文档版本	V1.0
发布日期	20210618

APT32F102 ADC 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 ETCB 配置 1

3.2 ETCB 事件对应表 3

3.3 GPIO 触发 BT 输出 7

3.4 BT0 触发 LPT 输出 10

4. 程序下载和运行 12

1 概述

本文介绍了在APT32F102中使用ETCB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 ETCB 配置

基于 APT32F102 完整的库文件系统，可以对 ETCB 进行配置。

- **硬件配置：**

ETCB 模块是用来将一个 IP 的信息传递到另一个 IP，可以有效的减少对 CPU 的中断请求，从而降低 CPU 的负载。8 个可配置的触发通道，其中通道 0 支持多个源触发单个目标，通道 1-2 支持单个源触发多个目标。剩下的只能单一源触发单一目标。

- **注意:**

如果某个 IP 的事件源一直不停的以一个非常高的频率触发，那么 ETCB 模块有可能会丢失一部分触发信号。

事件源输出触发后，ETCB 模块需要一个时钟处理事件转发，即一个时钟后事件到目标模块。

一个目标只能被一个通道选择， 如果 2 个或者多个通道都选择了同一个目标，那么序号小的目标有更高的优先级。

- **模块框图：**

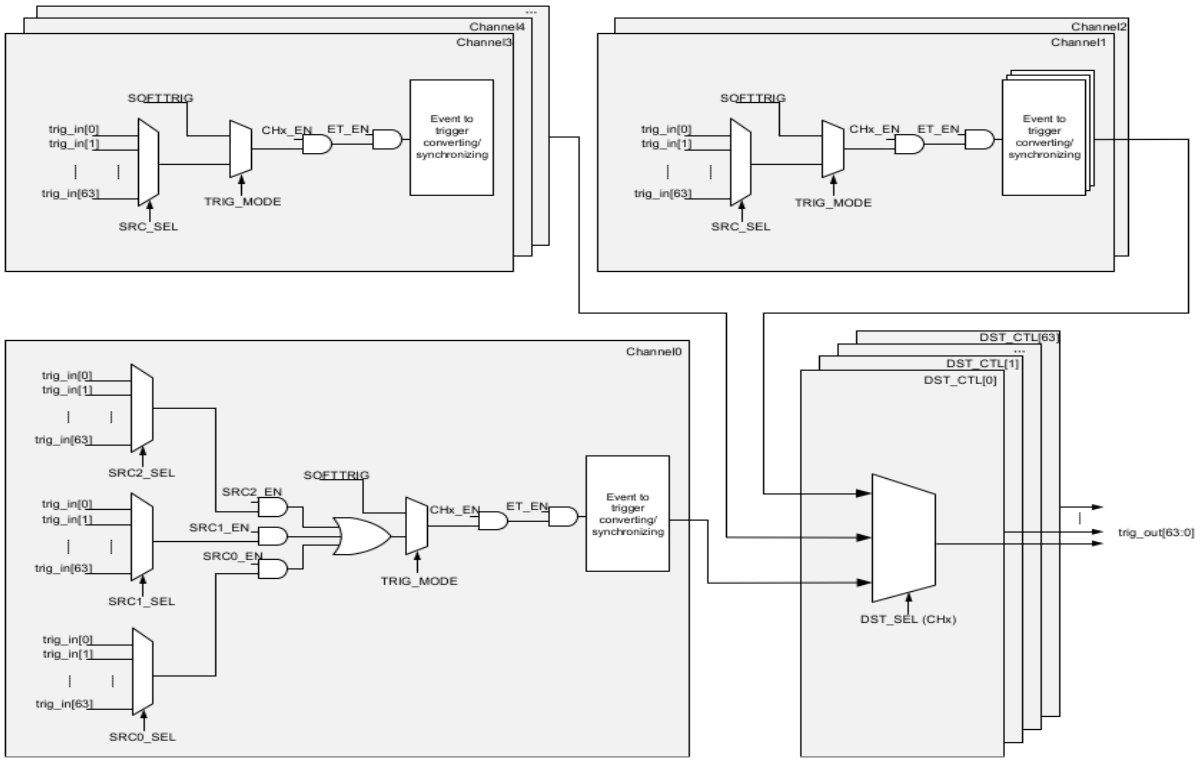


图 3.1.1 ETCB 框图

- **软件配置：**

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

```
/*************************************************************/
//ET Initial
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
void ET_CONFIG(void)
{
  ET_DeInit();
  //BT0 触发 LPT
  /*ET_CH0_SRCSEL(ET_SRC0,ENABLE,ET_BT_SYNC0);
  ET_CH0_CONTROL(ENABLE,TRG_HW,ET_LPT_TRGSRC);*/
  //LPT 触发 BT0
  /*ET_CH0_SRCSEL(ET_SRC0,ENABLE,ET_LPT_SYNC);
  ET_CH0_CONTROL(ENABLE,TRG_HW,ET_BT0_TRGSRC0);*/
  //GTP0 SYNC0  触发 LPT
  ET_CH0_SRCSEL(ET_SRC0,ENABLE,ET_EXI_SYNC0);
  ET_CH0_CONTROL(ENABLE,TRG_HW,ET_EPT0_TRGSRC3);
  //LPT 触发 GTP0 SYNC0
  /*ET_CH0_SRCSEL(ET_SRC0,ENABLE,ET_LPT_SYNC);
```

```
    ET_CH0_CONTROL(ENABLE,TRG_HW,ET_GPT0_TRGSRC0);*/
    ET_ENABLE();
  }
```

- **代码说明：**

  ET_DeInit(); ----用于重置所有寄存器的值

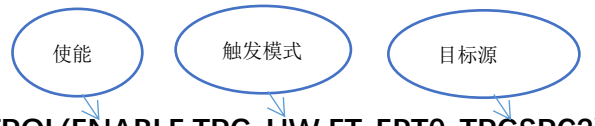
  ET_CH0_SRCSEL(); ----用于配置通道 0 的触发源

  ET_CH0_CONTROL(); ----用于配置通道 0 的目标源

  ET_ENABLE(); ----用于使能 ETCB

- **函数参数说明：**

ET_CH0_SRCSEL(ET_SRC0,ENABLE,ET_EXI_SYNC0);



ET_CH0_CONTROL(ENABLE,TRG_HW,ET_EPT0_TRGSRC3);

3.2 ETCB 事件对应表

事件源都是来自片上各 IP 模块。当 IP 在工作时，这些事件就会产生，而并不需要相应的中断使能。事件序号与 IP 的对应关系如下表格。

源序号	事件源	目标序号	目标事件
0 （0H）	LPT_TRGOUT0	0 （0H）	LPT_SYNCIN0
1 （1H）	RSVD	1 （1H）	RSVD
2 （2H）	RSVD	2 （2H）	BT0_SYNCIN0
3 （3H）	RSVD	3 （3H）	BT0_SYNCIN1

4 （4H）	EXI_ TRGOUT0	4 （4H）	BT1_SYNCIN0
5 （5H）	EXI_ TRGOUT1	5 （5H）	BT1_SYNCIN1
6 （6H）	EXI_ TRGOUT2	6 （6H）	ADC_SYNCIN0
7 （7H）	EXI_TRGOUT3	7 （7H）	ADC_SYNCIN1
8 （8H）	EXI_TRGOUT4	8 （8H）	ADC_SYNCIN2
9 （9H）	EXI_TRGOUT5	9 （9H）	ADC_SYNCIN3
10 （AH）	RTC_TRGOUT0	10 （AH）	ADC_SYNCIN4
11 （BH）	RTC_TRGOUT1	11 （BH）	ADC_SYNCIN5
12 （CH）	BT_TRGOUT0	12 （CH）	RSVD
13 （DH）	BT_TRGOUT1	13 （DH）	RSVD
14 （EH）	RSVD	14 （EH）	RSVD
15 （FH）	RSVD	15 （FH）	RSVD
16 （10H）	EPT0_TRGOUT0	16 （10H）	EPT0_SYNCIN0
17 （11H）	EPT0_TRGOUT1	17 （11H）	EPT0_SYNCIN1
18 （12H）	EPT0_TRGOUT2	18 （12H）	EPT0_SYNCIN2
19 （13H）	EPT0_TRGOUT3	19 （13H）	EPT0_SYNCIN3
20 （14H）	RSVD	20 （14H）	EPT0_SYNCIN4
21 （15H）	RSVD	21 （15H）	EPT0_SYNCIN5
22 （16H）	RSVD	22 （16H）	RSVD

23 （17H）	RSVD	23 （17H）	RSVD
24 （18H）	RSVD	24 （18H）	RSVD
25 （19H）	RSVD	25 （19H）	RSVD
26 （1AH）	RSVD	26 （1AH）	RSVD
27 （1BH）	RSVD	27 （1BH）	RSVD
28 （1CH）	RSVD	28 （1CH）	RSVD
29 （1DH）	RSVD	29 （1DH）	RSVD
30 （1EH）	RSVD	30 （1EH）	RSVD
31 （1FH）	RSVD	31 （1FH）	RSVD
32 （20H）	GPT0_TRGOUT0	32 （20H）	RSVD
33 （21H）	GPT0_TRGOUT1	33 （21H）	RSVD
34 （22H）	RSVD	34 （22H）	RSVD
35 （23H）	RSVD	35 （23H）	RSVD
36 （24H）	RSVD	36 （24H）	GPT0_SYNCIN0
37 （25H）	RSVD	37 （25H）	GPT0_SYNCIN1
38 （26H）	RSVD	38 （26H）	GPT0_SYNCIN2
39 （27H）	RSVD	39 （27H）	GPT0_SYNCIN3
40 （28H）	RSVD	40 （28H）	GPT0_SYNCIN4
41 （29H）	RSVD	41 （29H）	GPT0_SYNCIN5

42 （2AH）	RSVD	42 （2AH）	RSVD
43 （2BH）	RSVD	43 （2BH）	RSVD
44 （2CH）	RSVD	44 （2CH）	RSVD
45 （2DH）	RSVD	45 （2DH）	RSVD
46 （2EH）	RSVD	46 （2EH）	RSVD
47 （2FH）	RSVD	47 （2FH）	RSVD
48 （30H）	ADC_TRGOUT0	48 （30H）	RSVD
49 （31H）	ADC_TRGOUT1	49 （31H）	RSVD
50 （32H）	RSVD	50 （32H）	RSVD
51 （33H）	RSVD	51 （33H）	RSVD
52 （34H）	RSVD	52 （34H）	RSVD
53 （35H）	RSVD	53 （35H）	RSVD
54 （36H）	RSVD	54 （36H）	RSVD
55 （37H）	RSVD	55 （37H）	RSVD
56 （38H）	RSVD	56 （38H）	RSVD
57 （39H）	RSVD	57 （39H）	RSVD
58 （3AH）	RSVD	58 （3AH）	RSVD
59 （3BH）	RSVD	59 （3BH）	RSVD
60 （3CH）	TOUCH_TRGOUT	60 （3CH）	TOUCH_SYNCIN

61 （3DH）	RSVD	61 （3DH）	RSVD
62 （3EH）	RSVD	62 （3EH）	RSVD
63 （3FH）	RSVD	63 （3FH）	RSVD

3.3 GPIO 触发 BT 输出

选择内部主频 48MHz 作为系统时钟，通过 GPIO 的外部中断事件触发 BT0- PB2 输出波形.

- **编程要点：**
  1. 配置对应模块的功能以及触发事件
  2. 配置 ETCB 对应事件源和目标事件

```
/*****************************************************************/
//GPIO Initial
//EntryParameter:NONE
//ReturnValue:NONE
/*****************************************************************/
void GPIO_CONFIG(void)
{
    GPIO_PullHigh_Init(GPIOA0,4);
//------------    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(PA0,4,Selete_EXI_PIN4);
    GPIOA0_EXI_Init(EXI4);                                  //PA0.4 as input
    EXTI_trigger_CMD(ENABLE,EXI_PIN4,_EXIFT);
    EXTI_interrupt_CMD(ENABLE,EXI_PIN4);
    GPIO_EXTI_interrupt(GPIOA0,0b00000000010000);
      //
      EXI3_Int_Enable();                                    //EXI4~EXI9 INT Vector
}
/*****************************************************************/
//BT Initial
//EntryParameter:NONE
//ReturnValue:NONE
/*****************************************************************/
```

```
void BT_CONFIG(void)
{
    BT_DeInit(BT0);
    BT_IO_Init(BT0_PB02);
    BT_Configure(BT0,BTCLK_EN,0,BT_IMMEDIATE,BT_CONTINUOUS,BT_PCLKDIV);//TCLK=PCLK/(0+1)
    BT_ControlSet_Configure(BT0,BT_START_HIGH,BT_IDLE_LOW,BT_SYNC_EN,BT_SYNCMD_EN,BT_OSTMDX_ONCE,BT_AREARM_DIS,BT_CNTRLD_EN);
    //BT_ControlSet_Configure(BT0,BT_START_HIGH,BT_IDLE_LOW,BT_SYNC_EN,BT_SYNCMD_DIS,BT_OSTMDX_ONCE,BT_AREARM_DIS,BT_CNTRLD_EN
);
    BT_Trigger_Configure(BT0,BT_TRGSRC_PEND,BT_TRGOE_DIS);
    BT_Period_CMP_Write(BT0,48000,24000);
    //BT_Start(BT0);
}
/*****************************************************************/
//ET Initial
//EntryParameter:NONE
//ReturnValue:NONE
/*****************************************************************/
void ET_CONFIG(void)
{
    ET_DeInit();
    //GPIO EXI 触发BT
    ET_CH0_SRCSEL(ET_SRC0,ENABLE,ET_EXI_SYNC0);
    ET_CH0_CONTROL(ENABLE,TRG_HW,ET_BT0_TRGSRC0);
    ET_ENABLE();
}
/*******************************************************************************/
//APT32F102_init
//EntryParameter:NONE
//ReturnValue:NONE
/*******************************************************************************/
void APT32F102_init(void)
{
//-----------------------------------------------------------------/
//Peripheral clock enable and disable
//EntryParameter:NONE
//ReturnValue:NONE
//-----------------------------------------------------------------/
    SYSCON->PCER0=0xFFFFFFF;                                    //PCLK Enable   0x00410071
    SYSCON->PCER1=0xFFFFFFF;                                    //PCLK Enable
    while(!(SYSCON->PCSR0&0x1));                                //Wait PCLK enabled
//-----------------------------------------------------------------/
//ISOSC/IMOSC/EMOSC/SYSCLK/IWDT/LVD/EM_CMFAIL/EM_CMRCV/CMD_ERR OSC stable interrupt
//EntryParameter:NONE
//ReturnValue:NONE
```

```
//-------------------------------------------------------------/
    SYSCON_CONFIG();                                    //syscon   initial
//------------ EVTRG function ---------------------------------/
   SYSCON->EVTRG=0X04|0x01<<20;                        //SYSCON_trgsrc0
   SYSCON->EVPS=0X00;
//
     CK_CPU_EnAllNormalIrq();                          //enable all IRQ
//-------------------------------------------------------------/
//Other IP config
//-------------------------------------------------------------/
   GPIO_CONFIG();                                      //GPIO initial
   BT_CONFIG();                                        //BT initial
   ET_CONFIG();
}

/*************************************************************************/
//main
/*************************************************************************/
int main(void)
{
   //delay_nms(5000);
   APT32F102_init();
     while(1)
   {
          SYSCON_IWDCNT_Reload();
     }
}
```

- **波形输出：**

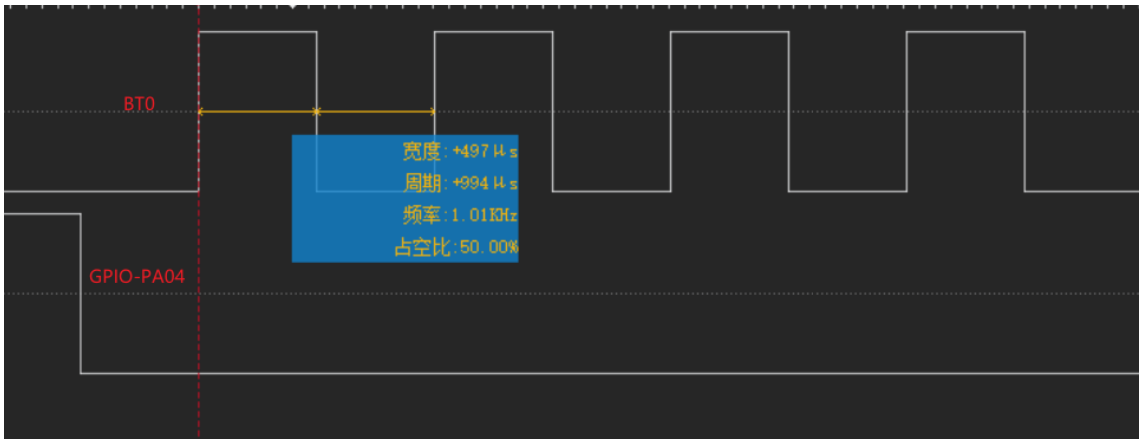


图 3.3.1 BT 输出波形

3.4 BT0 触发 LPT 输出

选择内部主频 48MHz 作为系统时钟，通过 BT0 输出 PEND 状态时触发 LPT 输出波形。

```
/*************************************************************/
// ETCB 测试     BT0 触发 LPT
/*************************************************************/
void ETCB_Init()
{
    BT_DeInit(BT0);                                                    //恢复默认值
    BT_IO_Init(BT0_PB02);                                              //设置 BT 输出口
    BT_Configure(BT0, BTCLK_EN, 0, BT_IMMEDIATE, BT_CONTINUOUS, BT_PCLKDIV); // TCLK = PCLK/(0+1)
    BT_ControlSet_Configure(
        BT0, BT_START_HIGH, BT_IDLE_LOW, BT_SYNC_DIS, BT_SYNCMD_DIS, BT_OSTMDX_ONCE, BT_AREARM_DIS, BT_CNTRLD_EN);
    BT_Trigger_Configure(BT0, BT_TRGSRC_PEND, BT_TRGOE_EN);            //BT_EVTRG  (事件触发控制寄存器)
    BT_Period_CMP_Write(BT0, 2000, 500);
    //
    BT_Start(BT0);                                                     //启动

    //
    LPT_DeInit();                                                      // LPT Deinit
    LPT_IO_Init(LPT_OUT_PB01);
  LPT_Configure(LPTCLK_EN, LPT_PCLK_DIV4, LPT_IMMEDIATE, LPT_PSC_DIV0, 0, LPT_OPM_CONTINUOUS);

  LPT_ControlSet_Configure(LPT_SWSYNDIS,  LPT_IDLE_Z,  LPT_PRDLD_DUTY_END,  LPT_POL_HIGH,  LPT_FLTDEB_00,  LPT_PSCLD_0,
    LPT_CMPLD_IMMEDIATELY);
LPT_SyncSet_Configure(LPT_TRGEN_EN, LPT_OSTMD_ONCE, LPT_AREARM_DIS);

    LPT_Period_CMP_Write(1000, 500);

    LPT_ConfigInterrupt_CMD(ENABLE, LPT_PEND);
    //
    // LPT_Start();
    //
    ET_DeInit();

    // BT0 触发 LPT
    ET_CH0_SRCSEL(ET_SRC0, ENABLE, ET_BT_SYNC0);                       //触发源使能
    ET_CH0_CONTROL(ENABLE, TRG_HW, ET_LPT_TRGSRC);
    ET_ENABLE();
}
/*************************************************************************************/
/*************************************************************************************/
```

```
// APT32F102_init                                                                    /
// EntryParameter:NONE                                                               /
// ReturnValue:NONE                                                                  /
/*******************************************************************************/
/*******************************************************************************/
/*******************************************************************************/
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/SYSCLK/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
    //-------------------------------------------------------------
    // initial
    ETCB_Init();
}
/*****************************************************************/
//main
/*****************************************************************/
int main(void)
{
    //delay_nms(5000);
    APT32F102_init();
      while(1)
    {
            SYSCON_IWDCNT_Reload();
            //
      }
}
```

- **输出波形：**

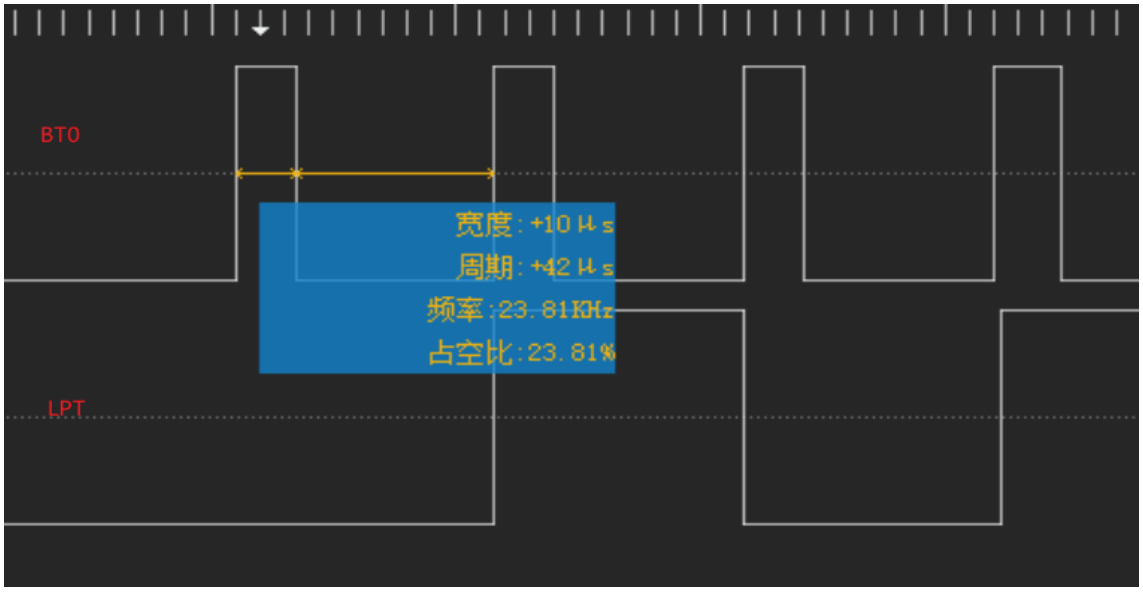


图 3.4.1 LPT 输出波形

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可验证如图 3.3.1 图 3.4.1 输出波形