

文档版本	V1.0
发布日期	20210610

APT32F102 GPT 应用指南

APTCHIP



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 GPT 配置.....	1
3.2 GPT PWM 配置.....	4
3.3 GPT 输入捕捉配置.....	6
4. 程序下载和运行	10

1 概述

本文介绍了在APT32F102中使用GPT的应用范例。

2. 适用的硬件

该例程使用于 APT32F102x 系列学习板

3. 应用方案代码说明

3.1 GPT 配置

基于 APT32F102 完整的库文件系统，可以对 GPT 进行配置。

- **硬件配置：**

GPT 是一个 16 位的通用定时器模块，支持波形发生和捕捉模式。有两路独立的 PWM 输出，支持同步触发控制。GPTA0 代表第一个 GPTA 模块，GPTA_CHA 和 GPTA_CHB 是 GPTA 在 GPIO 上映射的双向输入输出端口。

GPT 的周期寄存器是由两个物理寄存器活动寄存器和影子寄存器组成，它们是共享一个物理访问地址，可以通过设置来操作。

注意 GPT 下的触发优先级是由硬件决定。

- **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 管脚功能定义

- **软件配置：**

可在 apt32f102_initial.c 文件中 GPT_CONFIG 进行初始化的配置

```

/*****
//GPT0 Functions
//EntryParameter:NONE
//ReturnValue:NONE
*****/

void GPT0_CONFIG(void)
{
    GPT_IO_Init(GPT_CHA_PA09);
    GPT_IO_Init(GPT_CHB_PA010);
    GPT_Configure(GPTCLK_EN,GPT_PCLK,GPT_IMMEDIATE,1);
    GPT_WaveCtrl_Configure(GPT_INCREASE,GPT_SWSYNDIS,GPT_IDLE_LOW,GPT_PRDLT_PEND,GPT_OPM_CONTINUOUS,GPT_BURST_DIS,GPT_
CKS_PCLK,GPT_CG_CHAX,GPT_CGFLT_00,GPT_PRDLT_ZERO);
    GPT_Period_CMP_Write(2000,1500,50);
    GPT_WaveLoad_Configure(GPT_WAVEA_IMMEDIATE,GPT_WAVEB_SHADOW,GPT_AQLDA_ZERO,GPT_AQLDB_ZERO);
    GPT_WaveOut_Configure(GPT_CHA,GPT_CASEL_CMPA,GPT_CBSEL_CMPA,2,0,1,1,0,0,0,0,0);
    GPT_WaveOut_Configure(GPT_CHB,GPT_CASEL_CMPA,GPT_CBSEL_CMPB,2,0,0,0,1,1,0,0,0);
    //GPT_SyncSet_Configure(GPT_SYNCUSR0_EN,GPT_OST_CONTINUOUS,GPT_TXREARM_DIS,GPT_TRG00SEL_SR0,GPT_TRG10SEL_SR0,GPT_A
REARM_DIS);
    //GPT_Trigger_Configure(GPT_SRCSEL_TRGUSR0EN,GPT_BLKINV_DIS,GPT_ALIGNMD_PRD,GPT_CROSSMD_DIS,5,5);
    //GPT_EVTRG_Configure(GPT_TRGSRC0_PRD,GPT_TRGSRC1_PRD,GPT_ESYN00E_EN,GPT_ESYN10E_EN,GPT_CNT0INIT_EN,GPT_CNT1INIT_E
N,3,3,3,3);
    GPT_Start();
    GPT_ConfigInterrupt_CMD(ENABLE,GPT_INT_PEND);
    GPT_INT_ENABLE();
    //INTC_ISER_WRITE(GPT0_INT);
    //INTC_IWER_WRITE(GPT0_INT);
}

```

● 代码说明：

GPT_IO_Init(); ----用于配置 GPIO 为 GPT 复用功能

GPT_Configure(); ----用于配置 GPT 时钟

GPT_WaveCtrl_Configure(); ----用于配置 GPT 工作模式

GPT_Period_CMP_Write(); ----用于配置 GPT 的周期和比较值

GPT_WaveLoad_Configure(); ----用于配置 GPT 波形输出载入

GPT_WaveOut_Configure(); ----用于波形输出控制

GPT_SyncSet_Configure(); ----用于配置触发使能

GPT_Trigger_Configure(); ----用于配置数字比较器滤波

GPT_EVTRG_Configure(); ----用于配置事件触发选择

GPT_Start(); ----用于启动 GPT 计数器

GPT_ConfigInterrupt_CMD();----用于配置 GPT 中断

GPT_INT_ENABLE(); ----用于开启 GPT 中断

INTC_ISER_WRITE();----用于使能 GPT 对应中断

INTC_IWER_WRITE(); ----用于使能 GPT 低功耗唤醒功能

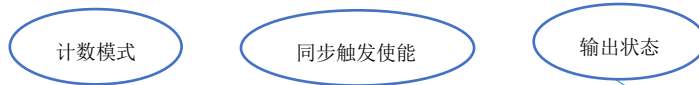
● 参数说明:



GPT_IO_Init(GPT_CHA_PA09);



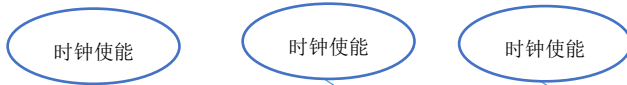
GPT_Configure(GPTCLK_EN,GPT_PCLK,GPT_SHADOW,1);



GPT_WaveCtrl_Configure(GPT_INCREASE,GPT_SWSYNDIS,GPT_IDLE_LOW,



GPT_PRDLD_PEND,GPT_OPM_CONTINUOUS,GPT_BURST_DIS,GPT_CKS_PCLK,



GPT_CG_CHAX,GPT_CGFLT_00,GPT_PRDLD_ZERO);



GPT_Period_CMP_Write(2000,1500,50);



GPT_WaveLoad_Configure(GPT_WAVEA_IMMEDIATE,GPT_WAVEB_SHADOW,



3.2 GPT PWM 配置

选择内部主频 48MHz 作为系统时钟。 PA0.9->CHA: 输出周期为 100us，占空比为 50us
 PA0.10->CHB: 输出周期为 100us，占空比为 25us 。

```

/*****/
//GPT0 Functions
//EntryParameter:NONE
//ReturnValue:NONE
/*****/

void GPT0_CONFIG(void)
{
    GPT_IO_Init(GPT_CHA_PA09);
    GPT_IO_Init(GPT_CHB_PA010);
    GPT_Configure(GPTCLK_EN,GPT_PCLK,GPT_SHADOW,0);
    GPT_WaveCtrl_Configure(GPT_INCREASE,GPT_SWSYNDIS,GPT_IDLE_LOW,GPT_PRDLD_PEND,GPT_OPM_CONTINUOUS,GPT_BURST_DIS,GPT_
CKS_PCLK,GPT_CG_CHAX,GPT_CGFLT_00,GPT_PRDLD_ZERO);
    GPT_Period_CMP_Write(48000,24000,12000);
    GPT_WaveLoad_Configure(GPT_WAVEA_SHADOW,GPT_WAVEB_SHADOW,GPT_AQLDA_ZERO,GPT_AQLDB_ZERO);
    GPT_WaveOut_Configure(GPT_CHA,GPT_CASEL_CMPA,GPT_CBSEL_CMPA,2,0,1,0,0,0,0,0,0);
    GPT_WaveOut_Configure(GPT_CHB,GPT_CASEL_CMPA,GPT_CBSEL_CMPB,2,0,0,0,1,0,0,0,0);
    GPT_Start();
}

void APT32F102_init(void)
{
//-----/
//Peripheral clock enable and disable
//EntryParameter:NONE
//ReturnValue:NONE
//-----/
    SYSCON->PCER0=0xFFFFFFFF;           //PCLK Enable 0x00410071
    SYSCON->PCER1=0xFFFFFFFF;           //PCLK Enable
    while(!(SYSCON->PCSR0&0x1));         //Wait PCLK enabled
//-----/
//ISOSC/IMOSC/EMOSC/SYSCCLK/IWDT/LVD/EM_CMFAIL/EM_CMRCV/CMD_ERR OSC stable interrupt
//EntryParameter:NONE
//ReturnValue:NONE
//-----/

```

```

SYSCON_CONFIG(); //syscon initial
CK_CPU_EnAllNormalIrq(); //enable all IRQ
//-----/
//Other IP config
//-----/
GPT0_CONFIG(); //GPT0 initial
}
/*****/
//main
/*****/
int main(void)
{
    APT32F102_init();
    while(1)
    {
        SYSCON_IWDCNT_Reload();
    }
}

```

● **函数参数说明:**





● 输出波形:

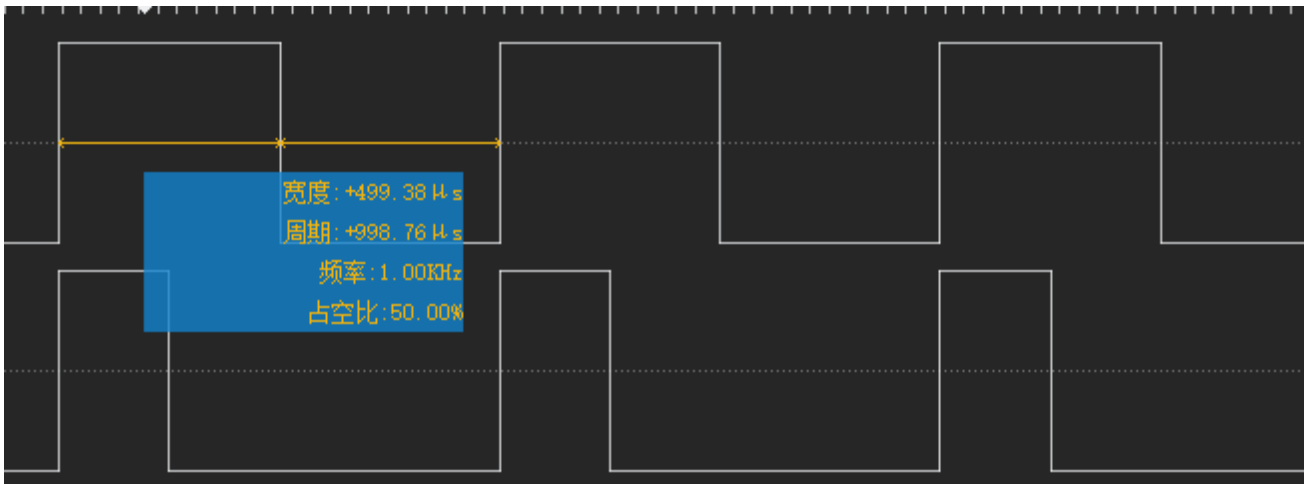


图 3.2.1 GPT 输出波形

3.3 GPT 输入捕捉配置

选择内部主频 48MHz 作为系统时钟，PA0.0 作为捕捉输入口，R_CMPA_BUF 存储低电平计数值，R_CMPB_BUF 存储周期计数值。

● 编程要点:

1. 配置 GPIO 的外部触发

2. 配置 GPT 捕获模式，设置 ETCB 选择 GPTA SYNCIN0 的输入事件

3. 配置 GPT 中断

```

/*****/
//GPIO Initial
//EntryParameter:NONE
//ReturnValue:NONE
/*****/
void GPIO_CONFIG(void)
{
//----- EXI FUNTION -----/
//EXI0_INT= EXI0/EXI16,EXI1_INT= EXI1/EXI17, EXI2_INT=EXI2-EXI3/EXI18/EXI19, EXI3_INT=EXI4-EXI9, EXI4_INT=EXI10-EXI15
GPIO_IntGroup_Set(PA0,0,Selete_EXI_PIN0); //EXI0 set PA0.0
GPIOA0_EXI_Init(EXI0); //PA0.0 as input
EXTI_trigger_CMD(ENABLE,EXI_PIN0_EXIRT);
EXTI_interrupt_CMD(ENABLE,EXI_PIN0); //enable EXI
GPIO_EXTI_interrupt(GPIOA0,0b0000000000000001);
}
/*****/
//GPT0 Functions
//EntryParameter:NONE
//ReturnValue:NONE
/*****/
void GPT0_CONFIG(void)
{
GPT_Configure(GPTCLK_EN,GPT_PCLK,GPT_SHADOW,0);
GPT_Capture_Config(EPT_CNTMD_increase,GPT_CAPMD_Continue,GPT_CAP_EN,GPT_LDARST_EN,GPT_LDBRST_DIS,GPT_LDCRST_DIS,GPT_LDDRS
T_DIS,1);
GPT0->SYNCR = 0X04;
GPT_Period_CMP_Write(0xFFFF,0,0);
GPT_ConfigInterrupt_CMD(ENABLE,GPT_INT_CAPLD0);
GPT_ConfigInterrupt_CMD(ENABLE,GPT_INT_CAPLD1);
GPT_INT_ENABLE();
GPT_Start();
}
void APT32F102_init(void)
{
//-----/
//Peripheral clock enable and disable
//EntryParameter:NONE
//ReturnValue:NONE
//-----/
SYSCON->PCER0=0xFFFFFFFF; //PCLK Enable 0x00410071

```

```

SYSCON->PCER1=0xFFFFFFFF; //PCLK Enable
while(!((SYSCON->PCSR0&0x1)); //Wait PCLK enabled
//-----/
//ISOSC/IMOSC/EMOSC/SYSCCLK/IWDT/LVD/EM_CMFAIL/EM_CMRCV/CMD_ERR OSC stable interrupt
//EntryParameter:NONE
//ReturnValue:NONE
//-----/
    SYSCON_CONFIG(); //syscon initial
    CK_CPU_EnAllNormalIrq(); //enable all IRQ
//-----/
//Other IP config
//-----/
    GPIO_CONFIG(); //GPIO initial
    GPT0_CONFIG(); //GPT0 initial
}
/*****/
//main
/*****/
int main(void)
{
    APT32F102_init();
    while(1)
    {
        SYSCON_IWDCNT_Reload();
    }
}
/*****/
//GPT0 Interrupt
//EntryParameter:NONE
//ReturnValue:NONE
/*****/
void GPT0IntHandler(void)
{
    // ISR content ...
    //Interrupt
    if((GPT0->MISR&GPT_INT_TRGEV0)==GPT_INT_TRGEV0)
    {
        GPT0->ICR = GPT_INT_TRGEV0;
    }
    else if((GPT0->MISR&GPT_INT_TRGEV1)==GPT_INT_TRGEV1)
    {
        GPT0->ICR = GPT_INT_TRGEV1;
    }
}

```

```

else if((GPT0->MISR&GPT_INT_TRGEV2)==GPT_INT_TRGEV2)
{
    GPT0->ICR = GPT_INT_TRGEV0;
}
else if((GPT0->MISR&GPT_TRGEV3)==GPT_INT_TRGEV3)
{
    GPT0->ICR = GPT_INT_TRGEV3;
}
else if((GPT0->MISR&GPT_INT_CAPLD0)==GPT_INT_CAPLD0)
{
    GPT0->ICR = GPT_INT_CAPLD0;
    EXTI_trigger_CMD(DISABLE, EXI_PIN0, _EXIRT);
    EXTI_trigger_CMD(ENABLE, EXI_PIN0, _EXIFT);
    R_CMPA_BUF = GPT0->CMPA;
}
else if((GPT0->MISR&GPT_INT_CAPLD1)==GPT_INT_CAPLD1)
{
    GPT0->ICR = GPT_INT_CAPLD1;
    EXTI_trigger_CMD(ENABLE, EXI_PIN0, _EXIRT);
    EXTI_trigger_CMD(DISABLE, EXI_PIN0, _EXIFT);
    R_CMPB_BUF = GPT0->CMPB;
}
else if((GPT0->MISR&GPT_INT_CAPLD2)==GPT_INT_CAPLD2)
{
    GPT0->ICR = GPT_INT_CAPLD2;
}
else if((GPT0->MISR&GPT_INT_CAPLD3)==GPT_INT_CAPLD3)
{
    GPT0->ICR = GPT_INT_CAPLD3;
}
else if((GPT0->MISR&GPT_INT_CAU)==GPT_INT_CAU)
{
    GPT0->ICR = GPT_INT_CAU;
}
else if((GPT0->MISR&GPT_INT_CAD)==GPT_INT_CAD)
{
    GPT0->ICR = GPT_INT_CAD;
}
else if((GPT0->MISR&GPT_INT_CBU)==GPT_INT_CBU)
{
    GPT0->ICR = GPT_INT_CBU;
}

```

```

else if((GPT0->MISR&GPT_INT_CBD)==GPT_INT_CBD)
{
    GPT0->ICR = GPT_INT_CBD;
}
else if((GPT0->MISR&GPT_INT_PEND)==GPT_INT_PEND)
{
    GPT0->ICR = GPT_INT_PEND;
}
}
    
```

● 代码说明:

GPT_Configure(); ----用于配置 GPT 时钟

GPT_Capture_Config(); ----用于配置 GPT 捕捉模式

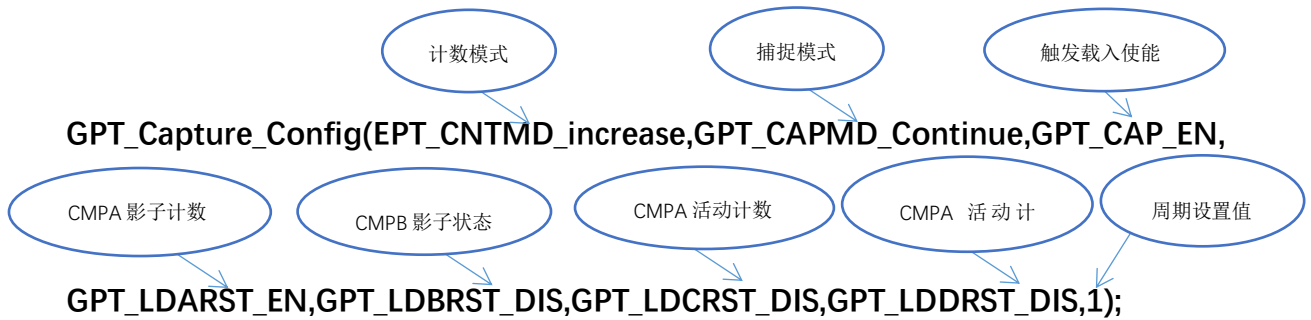
GPT_Period_CMP_Write(); ----用于周期设置

GPT_ConfigInterrupt_CMD(); ----用于配置 GPT 中断

GPT_INT_ENABLE(); ----用于开启中断

GPT_Start(); ----用于启动 GPT

● 函数参数说明:



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，图 3.2.1 所示波形