文档版本	V1.0.0
发布日期	20221026

APT32F110x 基于 CSI 库 GPTA 应用指南

目录

1 概述

本文介绍了APT32F110x中通用定时器GPTA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 GPTA 模块介绍

基于 APT32F110x 完整的库文件系统，可以对 GPTA0 和 GPTA1 进行配置。以 GPTA0 为例，GPTA1 的应用，只需要把程序中的实例换为 GPTA1 即可。

- **主要特性**

GPTA 内部包含一个 16/24 位的定时/计数模块（GPTA0 为 24 位; GPTA1 为 16 位）,支持 2 种工作模式,捕捉模式和波形发生器模式)。

GPTA0_CHA,GPTA0_CHB 和 GPTA1_CHA,GPTA1_CHB 分别是 GPTA0 和 GPTA1 在 GPIO 口上映射的双向输入输出口。

GPTA 模块能和其它外设一起，被同步触发。

GPTA 的事件触发，能作为同步触发信号源。

GPTA 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**

每个 GPTA 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，波形生成控制模块，捕捉模块，同步触发控制模块（输入），事件触发模块（输出），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
GPTA_CHA	时钟控制使能	输出波形	输出波形
GPTA_CHB	时钟控制使能	输出波形	输出波形

图 3.1.1 管脚描述

GPTA_CHA 和 GPTA_CHB 是 GPTA 在 GPIO 上映射的双向输入输出端口。在群脉冲模式下（GPTA_CR[BURST]使能），GPTA_CHA 和 GPTA_CHB 可以作为门控时钟的时钟控制输入信号。

- **模块框图：**

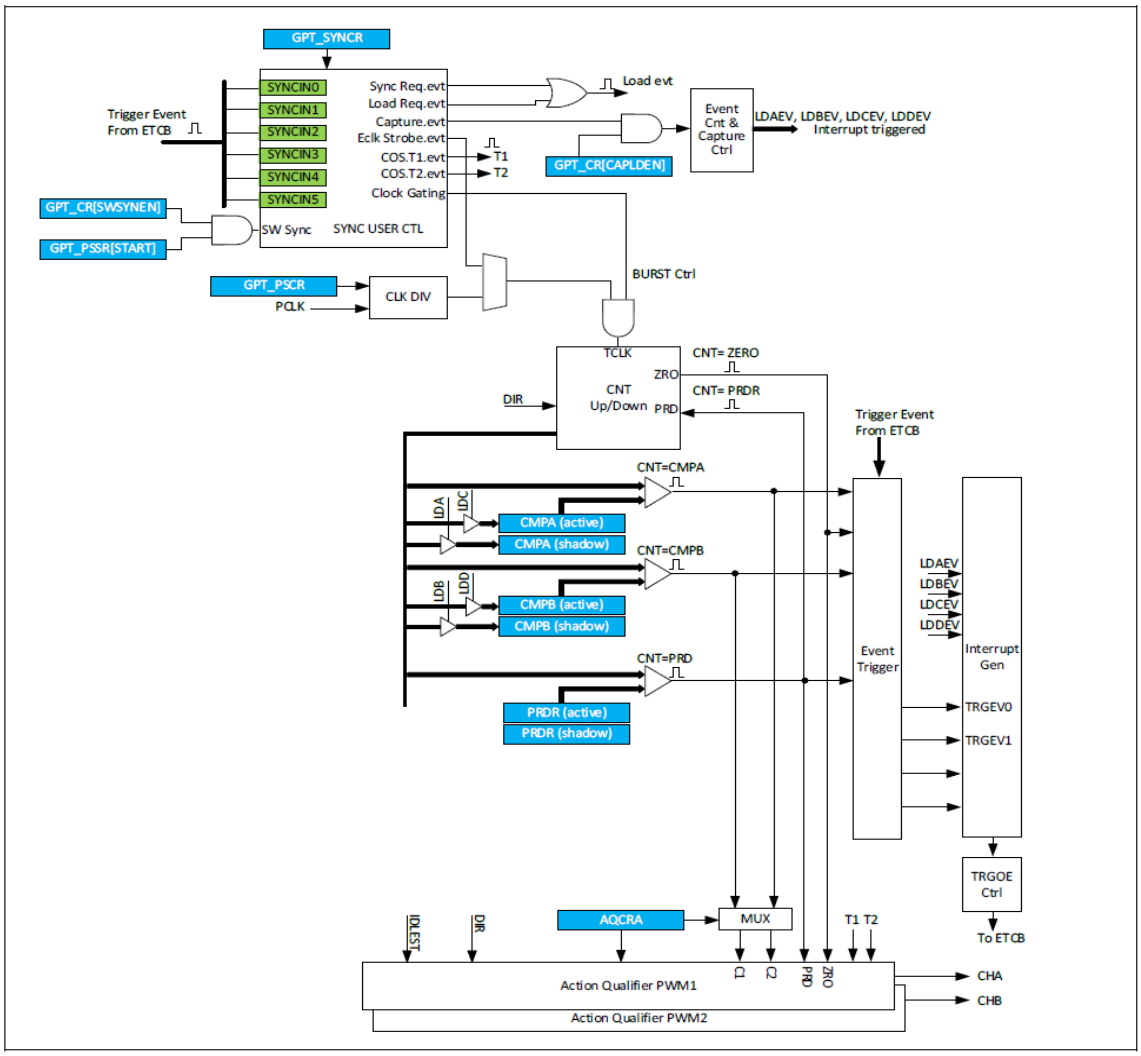

图 3.1.2 模块框图

3.2 PWM 波形输出

可在 system.c 文件中 gpta0_pwm_config ()函数进行初始化的配置。

可根据 gpta0_pwm_config ()，修改内部的对象名 GPTA0 为 GPTA1，对模块 GPTA1 进行配置。注意 GPTA0 的定时器为 24 位; GPTA1 的定时器为 16 位。

```
void gpta0_pwm_config (void)

{

    csi_pin_set_mux(PB010,        PB010_GPTA0_CHA);//28

    csi_pin_set_mux(PB011,        PB011_GPTA0_CHB);//29

    //csi_pin_set_mux(PB04,      PB04_GPTA1_CHA);

    //csi_pin_set_mux(PA10,      PA10_GPTA1_CHB);

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

   //csi_gpta_channel_cmpload_config(GPTA0,GPTA_CMPLD_SHDW,GPTA_LDCMP_ZRO ,GPTA_CAMPA);

    //csi_gpta_channel_cmpload_config(GPTA0,GPTA_CMPLD_SHDW,GPTA_LDCMP_ZRO ,GPTA_CAMPB);

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

    csi_gpta_pwmconfig_t tPwmCfg;

    tPwmCfg.byWorkmod          = GPTA_WAVE;

    tPwmCfg.byCountingMode     = GPTA_UPDNCNT;

    tPwmCfg.byOneshotMode      = GPTA_OP_CONT;

    tPwmCfg.byStartSrc         = GPTA_SYNC_START;

    tPwmCfg.byPscld            = GPTA_LDPSCR_ZRO;

    tPwmCfg.byDutyCycle        = 20;

    tPwmCfg.wFreq                    = 10000;

    tPwmCfg.wInt               = GPTA_INTSRC_CBU;

    csi_gpta_wave_init(GPTA0, &tPwmCfg);

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

    //csi_gpta_channel_aqload_config(GPTA0,GPTA_LD_SHDW,GPTA_LDCMP_PRD ,

GPTA_CHANNEL_1);

//csi_gpta_channel_aqload_config(GPTA0,GPTA_LD_SHDW,GPTA_LDCMP_PRD ,

GPTA_CHANNEL_2);



    csi_gpta_pwmchannel_config_t   tEptchannelCfg;

    tEptchannelCfg.byActionZro    =    GPTA_LO;

    tEptchannelCfg.byActionPrd    =    GPTA_NA;
```

```
    tEptchannelCfg.byActionC1u    =    GPTA_HI;
    tEptchannelCfg.byActionC1d    =    GPTA_LO;
    tEptchannelCfg.byActionC2u    =    GPTA_HI;
    tEptchannelCfg.byActionC2d    =    GPTA_LO;
    tEptchannelCfg.byActionT1u    =    GPTA_LO;
    tEptchannelCfg.byActionT1d    =    GPTA_LO;
    tEptchannelCfg.byActionT2u    =    GPTA_NA;
    tEptchannelCfg.byActionT2d    =    GPTA_NA;
    tEptchannelCfg.byChoiceC1sel  =    GPTA_CMPA;
    tEptchannelCfg.byChoiceC2sel  =    GPTA_CMPA;

csi_gpta_channel_config(GPTA0,&tEptchannelCfg,GPTA_CHANNEL_1);   tEptchannelCfg.byChoiceC1sel   =    GPTA_CMPB;
    tEptchannelCfg.byChoiceC2sel  =    GPTA_CMPB;
    csi_gpta_channel_config(GPTA0, &tEptchannelCfg,   GPTA_CHANNEL_2);
    //-------------------------------------------------------------------------
    //csi_gpta_Global_load_control_config_t   Globaldemo;
    //Globaldemo.bGlden   =       DISABLE;
    //Globaldemo.byGldmd= GPTA_LDGLD_SW;
    //Globaldemo.bOstmd           =       GPTA_LDMD_ANYTIME;
    //Globaldemo.bGldprd          =       0 ;
    //Globaldemo.byGldcnt   =     0 ;
    //csi_gpta_global_config(GPTA0,&Globaldemo);
    //csi_gpta_gldcfg(GPTA0 ,bycmpa_A ,ENABLE);
    //csi_gpta_gldcfg(GPTA0 ,bycmpb_A ,ENABLE);
    //csi_gpta_global_rearm(GPTA0) ;
    //csi_gpta_global_sw(GPTA0) ;
    //-------------------------------------------------------------------------
    //csi_gpta_set_evtrg(GPTA0, GPTA_TRG_OUT0, GPTA_TRG01_ZRO);
    //csi_gpta_set_evtrg(GPTA0, GPTA_TRG_OUT1, GPTA_TRG01_PRD);
```

```
    //csi_gpta_int_enable(GPTA0, GPTA_INTSRC_TRGEV0 , ENABLE);
    //csi_gpta_int_enable(GPTA0, GPTA_INTSRC_TRGEV1 , ENABLE);
    //csi_gpta_set_evcntinit(GPTA0, GPTA_TRG_OUT0, 5, 0);
    //-------------------------------------------------------------------------
    //csi_gpta_feglk_config_t   FEGLKcfg;
    //FEGLKcfg.byPrdr      = 0;
    //FEGLKcfg.byRssr      = 1;
    //FEGLKcfg.byCmpa      = 1;                                    //FEGLKcfg.byCmpb      = 1;
                                                                  //FEGLKcfg.byGld2      = 0;

    //FEGLKcfg.byEmslclr = 1;
    //FEGLKcfg.byEmhlclr = 1;
    //FEGLKcfg.byEmicr    = 1;
    //FEGLKcfg.byEmfrcr   = 1;
    //FEGLKcfg.byAqosf    = 1;
    //FEGLKcfg.byAqcsf    = 1;
    //csi_gpta_reglk_config(GPTA0,&FEGLKcfg);
    //-------------------------------------------------------------------------
    csi_gpta_start(GPTA0);
}
```

- **代码说明：**

1. csi_pin_set_mux(PB010，PB010_GPTA0_CHA);

用于配置 GPTA0/1 映射到 GPIO 上的输入输出口。

2.csi_gpta_wave_init(GPTA0, &tPwmCfg);

根据 tPwmCfg 结构体初始化赋值，对 GPTA0/1 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，分为软件使能和同步触发使能控制

tPwmCfg.byPscld--- GPTA 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---GPTA 模块内的中断方式使能，在 csi_gpta_wave_init()内已打开 CPU 内核对应 GPTA 模块的中断

3.csi_gpta_channel_config(GPTA0, &tEptchannelCfg, GPTA_CHANNEL_1);

根据结构体 **tEptchannelCfg** 的赋值，对 GPTA0/1 的 CHA 和 CHB 两个波形输出通道进行功能配置。 由于 GPTA0/1 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。

tEptchannelCfg.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

tEptchannelCfg.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

tEptchannelCfg.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

tEptchannelCfg.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

tEptchannelCfg.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

tEptchannelCfg.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

tEptchannelCfg.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

tEptchannelCfg.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

tEptchannelCfg.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

tEptchannelCfg.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

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

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

4. csi_gpta_channel_cmpload_config(GPTA0,GPTA_CMPLD_SHDW,GPTA_LDCMP_ZRO , GPTA_CAMPA);

对应模块，对应通道的 CMP 比较值寄存器载入方式和条件的配置。

GPTA0/1---选择对应模块

GPTA_CMPLD_SHDW/ GPTA_CMPLD_IMM---CMP 值载入模式：Immediate/Shadow

注意 ：如果更新的比较值比更新前小，且立即更新发生时计数器已经超过更新的比较值，本周期将不会发生 match 事件。

GPTA_LDCMP_ZRO---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。

GPTA_CAMPA---选择 CHA 或者 CHB 通道的 CMP 寄存器。

5. **csi_gpta_channel_aqload_config(GPTA0,GPTA_LD_SHDW,GPTA_LDCMP_PRD, GPTA_CHANNEL_1);**

配置对应模块对应通道的 AQLDR 波形控制寄存器的载入模式：Immediate/Shadow

GPTA0/1---选择对应模块

GPTA_LD_IMM /GPTA_LD_SHDW---波形输出控制寄存器的载入模式选择

GPTA_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。

GPTA_CHANNEL_1---选择 CHA 或者 CHB 通道下的 AQLDR

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。

6. **csi_gpta_global_config(GPTA0,&Globaldemo);**

全局载入控制寄存器配置，基于结构体 **Globaldemo** 初始值。

GPTA0/1---选择对应模块

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

Globaldemo.byGldmd---全局载入触发事件选择

Globaldemo.bOstmd---单次载入模式使能设置

Globaldemo.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

Globaldemo.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_gpta_gldcfg(GPTA0 ,bycmpa_A ,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。

GPTA0/1---选择对应模块

bycmpa_A---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_gpta_global_rearm(GPTA0) ;**

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

9. **csi_gpta_global_sw(GPTA0) ;**

软件产生一次全局触发事件

10. **csi_gpta_set_evtrg(GPTA0, GPTA_TRGOUT0, GPTA_TRG01_ZRO);**

GPTA 事件触发输出配置

GPTA0/1---选择对应模块

GPTA_TRGOUT0---事件触发端口选择 0~1

GPTA_TRG01_ZRO---触发端口对应的事件选择

11. **csi_gpta_int_enable(GPTA0, GPTA_INTSRC_TRGEV0 , ENABLE);**

使能 GPTA 内部的中断和 CPU 内核中断

GPTA0/1---选择对应模块

GPTA_INTSRC_TRGEV0---GPTA 模块内部中断源选择

12. **csi_gpta_set_evcntinit(GPTA0, GPTA_TRG_OUT0, 5, 0);**

GPTA 事件触发计数器设置，TRGSEL0 每 5 次产生一次事件输出。

GPTA0/1---选择对应模块

GPTA_TRGOUT0---选择触发端口

“5”---计数次数

“0”---计数起始值

13. csi_gpta_reglk_config(GPTA0,&FEGLKcfg);

GPTA0/1 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。

GPTA0/1---作为目标模块

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_gpta_start(GPTA0);

启动对应模块

- **波形输出：**

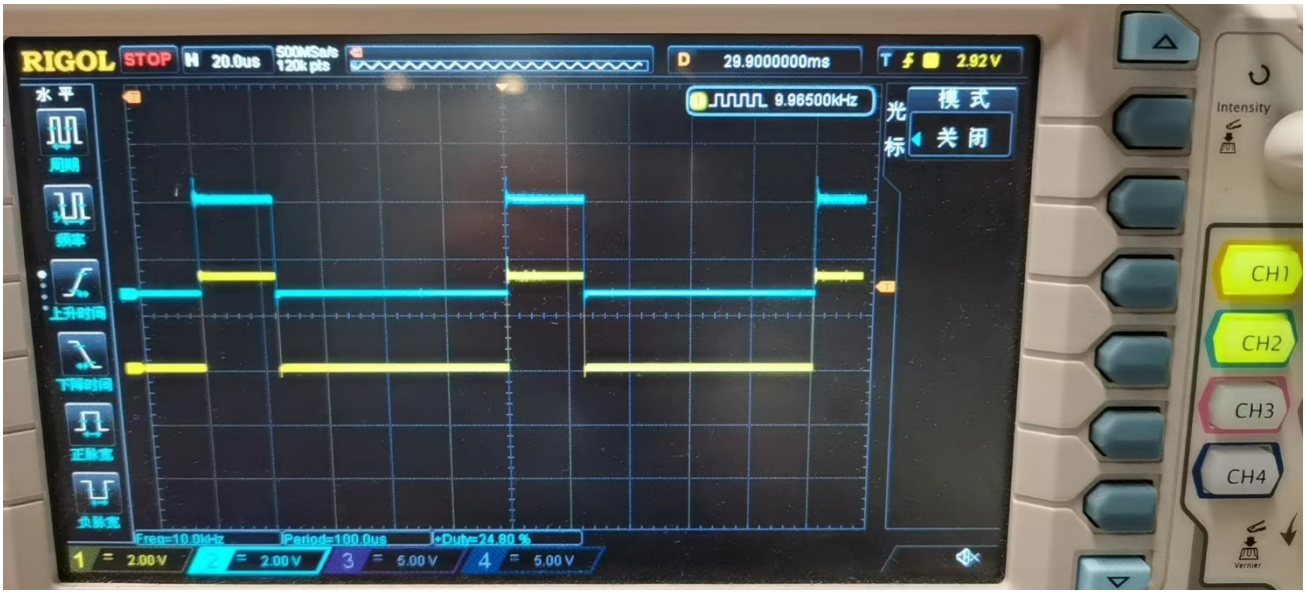

图 3.2.1 波形输出

3.3 捕获模式

- **模块框图：**

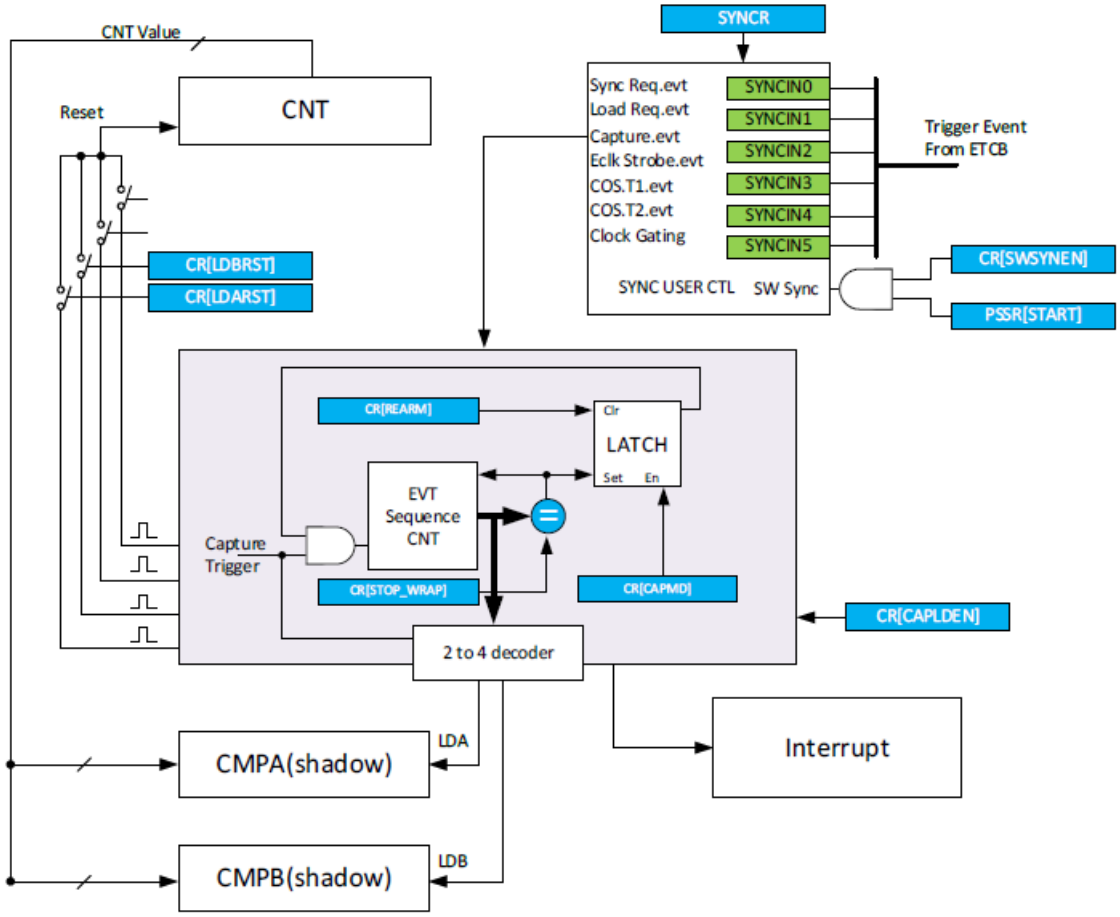

图 3.3.1 模块框图

可在 system.c 文件中 gpta0_capture_config()函数进行初始化的配置。

```
//---------------GPTA 捕获模式-------------------------------------

void gpta0_capture_config(void)

{

    uint8_t ch;

//------------input gpio exi ----------------------------------

    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, 1);
//----------------ETCB--------------------------------------
    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_GPTA0_SYNCIN2;
    tEtbConfig.byDstIp1   = 0xff;
    tEtbConfig.byDstIp2   = 0xff;
    tEtbConfig.byTrgMode = ETB_HARDWARE_TRG;
    csi_etb_init();
    ch = csi_etb_ch_alloc(tEtbConfig.byChType);
//  if(ch < 0)return -1;                                    //ch < 0,则获取通道号失败
    csi_etb_ch_config(ch, &tEtbConfig);
//---------------gpta0-------------------------------------------------
            csi_gpta_captureconfig_t tPwmCfg;
            tPwmCfg.byWorkmod          = GPTA_CAPTURE;
            tPwmCfg.byCountingMode   = GPTA_UPCNT;
            tPwmCfg.byOneshotMode     = GPTA_OP_CONT;
            tPwmCfg.byStartSrc           = GPTA_SYNC_START;
            tPwmCfg.byPscld               = GPTA_LDPSCR_ZRO;
            tPwmCfg.byCaptureCapmd    = 0;
            tPwmCfg.byCaptureStopWrap=2-1;
            tPwmCfg.byCaptureLdaret   =0;
            tPwmCfg.byCaptureLdbret   =1;
            tPwmCfg.wInt             =GPTA_INTSRC_CAPLD0;
    //      tPwmCfg.byBurst     =true;
    //      tPwmCfg.byCgsrc     = GPTA_CGSRC_TIOB;
```

```
    //      tPwmCfg.byCgflt     = GPTA_CGFLT_BP;
            csi_gpta_capture_init(GPTA0, &tPwmCfg);
    //------------------------------------------------------------------
            csi_gpta_set_sync(GPTA0,GPTA_TRG_SYNCEN2,GPTA_TRG_CONTINU,
                              GPTA_AUTO_REARM_ZRO);
    //      csi_gpta_set_extsync_chnl(GPTA0, GPTA_TRG_SYNCEN2,0);
    //      csi_gpta_set_evtrg(GPTA0, GPTA_TRG_OUT0, GPTA_TRG01_SYNC);
    //      csi_gpta_int_enable(GPTA0, GPTA_INTSRC_TRGEV0,true);
    //------------------------------------------------------------------
    //      csi_gpta_filter_config_t    tpFiltercfg;
    //      tpFiltercfg.byFiltSrc               =GPTA_FILT_SYNCIN2;
    //      tpFiltercfg.byWinInv                =1;
    //      tpFiltercfg.byWinAlign              =GPTA_ALIGN_ZRO;

    //      tpFiltercfg.byWinCross                  =1;
    //      tpFiltercfg.byWinOffset             =0xffff/2;
    //      tpFiltercfg.byWinWidth                  =0xffff/2;
    //      csi_gpta_set_sync_filter(GPTA0, &tpFiltercfg);
    //------------------------------------------------------------------
            csi_gpta_start(GPTA0);//start    timer
}
```

- **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_BOTH_EDGE);**

配置对应 GPIO 的外部中断组，设置双沿触发

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：0~11 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

GPTA0 部分

7) **csi_gpta_capture_init(GPTA0, &tPwmCfg);**

根据结构体变量 **tPwmCfg** 初始化值配置捕捉模块

tPwmCfg.byWorkmod---GPTA0 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---GPTA 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_gpta_set_sync(GPTA0,GPTA_TRG_SYNCEN2,GPTA_TRG_CONTINU, GPTA_AUTO_REARM_ZRO);**

选择外部触发通道，并配置工作模式和重载模式

GPTA_TRG_SYNCEN2---选 SYNCIN2 为外部同步触发通道

GPTA_TRG_CONTINU---工作模式：连续 OR 单次

9) csi_gpta_start(GPTA0);

启动 GPTA 计数器

GPTA0 中断部分

__attribute__((weak)) void gpta0_initen_irqhandler(csp_gpta_t *ptGptaBase){}

里面的如下部分：

```
if(((csp_gpta_get_misr(ptGptaBase)&GPTA_INT_CAPLD0))==GPTA_INT_CAPLD0)
  {
     val_buff_t[0]=csp_gpta_get_cmpa(ptGptaBase);
     val_buff_t[1]=csp_gpta_get_cmpb(ptGptaBase);
     csp_gpta_clr_int(ptGptaBase, GPTA_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。

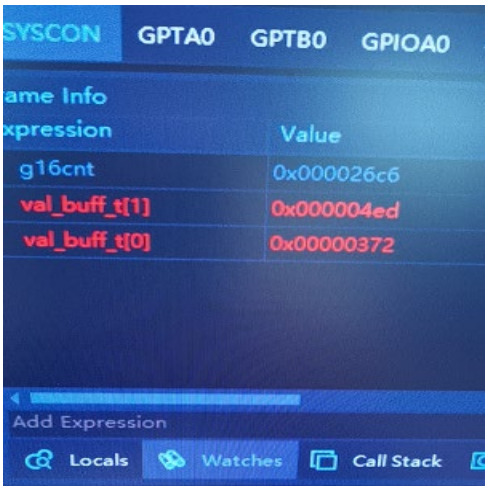

图 3.3.2 捕捉数值

计算公式：val_buff_t[]/48000000　　48000000 为 GPTA 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

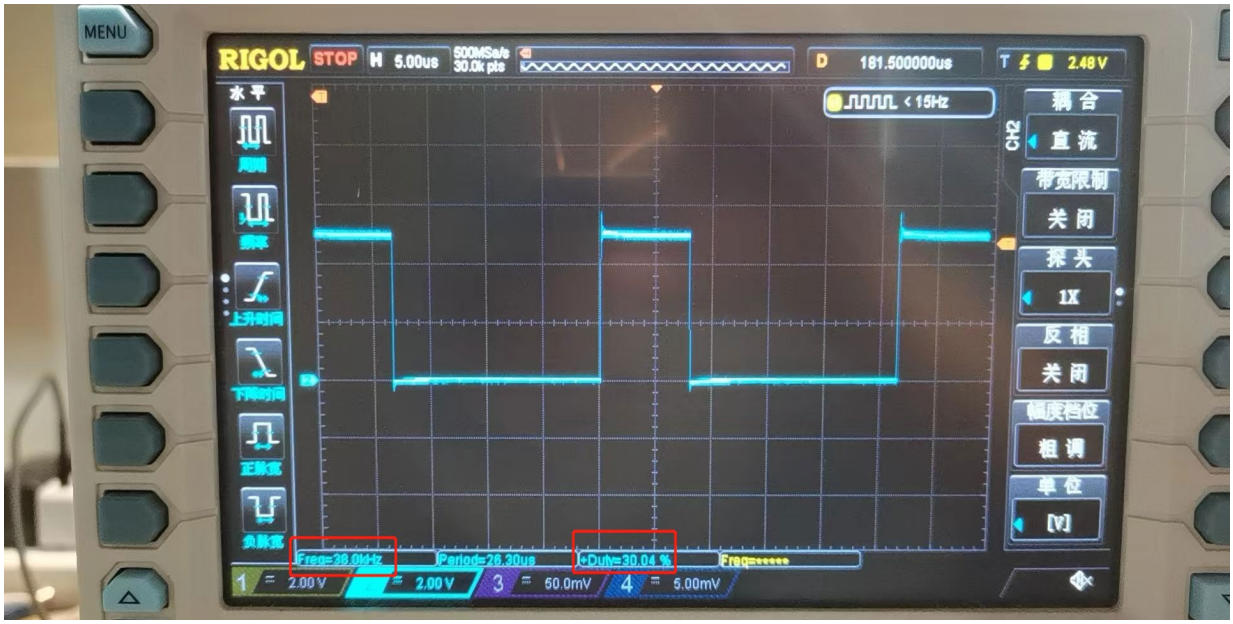

图 3.3.3 输入信号

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

3.4 同步触发输入滤波

csi_gpta_set_sync_filter(GPTA0, &tpFiltercfg);

外部触发输入窗口滤波器根据结构体变量 **tpFiltercfg** 的初始值配置

tpFiltercfg.byFiltSrc---滤波模块的输入信号源选择

tpFiltercfg.byWinInv ---窗口使能反转控制

0h：窗口不反转，窗口有效区间禁止滤波输入;

1h：窗口反转，窗口有效区间使能滤波输入

tpFiltercfg.byWinAlign---窗口对齐模式选择

tpFiltercfg.byWinCross---滤波窗跨周期使能:

0h：禁止跨窗口对齐点; 1h：允许跨窗口对齐点

tpFiltercfg.byWinOffset---滤波窗偏移位置调整

tpFiltercfg.byWinWidth---滤波窗宽度

3.5同步触发输出

1) **csi_gpta_set_extsync_chnl(GPTA0, GPTA_TRG_SYNCEN2,0);**

SYNCIN2---TRGSEL0 GTPA_SYNCR 输入触发通道直通作为 TRGSRC0 的 ExtSync 条件的选择

“0”--- TRGSEL0/1 通道选择

2) **csi_gpta_set_evtrg(GPTA0,GPTA_TRGOUT0 , GPTA_TRG01_SYNC);**

GPTA_TRGOUT0--- 触发事件通道

GPTA_TRG01_SYNC---触发事件的源选择

3) **csi_gpta_int_enable(GPTA0, GPTA_INTSRC_TRGEV0,true);**

GPTA 内部 TRGEV0 中断使能，同时使能对应的 CPU 中断。

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