,占空比不同，分别为 20%、40%、60%、80%。

(可在 user_demo.c 文件中的 ept_pwm_demo ()进行配置)

- ETP 输出引脚选择：

PA1.3->CHAX (20%占空比)/ PA1.4->CHBX(40%占空比) / PA1.5->CHCX(60%占空比) / PA1.6->CHD(80%占空比)

```
int ept_pwm_demo(void)
{
    int iRet = 0;
//-------------------------------------------------------------------------------------------------------------------
    csi_pin_set_mux(PA13, PA13_EPT_CHAX);                   //PIN17
    csi_pin_set_mux(PA14, PA14_EPT_CHBX);                   //PIN18
    csi_pin_set_mux(PA15, PA15_EPT_CHCX);                   //PIN19
    csi_pin_set_mux(PA16, PA16_EPT_CHD);                    //PIN20

//-------------------------------------------------------------------------------------------------------------------
  csi_ept_pwmconfig_t tPwmCfg;
    tPwmCfg.byWorkmod        = EPT_WAVE;                    //WAVE   波形模式
    tPwmCfg.byCountingMode   = EPT_UPDNCNT;                 //CNYMD  //计数方向
    tPwmCfg.byOneshotMode    = EPT_OP_CONT;                 //OPM    //单次或连续(工作方式)
    tPwmCfg.byStartSrc       = EPT_SYNC_START;              //软件使能同步触发使能控制（RSSR 中 START 控制位）//启动方式
    tPwmCfg.byPscld          = EPT_LDPSCR_ZRO;              //PSCR(分频)活动寄存器载入控制。活动寄存器在配置条件满足时，从影子寄存器载入更新值
    tPwmCfg.byDutyCycle      = 50;                          //pwm ouput duty cycle//PWM 初始值
    tPwmCfg.wFreq            = 10000;                       //pwm ouput frequency
    tPwmCfg.wInt             = 0;                           //interrupt
    csi_ept_wave_init(EPT0, &tPwmCfg);
//-------------------------------------------------------------------------------------------------------------------
    csi_ept_pwmchannel_config_t  channel;
    channel.byActionZro   =  LO;
    channel.byActionPrd   =  NA;
    channel.byActionC1u   =  HI;
    channel.byActionC1d   =  LO;
    channel.byActionC2u   =  NA;
    channel.byActionC2d   =  NA;
    channel.byActionT1u   =  LO;
    channel.byActionT1d   =  LO;
    channel.byActionT2u   =  NA;
    channel.byActionT2d   =  NA;
    channel.byChoiceC1sel =  EPT_CMPA;
    channel.byChoiceC2sel =  EPT_CMPA;
    csi_ept_channel_config(EPT0, &channel,  EPT_CHANNEL_1);//channel
    channel.byChoiceC1sel =  EPT_CMPB;
    channel.byChoiceC2sel =  EPT_CMPB;
    csi_ept_channel_config(EPT0, &channel,  EPT_CHANNEL_2);
    channel.byChoiceC1sel =  EPT_CMPC;
```

```
        channel.byChoiceC2sel   =    EPT_CMPC;
        csi_ept_channel_config(EPT0, &channel,   EPT_CHANNEL_3);
        channel.byChoiceC1sel   =    EPT_CMPD;
        channel.byChoiceC2sel   =    EPT_CMPD;
        csi_ept_channel_config(EPT0, &channel,   EPT_CHANNEL_4);
//-------------------------------------------------------------------------------------------------------------------
        csi_ept_change_ch_duty(EPT0,EPT_CAMPA, 20);
        csi_ept_change_ch_duty(EPT0,EPT_CAMPB, 40);
        csi_ept_change_ch_duty(EPT0,EPT_CAMPC, 60);
        csi_ept_change_ch_duty(EPT0,EPT_CAMPD, 80);
        csi_ept_start(EPT0);//start   timer
        while(1){
                }
        return iRet;
}
```

- **代码说明：**

  1. **csi_ept_wave_init():** **-----** 定时器波形模式参数基础设置（计数模式，周期，占空比等）
  2. **csi_ept_change_ch_duty():** **-----** 修改通道占空比

- **函数参数说明：**

  1. **csi_ept_wave_init(csp_ept_t *ptEptBase, csi_ept_pwmconfig_t *pteptPwmCfg);**

     ptEptBase: 同上

     pteptPwmCfg: 配置参数结构体

     pteptPwmCfg-> byWorkmod: 工作模式

     pteptPwmCfg-> byCountingMode: 计数方向

     pteptPwmCfg-> byOneshotMode: 计数器触发工作模式

     pteptPwmCfg-> byStartSrc: 软件使能同步触发使能控制

     pteptPwmCfg-> byPscld: PSCR 活动寄存器载入控制

     pteptPwmCfg-> byDutyCycle: PWM 初始占空比

     pteptPwmCfg-> wFreq: PWM 输出频率

pteptPwmCfg-> wInt: 中断控制

2. **csi_ept_change_ch_duty(csp_ept_t *ptEptBase, csi_ept_chtype_e eCh, uint32_t wActiveTime);**

   ptEptBase： 同上

   eCh：比较值寄存器选择

   wActiveTime： 占空比

- **输出波形：**

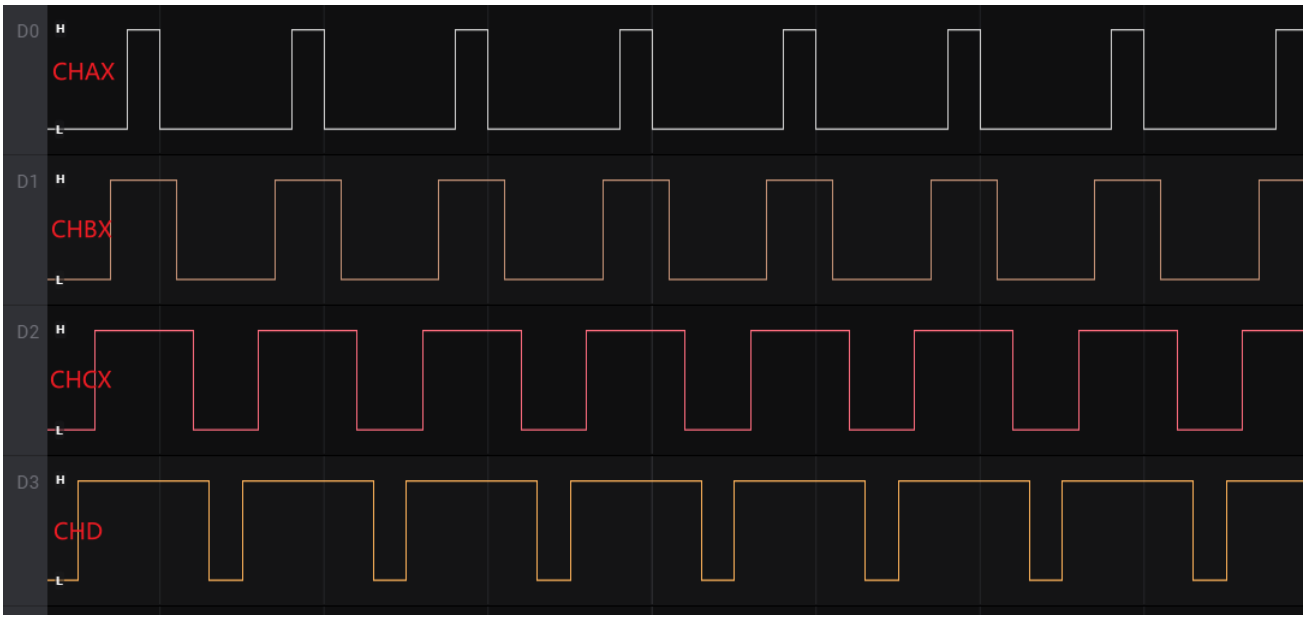

图 3.4.1　4 路独立输出波形

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。