文档版本	V1.0.0
发布日期	20221026

APT32F110x 基于 CSI 库 GPTB 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 GPTB 模块介绍 1

3.2 PWM 波形输出 3

3.3 PWM 波形输出带互补带死区 10

3.4 紧急模式控制 15

3.5 捕捉模式 21

4. 程序下载和运行 27

1 概述

本文介绍了APT32F110x中通用定时器GPTB应用。

2. 适用的硬件

该例程适用于 APT32F110x 系列学习板。

3. 应用方案代码说明

3.1 GPTB 模块介绍

基于 APT32F110x 完整的库文件系统，可以对 GPTB0 和 GPTB1 进行配置。以 GPTB0 为例，GPTB1 的应用，只需要把程序中的实例换为 GPTB1 即可。

- **主要特性**

GPTB 内部包含两个 16 计数模块,支持 2 种工作模式(捕捉模式和波形发生器模式)。

1 组独立的 PWM 互补输出+ 1 路独立的 PWM 输出

可编程的死区控制单元

支持可编程的相位控制

异常情况处理控制单元

GPTB 模块能和其它外设一起，被同步触发。

GPTB 的事件触发，能作为同步触发信号源。

GPTB 中，很多寄存器具有影子寄存器功能。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。每个影子寄存器对活动寄存器的更新条件均可以独立设置。全局载入模式可以重载这些配置，当全局载入使能时，所有影子寄存器对活动寄存器的更新都将受全局载入条件控制，并在全局载入条件满足时，将全部影子寄存器的当前值更新到对应的活动寄存器中。

增强型通用定时器(EPT)，增强型通用定时器 A (GPTA)和增强型通用定时器 B (GPTB)的

PRDR、CMPA 和 CMPB 等寄存器可以相互链接。

- **基本功能模块**

一个完整的 GPTB 模块由两个 TIMER 输入/输出通道组成。多个 GPTB 或多个 GPT 可以通过 SYNC 链连接，通过 SYNC 链，实现多个 GPTB 的同步工作。

每个 GPTB 根据功能划分，可以分为一下几个模块：

时钟控制模块，时基控制模块（计数器），计数器比较模块，死区控制模块，波形生成控制模块，捕捉模块，同步触发控制模块，紧急处理模块，寄存器链接控制模块。

- **管脚描述：**

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
GPT_CHAX	时钟控制使能	输出波形	输出波形
GPT_CHAY		输出波形	输出波形
GPT_CHB	时钟控制使能	输出波形	输出波形

图 3.1.1 管脚描述

GPT_CHAX,**GPT_CHAY** 和 **GPT_CHB**,是 GPTB 在 GPIO 口上映射的双向输入输出口。

在群脉冲模式下（GPTB_CR[BURST]使能），GPT_CHAX 和 GPT_CHB 可以作为门控时钟的时钟控制输入信号。

EBIx 为外部 GPIO 上映射的输入功能，作为紧急状态处理的触发信号。

- 模块框图：

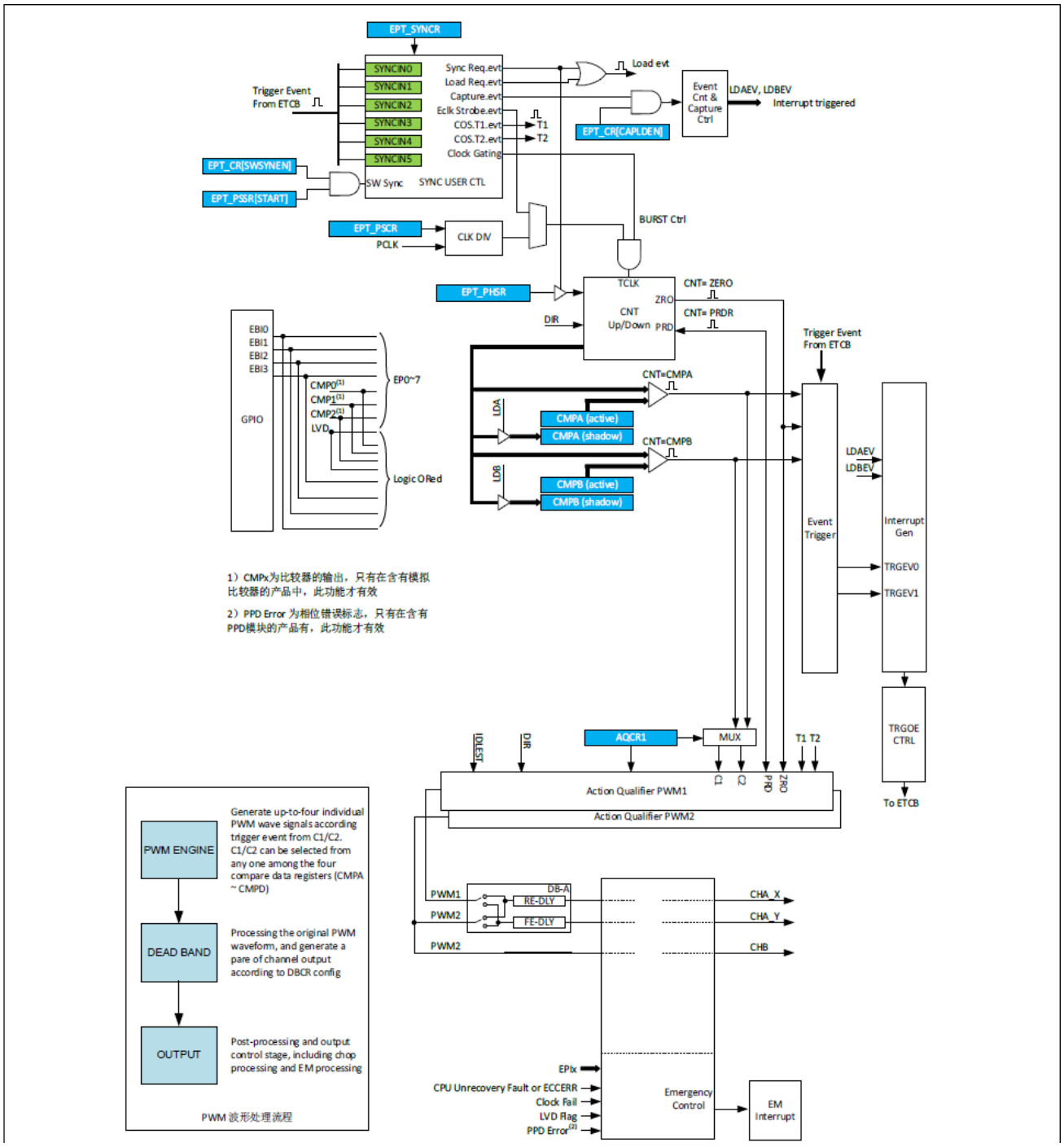

图 3.1.2 模块框图

3.2 PWM 波形输出

可在 system.c 文件中 void gptb0_pwm_config (void)函数进行初始化的配置。输出 10KHZ，占空比为 30%的 PWM 信号，高电平时间 30US。

```
void gptb0_config1(void)
{
        csi_pin_set_mux(PA013, PA013_GPTB0_CHAX);
    csi_pin_set_mux(PA014, PA014_GPTB0_CHAY);
    csi_pin_set_mux(PB04,   PB04_GPTB0_CHB   );

//--------------------------CMPLDR-----------------------------------------------
         //csi_gptb_channel_cmpload_config(GPTB0,GPTB_CMPLD_SHDW, GPTB_LDCMP_ZRO ,GPTB_CAMPA);
         //csi_gptb_channel_cmpload_config(GPTB0,GPTB_CMPLD_SHDW, GPTB_LDCMP_ZRO ,GPTB_CAMPB);
//-------------------------------------------------------------------------------
    csi_gptb_pwmconfig_t tPwmCfg;
        tPwmCfg.byWorkmod           = GPTB_WAVE;
        tPwmCfg.byCountingMode      = GPTB_UPDNCNT;
        tPwmCfg.byOneshotMode       = GPTB_OP_CONT;
        tPwmCfg.byStartSrc          = GPTB_SYNC_START;
        tPwmCfg.byPscld             = GPTB_LDPSCR_ZRO;
        tPwmCfg.byDutyCycle         = 30;
        tPwmCfg.wFreq                       = 10000;      //        必须小于PCLK
        tPwmCfg.wInt                        =  GPTB_INTSRC_NONE;                            csi_gptb_wave_init(GPTB0, &tPwmCfg);

//-----------------------------------------------------------------------------------------
//csi_gptb_channel_aqload_config(GPTB0,GPTB_LD_SHDW, GPTB_LDCMP_PRD ,GPTB_CHANNEL_1);
//csi_gptb_channel_aqload_config(GPTB0,GPTB_LD_SHDW, GPTB_LDCMP_PRD ,GPTB_CHANNEL_2);
//-----------------------------------------------------------------------------------------
        csi_gptb_pwmchannel_config_t    channel;
        channel.byActionZro     =     B_LO;
        channel.byActionPrd     =     B_NA;
        channel.byActionC1u     =     B_HI;
        channel.byActionC1d     =     B_LO;
        channel.byActionC2u     =     B_HI;
        channel.byActionC2d     =     B_LO;
        channel.byActionT1u     =     B_LO;
        channel.byActionT1d     =     B_LO;
        channel.byActionT2u     =     B_NA;
        channel.byActionT2d     =     B_NA;
        channel.byChoiceC1sel   =     GPTB_CMPA;
        channel.byChoiceC2sel   =     GPTB_CMPA;
        csi_gptb_channel_config(GPTB0, &channel,    GPTB_CHANNEL_1);
        channel.byChoiceC1sel   =     GPTB_CMPB;
        channel.byChoiceC2sel   =     GPTB_CMPB;
        csi_gptb_channel_config(GPTB0, &channel,    GPTB_CHANNEL_2);
//-------------------------------------------------------------------------------------------------------
```

```
/*      csi_gptb_Global_load_control_config_t   Gldcfg;
        Gldcfg.bGlden           =   ENABLE;//DISABLE
         Gldcfg.bOstmd           =   DISABLE;
         Gldcfg.bGldprd          =   0;
        Gldcfg.byGldcnt         =   0;
        Gldcfg.byGldmd=GPTB_LDGLD_SW;               csi_gptb_global_config(GPTB0,&Gldcfg);

        csi_gptb_gldcfg(GPTB0 ,bycmpa_B ,ENABLE);
        csi_gptb_gldcfg(GPTB0 ,bycmpb_B ,ENABLE);
        csi_gptb_global_rearm(GPTB0) ;
        csi_gptb_global_sw(GPTB0) ;
*/
//-------------------------------------------------------------------------------------
        csi_gptb_set_evtrg(GPTB0,GPTB_TRGOUT0,GPTB_TRG01_ZRO);
        csi_gptb_set_evtrg(GPTB0,GPTB_TRGOUT1,GPTB_TRG01_PRD);
        csi_gptb_int_enable(GPTB0, GPTB_INT_TRGEV0 , ENABLE);
        csi_gptb_int_enable(GPTB0, GPTB_INT_TRGEV1 , ENABLE);
        csi_gptb_start(GPTB0);

}
```

- **代码说明：**

1) csi_pin_set_mux(PA013, PA013_GPTB0_CHAX);

用于配置 GPTB0 映射到 GPIO 上的输入输出口。

2) **csi_gptb_channel_cmpload_config(GPTB0,GPTB_CMPLD_SHDW,GPTB_LDCMP_ZRO , GPTB_CAMPA);**

比较值寄存器的载入模式配置。

GPTB_CMPLD_SHDW---载入方式:影子寄存器，立即载入

GPTB_LDCMP_ZRO---载入条件选择

GPTB_CAMPA---CMPA,CMPB 选择

3) csi_gptb_channel_aqload_config(GPTB0,GPTB_LD_SHDW,GPTB_LDCMP_PRD , GPTB_CHANNEL_1);

波形输出控制寄存器的载入模式配置

GPTB_LD_SHDW---载入方式:影子寄存器，立即载入

GPTB_LDCMP_PRD---在 Shadow 模式下，载入条件的配置，3 个 BIT 位代表三种载入条件，可多选。如 001，010，011，111。

GPTB_CHANNEL_1---配置通道选择

注意：在改变 AQLDR 寄存器时 会清除相应的 AQCRx，所以必须放在配置 AQCR1/2 前面。

4) csi_gptb_wave_init(GPTB0, &tPwmCfg);

根据 **tPwmCfg** 结构体初始化赋值，对 GPTB0/1 进行配置。

tPwmCfg.byWorkmod --- 工作模式选择波形输出/捕获模式

tPwmCfg.byCountingMode--- 时基 CNT 计数模式选择

tPwmCfg.byOneshotMode--- 单次或连续(工作方式)

tPwmCfg.byStartSrc --- 启动方式，软件使能同步触发使能控制

tPwmCfg.byPscld--- GPTB 模块的时钟 TCLK 从 PCLK 分频后得到，PSC 活动寄存器载入控制

tPwmCfg.byDutyCycle---占空比参数，0~100

tPwmCfg.wFreq---输出 PWM 的频率。比如赋值 10000，则频率为 10KHZ，周期为 100 微秒

tPwmCfg.wInt---GPTB 模块内的中断方式使能，在 **csi_gptb_wave_init**（）内已打开 CPU 内核对应 GPTB 模块的中断

5) csi_gptb_channel_config(GPTB0, &channel, GPTB_CHANNEL_1);

根据结构体 **channel** 的赋值，对 GPTB0 的 CHAX 和 CHAY,CHB 两个波形输出通道进行功能配置。 由于 GPTB0 分别有两个独立的波形输出通道，所以此函数需要调用两次分别配置。其中 CHAY 如果无死区控制，输出波形和 CHB 相同。根据 **GPTB_CHANNEL_1** 选择通道。

channel.byActionZro ---当时基 CNT 计数器为零时，在对应的 PWM 上做出的波形输出动作定义

channel.byActionPrd ---当时基 CNT 计数器等于周期 PRDR 计数器值时，在对应的 PWM 上做出的波形输出动作定义

channel.byActionC1u---当 CNT 值等于 C1，且此时计数方向为递增时，在对应的 PWM 上做出的波形输出动作定义

channel.byActionC1d---当 CNT 值等于 C1，且此时计数方向为递减时，在对应的 PWM 上做出的波形输出动作定义

channel.byActionC2u---当 CNT 值等于 C2，且此时计数方向为递增时，在对应的 PWM 上做出的波形输出动作定义

channel.byActionC2d---当 CNT 值等于 C2，且此时计数方向为递减时，在对应的 PWM 上做出的波形输出动作定义

channel.byActionT1u ---当 T1 事件发生时，且此时计数方向为递增时，在对应的 PWM 上做出的波形输出动作定义

channel.byActionT1d---当 T1 事件发生时，且此时计数方向为递减时，在对应的 PWM 上做出的波形输出动作定义

channel.byActionT2u ---当 T2 事件发生时，且此时计数方向为递增时，在对应的 PWM 上做出的波形输出动作定义

channel.byActionT2d ---当 T2 事件发生时，且此时计数方向为递减时，在对应的 PWM 上做出的波形输出动作定义

channel.byChoiceC1sel---C1 比较值选择，CMPA 或 CMPB

channel.byChoiceC2sel---C2 比较值选择，CMPA 或 CMPB

6) csi_gptb_global_config(GPTB0,&Gldcfg);

全局载入控制寄存器配置，基于结构体 **Gldcfg** 初始值。

Gldcfg.bGlden--- 全局的 Shadow 到 Active 寄存器载入控制使能设置

Gldcfg.byGldmd---全局载入触发事件选择

Gldcfg.bOstmd---单次载入模式使能设置

Gldcfg.bGldprd---全局载入触发周期选择，可以选择 N 次触发条件满足后，才进行一次全局载入，0 代表立即载入

Gldcfg.byGldcnt---全局载入事件计数器，记录事件触发了多少次，3 个 BIT 位最大记录 7 次。

7) csi_gptb_gldcfg(GPTB0 ,bycmpa_B ,ENABLE);

当全局载入被使能后，配置 6 个独立的寄存器是否受控于全局载入影响。

GPTB0/1---选择对应模块

bycmpa_B---选择一个独立的寄存器

ENABLE/DISABLE---ENABLE 受控于全局载入控制，DISABLE 立即载入

8) csi_gptb_global_rearm(GPTB0) ;

重置 ONE SHOT 模式。ONE SHOT 模式下，一次触发后，需要重置模式才允许再次触发

9) csi_gptb_global_sw(GPTB0) ;

软件产生一次全局触发事件

10) csi_gptb_set_evtrg(GPTB0, GPTB_TRGOUT0, GPTB_TRG01_ZRO);

GPTB 事件触发输出配置

GPTB0/1---选择对应模块

GPTB_TRGOUT0---事件触发端口选择 0~1

GPTB_TRG01_ZRO---触发端口对应的事件选择

11) csi_gptb_int_enable(GPTB0, GPTB_INT_TRGEV0 , ENABLE);

使能 GPTB 内部的中断和 CPU 内核中断

GPTB_INT_TRGEV0---GPTB 模块内部中断源选择

12) csi_gptb_set_evcntinit(GPTB0, GPTB_TRGOUT0, 5, 0);

GPTB 事件触发计数器设置，TRGSEL0 每 5 次产生一次事件输出。

GPTB_TRGOUT0---选择触发端口

“5”---计数次数

“0”---计数起始值

13) csi_gptb_reglk_config(GPTB0,&FEGLKcfg);

GPTB0/1 的选定的寄存器作为目标寄存器链接到对应功能模块的源定时器，基于结构体 **FEGLKcfg** 的赋值。实现不同功能模块的多个定时器寄存器级联，修改其中一个，可同时改变。

FEGLKcfg---十一个目标寄存器链接到不同功能模块的同名寄存器，

1h:EPT0
2H:GPTA0
3H:GPTA1
4H:GPTB0
5H:GPTB1

14) csi_gptb_start(GPTB0);

启动定时器

- **波形输出：**

图 3.2.1 输出波形

高电平 30US，周期 100US，频率 10KHZ

3.3 PWM 波形输出带互补带死区

可在 system.c 文件中 void gptb0_pwm_dz_config (void)函数进行初始化的配置。输出 10KHZ，占空比为 50%的 PWM 信号，高电平时间 50US。在 **3.2PWM 波形输出**章节的基础上，增加了**死区控制模块：**

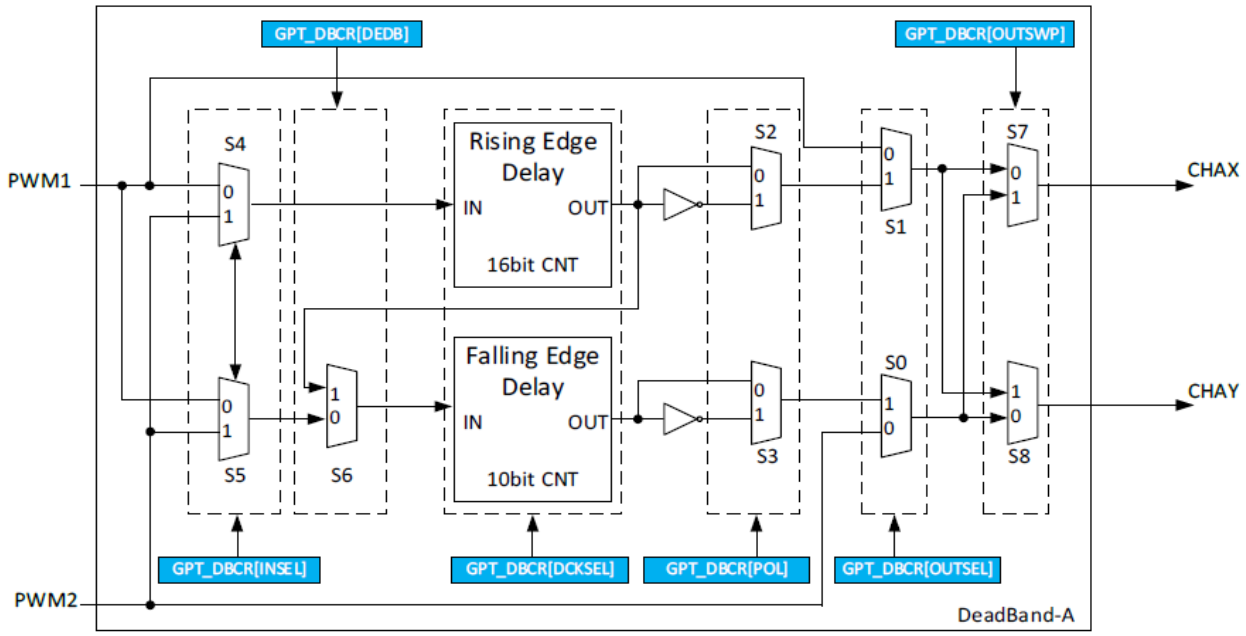

图 3.3.1 死区控制模块

```
void gptb0_pwm_dz_config(void)
```

```
{
    csi_pin_set_mux(PA013, PA013_GPTB0_CHAX);
      csi_pin_set_mux(PA014, PA014_GPTB0_CHAY);
      csi_pin_set_mux(PB04,   PB04_GPTB0_CHB   );
//-------------------------------------------------------------------------
    csi_gptb_pwmconfig_t tPwmCfg;
    tPwmCfg.byWorkmod          = GPTB_WAVE;
    tPwmCfg.byCountingMode   = GPTB_UPDNCNT;
    tPwmCfg.byOneshotMode     = GPTB_OP_CONT;
    tPwmCfg.byStartSrc         = GPTB_SYNC_START;
    tPwmCfg.byPscld             = GPTB_LDPSCR_ZRO;
    tPwmCfg.byDutyCycle        = 50;
    tPwmCfg.wFreq                      = 10000;
//  tPwmCfg.wInt                   = GPTB_INT_TRGEV0;
    csi_gptb_wave_init(GPTB0, &tPwmCfg);
//csi_gptb_set_sync(GPTB0,GPTB_TRG_SYNCEN2,GPTB_TRG_CONTINU,
GPTB_AUTO_REARM_ZRO);
//  csi_gptb_set_evtrg(GPTB0, GPTB_TRGOUT0, GPTB_TRGSRC_PE1);
//  csi_gptb_int_enable(GPTB0, GPTB_INT_TRGEV0,true);
//-------------------------------------------------------------------------
    csi_gptb_pwmchannel_config_t   tGptbchannelCfg;
    tGptbchannelCfg.byActionZro     =    B_LO;
    tGptbchannelCfg.byActionPrd     =    B_NA;
    tGptbchannelCfg.byActionC1u     =    B_HI;
    tGptbchannelCfg.byActionC1d     =    B_LO;
    tGptbchannelCfg.byActionC2u     =    B_NA;
    tGptbchannelCfg.byActionC2d     =    B_NA;
    tGptbchannelCfg.byActionT1u     =    B_LO;
    tGptbchannelCfg.byActionT1d     =    B_LO;
```

```
    tGptbchannelCfg.byActionT2u      =    B_NA;
    tGptbchannelCfg.byActionT2d      =    B_NA;
    tGptbchannelCfg.byChoiceC1sel    =    GPTB_CMPA;
    tGptbchannelCfg.byChoiceC2sel    =    GPTB_CMPA;
    csi_gptb_channel_config(GPTB0, &tGptbchannelCfg,   GPTB_CHANNEL_1);
    tGptbchannelCfg.byChoiceC1sel    =    GPTB_CMPB;
    tGptbchannelCfg.byChoiceC2sel    =    GPTB_CMPB;
    csi_gptb_channel_config(GPTB0, &tGptbchannelCfg,   GPTB_CHANNEL_2);


//csp_gptb_set_aqtscr(GPTB0,GPTB_T1,EP1);
     //--------death-zone-----------------------------------------------------------
    csi_gptb_deadzone_config_t   tGptbDeadZoneTime;
    tGptbDeadZoneTime.byDcksel                    = GPTB_DB_DPSC;
    tGptbDeadZoneTime.hwDpsc                      =   0;
    tGptbDeadZoneTime.hwRisingEdgereGister    = 500;
    tGptbDeadZoneTime.hwFallingEdgereGister   = 200;
    tGptbDeadZoneTime.byChaDedb                   = B_DB_AR_BF;
    csi_gptb_dz_config(GPTB0,&tGptbDeadZoneTime);
    tGptbDeadZoneTime.byChxOuselS1S0        = B_DBOUT_AR_BF;
    tGptbDeadZoneTime.byChxPolarityS3S2     = B_DB_POL_B;
    tGptbDeadZoneTime.byChxInselS5S4        = B_DBCHAIN_AR_AF;
    tGptbDeadZoneTime.byChxOutSwapS8S7      = B_DBCHAOUT_OUTA_A_OUTB_B;
      csi_gptb_channelmode_config(GPTB0,&tGptbDeadZoneTime,GPTB_CHANNEL_1);
    csi_gptb_start(GPTB0);//start   timer
}
```

- **代码说明：**

  波形生成参考 3.2 章节的说明，此处只介绍死区控制部分

1) csi_gptb_dz_config(GPTB0,&tGptbDeadZoneTime);

   根据结构体 tGptbDeadZoneTime 的初始化对实例 GPTB0/1 进行死区**时序**配置

tGptbDeadZoneTime.byDcksel---死区控制模块时钟源选择

tGptbDeadZoneTime.hwDpsc---延时模块时钟分频比 FDBCLK = FHCLK / (DPSC+1)

tGptbDeadZoneTime.hwRisingEdgereGister----上升沿延时（ns）“500”

tGptbDeadZoneTime.hwFallingEdgereGister ---下降沿延时（ns） “200”

tGptbDeadZoneTime.byChaDedb---在通道 1 选择是否使用死区双沿

2) csi_gptb_channelmode_config(GPTB0,&tGptbDeadZoneTime,GPTB_CHANNEL_1);

根据结构体 tGptbDeadZoneTime 的初始化对实例 GPTB0/1 进行死区**通道**配置

tGptbDeadZoneTime.byChxOuselS1S0---使能通道 A 的上升沿延时，使能通道 B 的下降沿延时

tGptbDeadZoneTime.byChxPolarityS3S2---通道 A 和通道 B 延时输出电平是否反向

tGptbDeadZoneTime.byChxInselS5S4---PWMA 作为上升和下降沿延时处理的输入信号

tGptbDeadZoneTime.byChxOutSwapS8S7---OUTA=通道 A 输出，OUTB=通道 B 输出

注：S0~S7 对应上面的**死区控制模块图**

- **波形输出：**

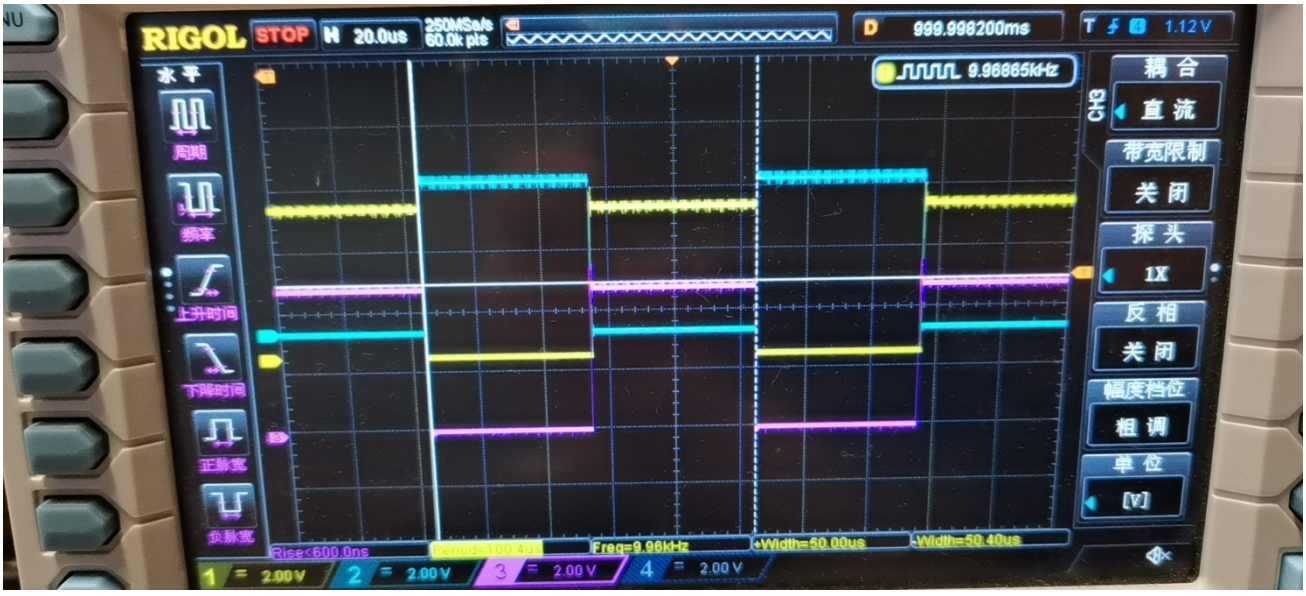

图 3.3.1 波形图一

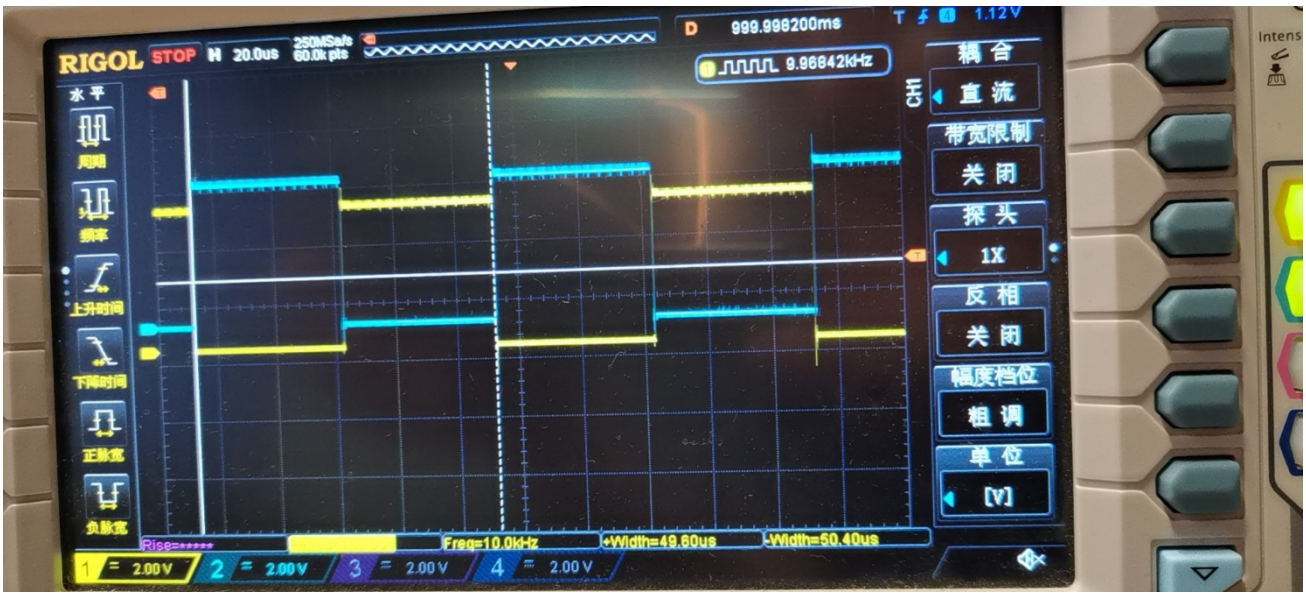

图 3.3.2 波形图二

图 3.3.3 下降沿死区时间-200ns

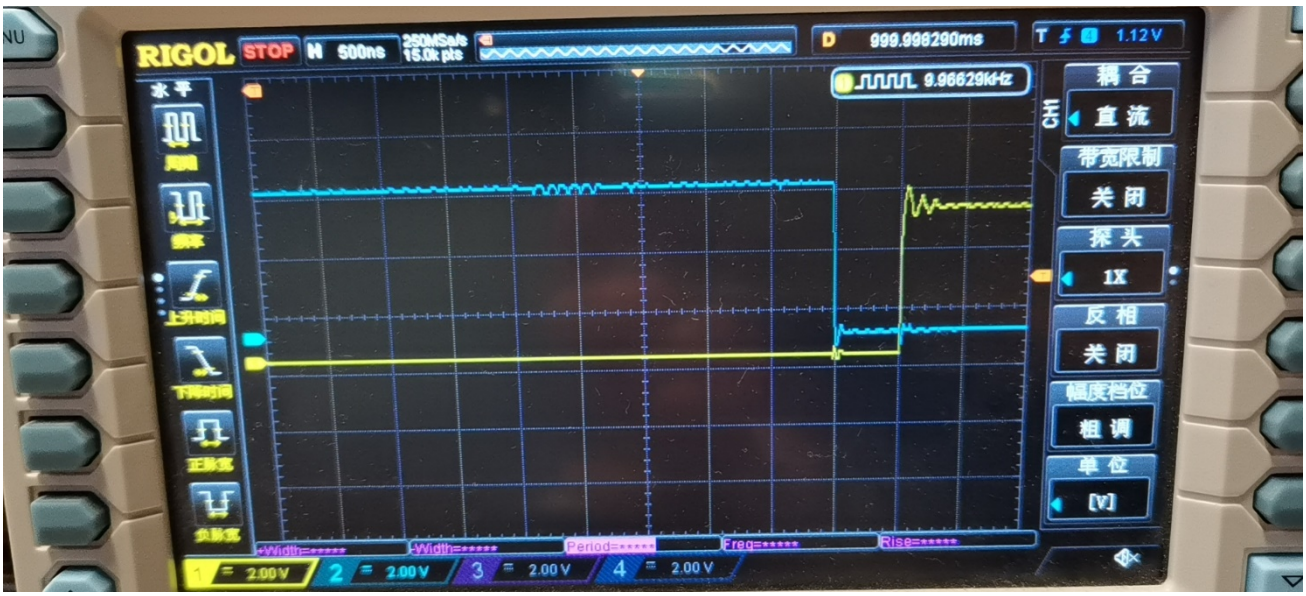

图 3.3.4 上升沿死区时间-500ns

3.4 紧急模式控制

在很多应用场合，PWM 输出需要对紧急异常状态做出相应的输出控制，实现过载保护，或故障处理。紧急事件处理模块可以对应外部触发，产生相应输出并产生相应中断。模块内支持 8 路紧急触发信号输入（EPx）通道，和 3 路系统故障触发通道(SYSFAIL)。

- **主要特性：**

紧急模式下，PWM 的输出可以被定义为：高电平输出，低电平输出，高阻，或者不进行动作。

对紧急模式的处置策略有两种：硬锁止（Hard-Lock），主要用于短路或者过流保护；软锁止（Soft-Lock），主要用于限流保护动作。

支持 8 路触发输入，每个触发输入可以独立选择触发信号源，实现 PWM 输出的保护联动。

独立的系统故障触发源。

独立的紧急模式触发中断源。

支持软件强制触发紧急状态。

- **模块框图：**

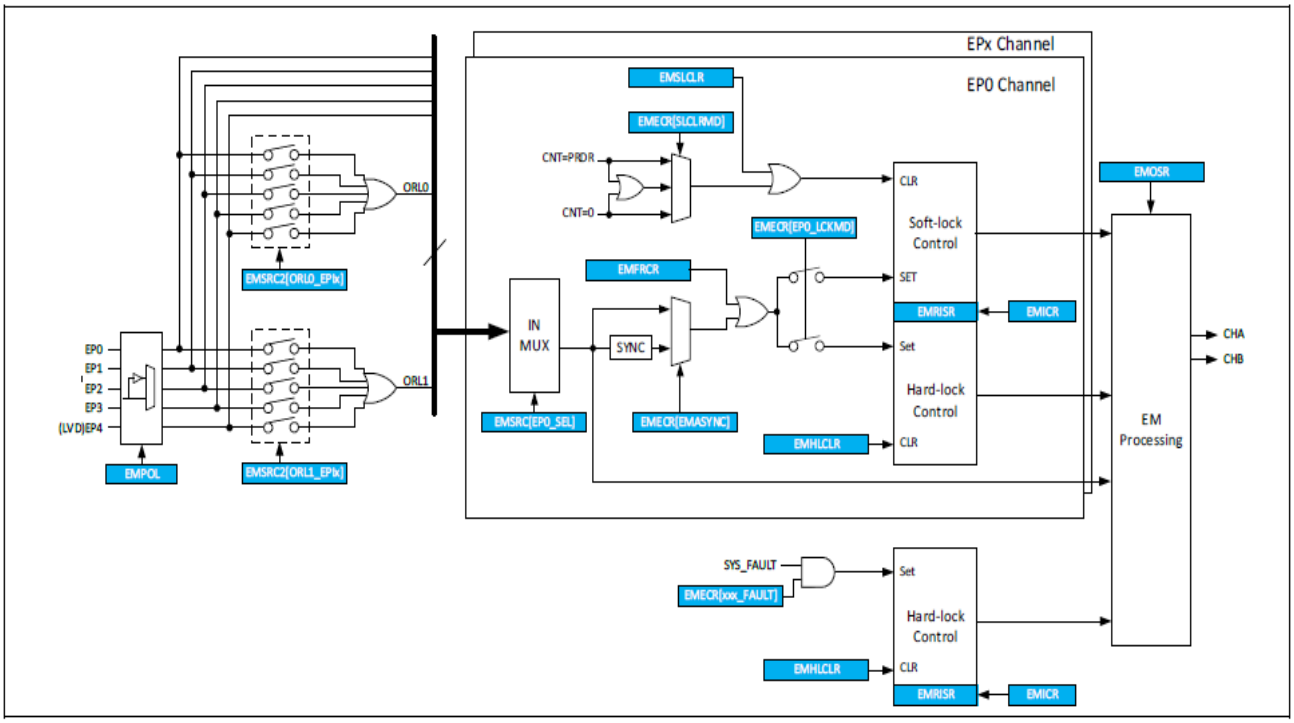

图 3.4.1 紧急控制模块

可在 system.c 文件中 gptb0_pwm_dz_em_config 函数进行初始化的配置。以章节 3.3 的基础上配置紧急模式的应用。

```
//-------------------------------------------------------------------

void gptb0_pwm_dz_em_config(void)

{

        csi_pin_set_mux(PA013, PA013_GPTB0_CHAX);

        csi_pin_set_mux(PA014, PA014_GPTB0_CHAY);

  //      csi_pin_set_mux(PB04,   PB04_GPTB0_CHB   );

        csi_pin_set_mux(PA09,PA09_EBI0);

  //      csi_pin_set_mux(PB04,PB04_EBI1);

  //-------------------------------------------------------------------

        csi_gptb_pwmconfig_t tPwmCfg;

        tPwmCfg.byWorkmod         = GPTB_WAVE;

        tPwmCfg.byCountingMode    = GPTB_UPDNCNT;

        tPwmCfg.byOneshotMode     = GPTB_OP_CONT;
```

```
    tPwmCfg.byStartSrc          = GPTB_SYNC_START;
    tPwmCfg.byPscld             = GPTB_LDPSCR_ZRO;
    tPwmCfg.byDutyCycle         = 50;
    tPwmCfg.wFreq                       = 10000;
    tPwmCfg.wInt          = GPTB_INTSRC_PEND;
    csi_gptb_wave_init(GPTB0, &tPwmCfg);
//-------------------------------------------------------------------------
    csi_gptb_pwmchannel_config_t   tGptbchannelCfg;
    tGptbchannelCfg.byActionZro     =    B_LO;
    tGptbchannelCfg.byActionPrd     =    B_NA;
    tGptbchannelCfg.byActionC1u     =    B_HI;
    tGptbchannelCfg.byActionC1d     =    B_LO;
    tGptbchannelCfg.byActionC2u     =    B_NA;
    tGptbchannelCfg.byActionC2d     =    B_NA;
    tGptbchannelCfg.byActionT1u     =    B_LO;
    tGptbchannelCfg.byActionT1d     =    B_LO;
    tGptbchannelCfg.byActionT2u     =    B_NA;
    tGptbchannelCfg.byActionT2d     =    B_NA;
    tGptbchannelCfg.byChoiceC1sel   =    GPTB_CMPA;
    tGptbchannelCfg.byChoiceC2sel   =    GPTB_CMPA;
    csi_gptb_channel_config(GPTB0, &tGptbchannelCfg,   GPTB_CHANNEL_1);
    tGptbchannelCfg.byChoiceC1sel   =    GPTB_CMPB;
    tGptbchannelCfg.byChoiceC2sel   =    GPTB_CMPB;
    csi_gptb_channel_config(GPTB0, &tGptbchannelCfg,   GPTB_CHANNEL_2);
//          csp_gptb_set_aqtscr(GPTB0,GPTB_T1,EP2);
//-------------------------------------------------------------------------
    csi_gptb_deadzone_config_t tGptbDeadZoneTime;
    tGptbDeadZoneTime.byDcksel                  = GPTB_DB_DPSC;
    tGptbDeadZoneTime.hwDpsc                    =      0;
```

```
    tGptbDeadZoneTime.hwRisingEdgereGister          = 500;
    tGptbDeadZoneTime.hwFallingEdgereGister   = 200;
    tGptbDeadZoneTime.byChaDedb                     = B_DB_AR_BF;
    csi_gptb_dz_config(GPTB0,&tGptbDeadZoneTime);
    tGptbDeadZoneTime.byChxOuselS1S0        = B_DBOUT_AR_BF;
    tGptbDeadZoneTime.byChxPolarityS3S2     = B_DB_POL_B;
    tGptbDeadZoneTime.byChxInselS5S4          = B_DBCHAIN_AR_AF;
    tGptbDeadZoneTime.byChxOutSwapS8S7     = B_DBCHAOUT_OUTA_A_OUTB_B;
    csi_gptb_channelmode_config(GPTB0,&tGptbDeadZoneTime,GPTB_CHANNEL_1);
//-----------------------------------------------------------------
    csi_gptb_emergency_config_t     tGptbEmergencyCfg;
    tGptbEmergencyCfg.byEpxInt            = B_EBI0 ;
    tGptbEmergencyCfg.byPolEbix     = B_EBI_POL_L;
    tGptbEmergencyCfg.byEpx      = B_EP3;
    tGptbEmergencyCfg.byEpxLckmd    = B_EP_HLCK;
    tGptbEmergencyCfg.byOsrshdw     = B_IMMEDIATE;
    tGptbEmergencyCfg.byFltpace0    = B_EPFLT0_2P;
    tGptbEmergencyCfg.byFltpace1    = B_EPFLT1_2P;
    if(tGptbEmergencyCfg.byEpxInt ==B_ORL0)
      {tGptbEmergencyCfg.byOrl0 = B_ORLx_EP0 |B_ORLx_EP1|B_ORLx_EP2;}
    if(tGptbEmergencyCfg.byEpxInt ==B_ORL1)
      {tGptbEmergencyCfg.byOrl1 = B_ORLx_EP4 |B_ORLx_EP5|B_ORLx_EP6;}
    csi_gptb_emergency_cfg(GPTB0,&tGptbEmergencyCfg);
    csi_gptb_emergency_pinxout(GPTB0,GPTB_EMCOAX,B_EM_OUT_L);
    csi_gptb_emergency_pinxout(GPTB0,GPTB_EMCOAY,B_EM_OUT_L);
    csi_gptb_emergency_int_enable(GPTB0,B_EM_INT_EP3);
//-------------------------------------------------------------------------
   // csi_gptb_set_sync(GPTB0,GPTB_TRG_SYNCEN3, GPTB_TRG_CONTINU,GPTB_AUTO_REARM_ZRO);
//      csi_gptb_set_evtrg (GPTB0, GPTB_TRGOUT0, GPTB_TRGSRC_EP0);
```

```
    //      csi_gptb_int_enable(GPTB0, GPTB_INT_TRGEV0,true);
            //-------------------------------------------------------------------------    //      csi_gptb_feglk_config_t   FEGLKcfg1;


    //      FEGLKcfg1.byPrdr    = 0;
    //      FEGLKcfg1.byRssr    = 1;
    //      FEGLKcfg1.byCmpa  = 1;
    //      FEGLKcfg1.byCmpb  = 1;
    //      FEGLKcfg1.byGld2    = 0;
    //      FEGLKcfg1.byEmslclr = 1;
    //      FEGLKcfg1.byEmhlclr = 1;
    //      FEGLKcfg1.byEmicr   = 1;
    //      FEGLKcfg1.byEmfrcr = 1;
    //      FEGLKcfg1.byAqosf   = 1;
    //      FEGLKcfg1.byAqcsf   = 1;
    //          csi_gptb_reglk_config(GPTB0,&FEGLKcfg1);
    //-------------------------------------------------------------------------
            csi_gptb_start(GPTB0);//start    timer
}
```

- **代码说明：**

在 3.3 章节的 PWM 波形输出带互补死区的基础上，增加了紧急模式的配置。波形输出部分见上一章节介绍，此处只介绍紧急模式配置。

1) **csi_pin_set_mux(PA09,PA09_EBI0);**

设置 PA09 为外部的紧急事件输入口

2) **csi_gptb_emergency_cfg(GPTB0,&tGptbEmergencyCfg);**

根据结构体 **tGptbEmergencyCfg** 对紧急事件模块配置

tGptbEmergencyCfg.byEpxInt---EPx 选择外部 IO 端口（EBI0~EBI4）

tGptbEmergencyCfg.byPolEbix---EBIx 的输入有效极性选择控制

tGptbEmergencyCfg.byEpx---使能 EPx

tGptbEmergencyCfg.byEpxLckmd---使能 软/硬 锁

tGptbEmergencyCfg.byOsrshdw---锁止端口状态载入方式

tGptbEmergencyCfg.byFltpace0---EP0、EP1、EP2 和 EP3 的数字去抖滤波检查周期数

tGptbEmergencyCfg.byFltpace1---EP4、EP5、EP6 和 EP7 的数字去抖滤波检查周期数

if(tGptbEmergencyCfg.byEpxInt ==B_ORL0)---逻辑或 0 通道如果使能

{tGptbEmergencyCfg.byOrl0 = B_ORLx_EP0 |B_ORLx_EP1|B_ORLx_EP2;}---配置

if(tGptbEmergencyCfg.byEpxInt ==B_ORL1)---逻辑或 1 通道如果使能

{tGptbEmergencyCfg.byOrl1 = B_ORLx_EP4 |B_ORLx_EP5|B_ORLx_EP6;}---配置

3) **csi_gptb_emergency_pinxout(GPTB0,GPTB_EMCOAX,B_EM_OUT_L);**

紧急状态下输出电平设置

GPTB_EMCOAX---输出通道选择 CHAX,CHAY,CHB

B_EM_OUT_L---输出状态选择 高阻，L,H,无变化

4) **csi_gptb_emergency_int_enable(GPTB0,B_EM_INT_EP3);**

使能 CPU 内部中断和紧急模块内部中断

B_EM_INT_EP3---模块内部中断选择（EP0~EP3，EOM_FAULT，MEM_FAULT，CPU_FAULT）

5) **csi_gptb_start(GPTB0);**

启动定时器

- **验证：**

在 3.3 章节输出的波形基础上，当 PA09 被拉低，PA013 和 PA014 输出的互补带死区

的 PWM 信号消失，两个 IO 口被拉低到了设置的低电平,同时触发中断.

在 main()函数中去清除硬件锁止中断标志位，通过示波器可观察到，PA09 拉高后，PA013 和 PA014 的输出波形将会恢复。

csp_gptb_clr_emHdlck(GPTB0, B_EP3);

清理对应模块的对应硬件锁止通道标注位。

3.5 捕捉模式

- **概述：**

  捕捉模式一般用于如下几个常见的应用：

  旋转机构的速度测量（比如霍尔传感器）

  位置传感器的脉冲间隔时间测量

  脉冲群的周期和占空比测量

- **模块框图：**

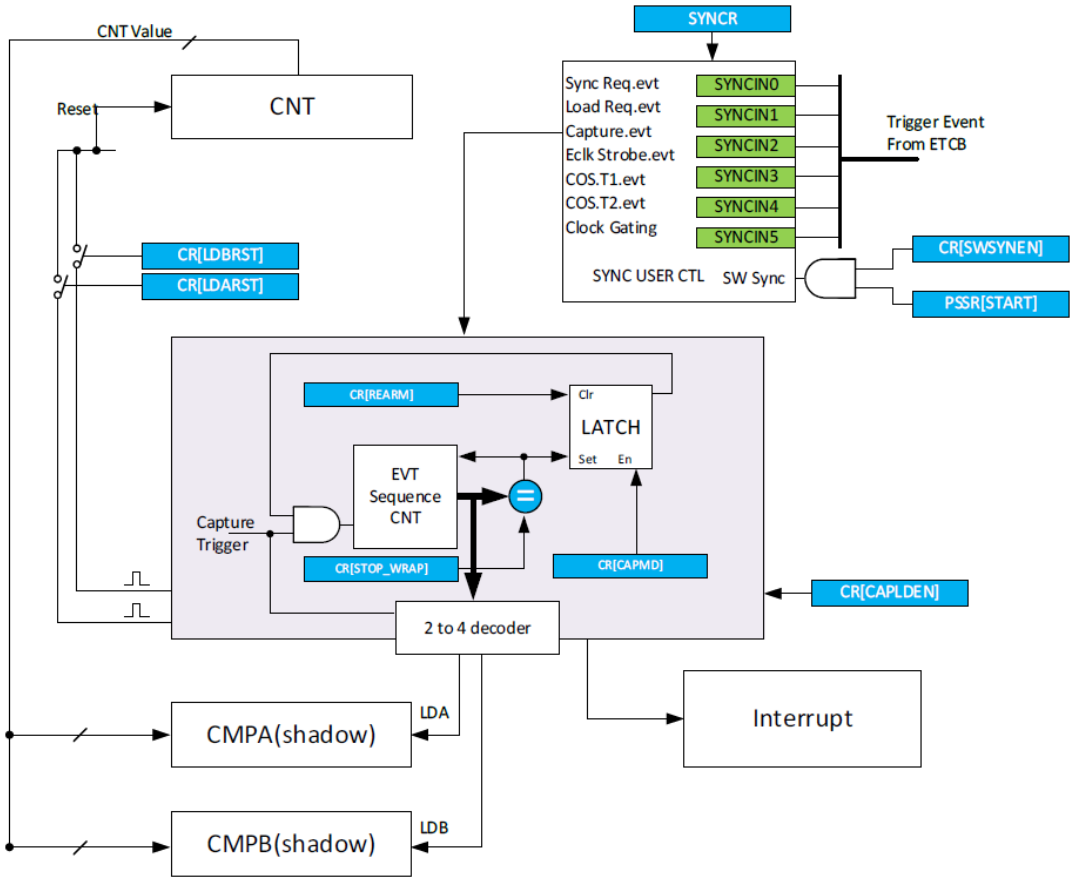

图 3.5.1 捕捉模式框图

可在 system.c 文件中 gptb0_capture_config();函数进行初始化的配置。

```
//-------------------------------------------------------------------------
void gptb0_capture_config(void)
{
        volatile uint8_t ch;
            csi_pin_set_mux(PA01,PA01_INPUT);
            csi_pin_pull_mode(PA01, GPIO_PULLUP);
            csi_pin_irq_mode(PA01,EXI_GRP1, GPIO_IRQ_BOTH_EDGE);
            csi_exi_set_evtrg(EXI_TRGOUT1, TRGSRC_EXI1, 0);
    //-------------------------------------------------------------------------
            csi_etb_config_t tEtbConfig;
            tEtbConfig.byChType   = ETB_ONE_TRG_ONE;
            tEtbConfig.bySrcIp    = ETB_EXI_TRGOUT1 ;
            tEtbConfig.bySrcIp1   = 0xff;
            tEtbConfig.bySrcIp2   = 0xff;
            tEtbConfig.byDstIp    =     ETB_GPTB0_SYNCIN2;
            tEtbConfig.byDstIp1   = 0xff;
            tEtbConfig.byDstIp2   = 0xff;
            tEtbConfig.byTrgMode = ETB_HARDWARE_TRG;
            csi_etb_init();
            ch = csi_etb_ch_alloc(tEtbConfig.byChType);
            csi_etb_ch_config(ch, &tEtbConfig);
    //-------------------------------------------------------------------------
            csi_gptb_captureconfig_t tPwmCfg;
            tPwmCfg.byWorkmod         = GPTB_CAPTURE;
            tPwmCfg.byCountingMode    = GPTB_UPCNT;
            tPwmCfg.byOneshotMode     = GPTB_OP_CONT;
            tPwmCfg.byStartSrc        = GPTB_SYNC_START;
```

```
        tPwmCfg.byPscld             = GPTB_LDPSCR_ZRO;
        tPwmCfg.byCaptureCapmd   = GPTB_CAPMD_CONT;
        tPwmCfg.byCaptureStopWrap=1;
        tPwmCfg.byCaptureLdaret   =0;
        tPwmCfg.byCaptureLdbret   =1;
        tPwmCfg.wInt           =GPTB_INT_CAPLD0;
        csi_gptb_capture_init(GPTB0, &tPwmCfg);
    //-------------------------------------------------------------------
csi_gptb_set_sync(GPTB0,GPTB_TRG_SYNCEN2, GPTB_TRG_CONTINU,GPTB_AUTO_REARM_ZRO);
    //-------------------------------------------------------------------
    //      csi_gptb_filter_config_t   tpFiltercfg;
    //      tpFiltercfg.byFiltSrc            =GPTB_FILT_SYNCIN2;
    //      tpFiltercfg.byWinInv             =1;
    //      tpFiltercfg.byWinAlign           =GPTB_ALIGN_ZRO;
    //      tpFiltercfg.byWinCross                 =1;
    //      tpFiltercfg.byWinOffset          =gGptb0Prd/2;
    //      tpFiltercfg.byWinWidth                 =gGptb0Prd/2;
    //        csi_gptb_set_sync_filter(GPTB0, &tpFiltercfg);
    //-------------------------------------------------------------------
        csi_gptb_start(GPTB0);
}
```

- 代码说明

GPIO 部分

1) csi_pin_set_mux(PA01,PA01_INPUT);

配置 PA01 口为信号输入口

2) csi_pin_pull_mode(PA01, GPIO_PULLUP);

设置内部上拉

3) **csi_pin_irq_mode(PA01,EXI_GRP1, GPIO_IRQ_FALLING_EDGE);**

配置对应 GPIO 的外部中断组，设置下降沿触发

4) **csi_exi_set_evtrg(EXI_TRGOUT1, TRGSRC_EXI1, 1);**

设置触发通道，触发源通道和触发条件：计数次数设置

EXI_TRGOUT0~5---事件触发通道，其中 0~3 可计数，4 和 5 不可以。

TRGSRC_EXI0~19---EXI 的触发源选择

1~15,触发条件:次数设置

ETCB 部分

5) **csi_etb_init();**

ETBC 模块的时钟和功能打开

6) **csi_etb_ch_config(ch, &tEtbConfig);**

根据结构体变量 **tEtbConfig** 配置 ETBC 通道

tEtbConfig.byChType--- ETCB 通道选择：00~11 四类通道：多对单，单对多，单对单，DMA

tEtbConfig.bySrcIp---作为触发源

tEtbConfig.bySrcIp1---多触发单时的触发源

tEtbConfig.bySrcIp2---多触发单时的触发源

tEtbConfig.byDstIp---触发目标事件选择

tEtbConfig.byDstIp1---单对多时的触发目标事件选择

tEtbConfig.byDstIp2---单对多时的触发目标事件选择

tEtbConfig.byTrgMode ---触发模式：硬件和软件方式

GPTB0 部分

7) **csi_gptb_capture_init(GPTB0, &tPwmCfg);**

根据结构体变量 **tPwmCfg** 初始化值配置捕捉模块

tPwmCfg.byWorkmod---GPTB0 模块功能选择 ：波形输出，捕捉模式

tPwmCfg.byCountingMode---时基 CNT 工作模式：递增，递减，递增递减

tPwmCfg.byOneshotMode---单次或连续捕捉模式

tPwmCfg.byStartSrc ---启动方式：软件，硬件从 ETCB 触发

tPwmCfg.byPscld---分频比寄存器从 SHADOW 载入到 ACTIVE 的条件选择：

CNT=0；CNT=PRD；CNT=0 或 CNT=PRD；不载入

tPwmCfg.byCaptureCapmd---捕捉模式：连续；单次

tPwmCfg.byCaptureStopWrap---捕捉事件计数器周期设置值，最大为 2

tPwmCfg.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)

tPwmCfg.byCaptureLdbret ---CMPB 捕捉载入后，计数器值计数状态控制位(1h：CMPB 触发后，计数器值进行重置;0h：CMPB 触发后，计数器值不进行重置)

tPwmCfg.wInt---GPTB 模块内中断使能

tPwmCfg.byBurst ---使能群脉冲模式

tPwmCfg.byCgsrc---选择 CG 的输入口 CHA or CHB //CHB 作为 CG 的输入源

tPwmCfg.byCgflt---门控输入数字滤波控制,N 个 TICK 有效。

8) **csi_gptb_set_sync(GPTB0,GPTB_TRG_SYNCEN2,GPTB_TRG_CONTINU, GPTB_AUTO_REARM_ZRO);**

选择外部触发通道，并配置工作模式和重载模式

GPTB_TRG_SYNCEN2---选 SYNCIN2 为外部同步触发通道

GPTB_TRG_CONTINU---工作模式：连续 OR 单次

9) csi_gpta_start(GPTB0);

启动 GPTB 计数器

GPTB0 中断部分

__attribute__((weak)) void gptb0_irqhandler_pro(csp_gptb_t *ptGptbBase)

里面如下部分

```
if(((csp_gptb_get_misr(ptGptbBase) & GPTB_INT_CAPLD0))==GPTB_INT_CAPLD0)
{
    val_buff_t[0]=csp_gptb_get_cmpa(ptGptbBase);
    val_buff_t[1]=csp_gptb_get_cmpb(ptGptbBase);
    csp_gptb_clr_int(ptGptbBase, GPTB_INT_CAPLD0);
}
```

val_buff_t[0]为捕捉到的高电平计数值

val_buff_t[1]为捕捉到的周期计数值

- **验证：**

  利用 CNTA 模块的波形生成功能，从 PA1.10 输出频率 38KHZ，占空比 30%的信号，用作捕获模式的输入信号源。

图 3.5.2 捕捉数据

计算公式：val_buff_t[]/48000000　　48000000 为 GPTB 部分的时钟，结果单位为秒，乘以 10 的 6 次方换算为微妙

val_buff_t[1]/48000000=26.27US　　val_buff_t[0]/48000000=18.375US

下图为输入的信号

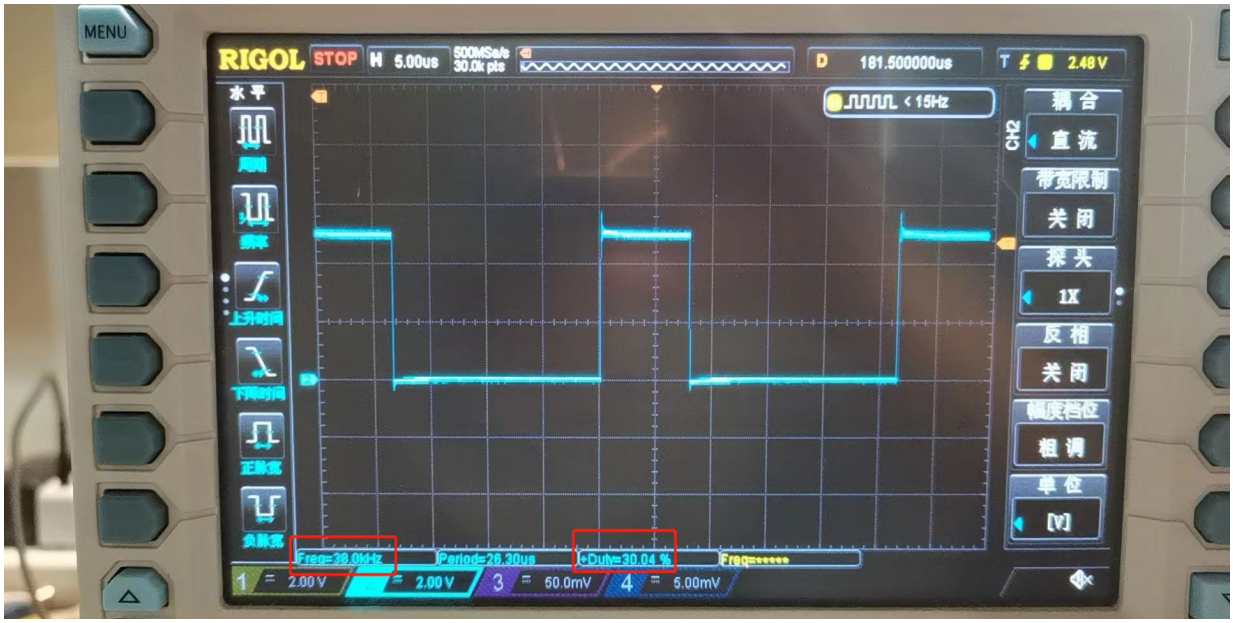

图 3.5.3 输入信号波形

实际输入的 38K 信号，占空比 30%，周期为 26.315US，高电平 7.89US

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. 通过示波器查看所示波形