BAT32G137（库函数版本）

Rev 1.0

修订历史

| 版本 | 日期 | 修订人 | 修订内容 |
|---|---|---|---|
| Rev1.1 | 22.6.27 | 缪勤文 张刚 | |
| | | | |

# 目录

# 1. 前言

BAT32G137 具有 1 个定时器单元 TimerA，以下定义为：TMA;TMA 定时单元有一个 16 位定时器。可以用作定时器、方波输出、外部输入脉冲宽度和周期的测量，以及外部事件计数。

# 2. 通用定时器 A 功能

以 BAT32G137 64PIN 为例，TMA 具有以下功能：

- 间隔定时器：能以固定间隔产生中断的基准定时器(递减计数)
- 方波输出：(递减计数)
- 外部事件计数器(递减计数)
- 外部脉冲宽度与周期测量

TMA 的运行时钟：FCLK、FCLK/2、FCLK/8、FIL、Fsub 或者 ENENTC 输入事件。

TMA 产生中断的条件有以下 3 种：

- 计数器发生下溢
- 脉冲测量模式：外部输入的有效宽度测量结束时
- 脉冲周期测量模式：检测到外部输入设置的有效边沿

# 3. 中微 BAT32G137 TMA 应用库简介

中微 BAT32G137 TMA 应用库是一个便于移植的标准库代码风格，用户只需要配置软件接口以及相关定时器参数进行配置、以及封装接口函数调用即可实现所需功能，节约时间，提高开发效率。应用库提供了以上功能的所有 demo code。

使用方式：

将应用层 tima_demo.c tima_demo.h 驱动层 tima.c tima.h、gpio.c gpio.h、isr.c isr.h 加入到工程中去;

## 3.1.应用例程使用

BAT32G137 64PIN 芯片引脚图


1、根据芯片引脚图：TMA 用作输出功能时数字功能为 TAO；作为输出或者输入时候数字功能为 TAIO；对于 P01，作为输出/输入功能是引脚的默认功能；对于 P00，可复用作为 TMA 输出脉冲波形的引脚；P01、P00、P50、P06、P31、P41 都可以作为 TMA 功能的引脚。

由上面分析：在使用 BAT32G137 TMA 时候需要注意引脚分配，需要查看各自的芯片引脚图。

### 3.1.1. 方波输出

```
1. void TMA_SquareOutput(TMA_Clk_t clk, uint16_t num)
2. {
3.  GPIO_InitTypeDef GPIO_InitStruct;
4.  TMA_InitTypeDef TIMA_InitStructure;
5.
6.  GPIO_PinAFConfig(GPIO_PORT0,GPIO_Pin_1,GPIO_P01,GROUP_AF_ODEFAULT);
     //P01 default used as TAIO output
```

```
7.
8.   GPIO_InitStruct.GPIO_Pin = GPIO_Pin_1;
9.   GPIO_InitStruct.GPIO_Mode = GPIO_Mode_OUT;
10.  GPIO_InitStruct.GPIO_Level = GPIO_Level_LOW;
11.  GPIO_InitStruct.GPIO_Ctrl = GPIO_Control_DIG;
12.  GPIO_Init(GPIO_PORT0,&GPIO_InitStruct);
13.
14.  TIMA_InitStructure.TMA_Clk = clk;   //specify the operation clk of
     tim
15.  TIMA_InitStructure.TMA_Period = num;                //specify the
     number of count clock
16.  TIMA_InitStructure.TMA_Mode = TMA_Mode_Square;      // Square mode
17.  TIMA_InitStructure.TMA_Polarity = TMA_Polarity_0;  //init level:(TI
     MA_Polarity_0/high level)(TIMA_Polarity_1/low level)
18.
19.  TMA_Init(&TIMA_InitStructure);
20.
21.  TMA_Start();
22. }
```

- 将 P01 用作 TAIO 功能，输出定时器 A 所产生的方波
- 配置 GPIO 为输出、数字功能、初始电平为低电平
- 选择 TMA 定时器的使用的运行时钟
- 选择方波功能
- 选择初始输出的电平为高电平
- 配置计数装载值，注意不能超过 65535

### 3.1.2. 外部事件计数

例程给出 2 种 TMA 外部事件计数功能：1、在 INTP4 引脚输出为高电平状态下，对外部输入事件的上升沿进行计数，计数到设置的数值时产生中断。2、对外部事件进行普通计数，即对外部输入事件的上升沿进行计数。

```
1.  /*****************************************************
2.  * Function Name: TMA_EventCount
3.  * @brief  The TMA operates as event counter.
4.  * @param  edge - specify the valid edge and count condition
5.  * @param  num - specify the count value
6.  * @return None
7.  *****************************************************/
8.  void TMA_EventCount_Init(uint16_t tma_edge, uint16_t tma_ctrol,uint1
    6_t num)
```

```
{
 GPIO_InitTypeDef GPIO_InitStruct={};
 TMA_InitTypeDef TIMA_InitStructure;

 GPIO_PinAFConfig(GPIO_PORT0,GPIO_Pin_1,GPIO_P01,GROUP_AF_ODEFAULT);
  //P01 default used as TAIO input event

 GPIO_InitStruct.GPIO_Pin = GPIO_Pin_1;
 GPIO_InitStruct.GPIO_Mode = GPIO_Mode_IN;
 GPIO_InitStruct.GPIO_Ctrl = GPIO_Control_DIG;
 GPIO_InitStruct.GPIO_PuPd = GPIO_PuPd_NOPULL;
 GPIO_Init(GPIO_PORT0,&GPIO_InitStruct);

 TIMA_InitStructure.TMA_Clk = TMA_FCLK_Div1;          //specify the o
peration clk of tim
 TIMA_InitStructure.TMA_Period  = num;                 //specify the
  number of count clock
 TIMA_InitStructure.TMA_Mode    = TMA_Mode_EventCount;      // Event
Count mode
 TIMA_InitStructure.TMA_EventPara.TIMA_Edge = tma_edge;
 TIMA_InitStructure.TMA_EventPara.TMA_Ctrl  = tma_ctrol;

 TMA_Init(&TIMA_InitStructure);

 ISR_Register(TMA_IRQn, tima_interrupt);  //中断服务路径注册
}
```

- 选择 P01 引脚作为外部输入事件的输入引脚，即使用 P01 的 TAIO 功能
- 配置 P01 的 GPIO 功能
- 选择 TMA 的运行时钟
- 设置外部事件计数值，由计数值到 0 时，产生中断
- 选择事件计数功能
- 选择外部事件的上升沿/下降沿或者双边沿进行计数
- 选择外部事件计数控制，如 INTP4 高/低电平下计数、定时器 M 单元输出波形高/低电平计数、定时器 TIM40 通道 2 通道 3 输出波形为高电平/低电平情况下计数、无控制条件下计数；

### 3.1.3.用作定时器

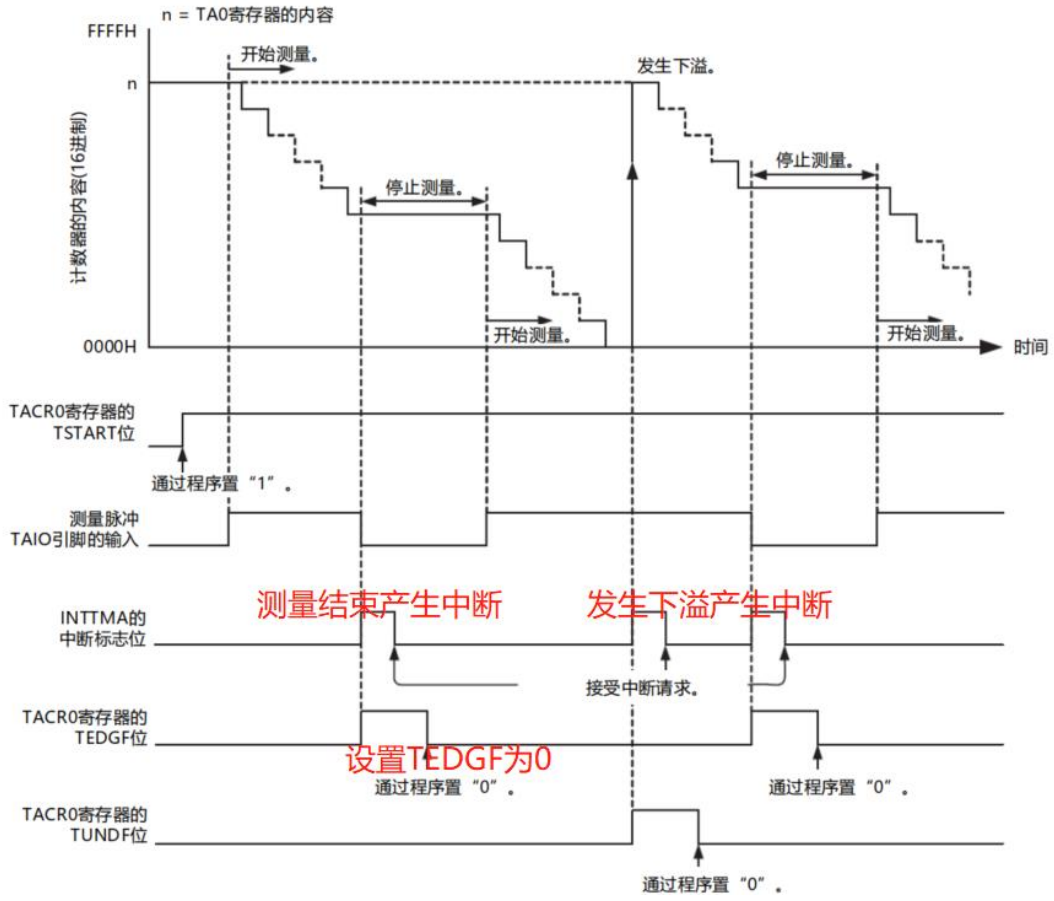
```
/*****************************************************************
* Function Name: TMA_IntervalTimer
* @brief This function interval timer and generate interval interrupt.
* @note   Interval timer period calculation:  1/clk *TIM_Period = timerValue (s)
* @return none
*****************************************************************/
void TMA_IntervalTimer(TMA_Clk_t clk, uint16_t num)
{
 TMA_InitTypeDef TIMA_InitStructure={};

 TIMA_InitStructure.TMA_Clk = clk;   //specify the operation clk of tim
 TIMA_InitStructure.TMA_Period = num;                //specify the number of count clock
 TIMA_InitStructure.TMA_Mode = TMA_Mode_Interval;      // timer mode

 TMA_Init(&TIMA_InitStructure);
 ISR_Register(TMA_IRQn, tima_timerinterrupt);  //中断服务路径注册
}
```

- 选择定时器运行时钟源
- 设置计数值
- 设置 TMA 工作在间隔定时器功能

  注意：定时时间:1/clk *countnum（单位 s）

### 3.1.4. 输入脉冲宽度测量

TMA 用作脉冲宽度测量时，将外部信号输入到具有 TAIO 功能的引脚，根据要求测量指定的电平，测量产生的时序如下示意图：




由图可知，在测量高电平结束时产生中断，需要设置 TEDGF 位为 0。同时在溢出时也会产生中断，这时候计算高电平测量时，需要使用上一次存储的定时器 TMA->TA0 值。

脉冲测量初始化

```
18.void TMA_Get_PulseWidth(uint16_t clk, TMA_Pulse_t level)
19.{
20.  uint32_t width;
21.  GPIO_InitTypeDef GPIO_InitStruct;
22.  TMA_InitTypeDef TIMA_InitStructure;
23.  ISR_InitTypeDef_t ISR_InitStructure;
24.
25.  GPIO_PinAFConfig(GPIO_PORT0,GPIO_Pin_1,GPIO_P01,GROUP_AF_ODEFAULT);
      //P01 used as TAIO input
26.
27.  GPIO_InitStruct.GPIO_Pin = GPIO_Pin_1;
28.  GPIO_InitStruct.GPIO_Mode = GPIO_Mode_IN;
29.  GPIO_InitStruct.GPIO_Ctrl = GPIO_Control_DIG;
30.  GPIO_InitStruct.GPIO_PuPd = GPIO_PuPd_NOPULL;
31.  GPIO_Init(GPIO_PORT0,&GPIO_InitStruct);
32.
33.  TIMA_InitStructure.TMA_Clk = clk;    //specify the operation clk of
    tim
```

```
34. TIMA_InitStructure.TMA_Period   = g_tma0_ta0_value;       //specify
    the number of count clock
35. TIMA_InitStructure.TMA_Mode     = TMA_Mode_PluseWidth;      // Event
    Count mode
36. TIMA_InitStructure.TMA_Pulse    = level;
37. TMA_Init(&TIMA_InitStructure);
38.}
```

- 选择 P01 引脚作为外部输入事件的输入引脚，即使用 P01 的 TAIO 功能
- 选择定时器运行时钟源
- 设置计数值默认给 0xffff
- 设置 TMA 工作在脉宽测量功能

脉冲测量函数

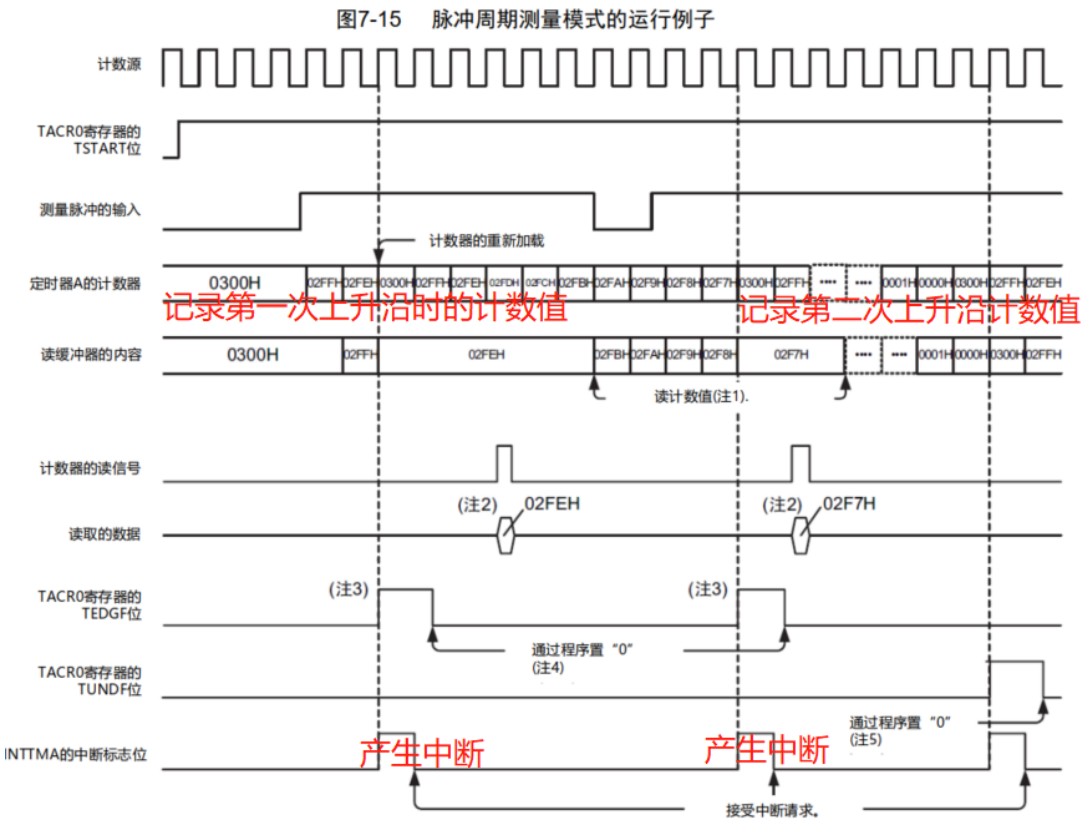
```
1.  uint32_t TMA_GetPulseWidth(void)
2.  {
3.   uint32_t width, period;
4.   static uint32_t tempwidth = 0xffff,tma_underflow_count=0;
5.
6.   while(INTC_GetPendingIRQ(TMA_IRQn) == 0);
7.   INTC_ClearPendingIRQ(TMA_IRQn); /* clear INTTMA interrupt flag */
8.
9.   if ((TMA->TACR0 & _20_TMA_UNDERFLOW_OCCUR) != 0U)
10.  {
11.   TMA->TACR0 &= (uint8_t)~_20_TMA_UNDERFLOW_OCCUR;
12.   tma_underflow_count += 1U;
13.  }
14.  if(TMA->TACR0 & _10_TMA_ACTIVE_EDGE_RECEIVED) //active edge ocuurs
15.  {
16.   TMA->TACR0 &= ~_10_TMA_ACTIVE_EDGE_RECEIVED;
17.   if (tma_underflow_count == 0U)
18.   {
19.    width = tempwidth - TMA->TA0;
20.    tempwidth = TMA->TA0;
21.   }
22.   else
23.   {
24.    width = tempwidth + (0xffff + 1) * 1 - TMA->TA0;
25.    tempwidth = TMA->TA0;
26.    tma_underflow_count = 0;
27.   }
28.  }
29.  return width;
```

```
30. }
```

- 产生中断时候，判断中断类型：高电平/低电平测量结束产生的中断，还是计数器下溢中断
- 在测量结束中断产生之后，将 TEDGF 位置 0，求出测量电平宽度；同时记录当前计数器值；
- 若产生下溢中断，需要结合上一次计数器值，求出电平宽度

### 3.1.5. 脉冲周期测量

TMA 的脉冲测量原理如下图：




TMA 在用作周期测量模式时，在第一次上升沿时候，产生中断，此时将设置计数值加载到计数器，图中将 0x2fff 加载到计数器；当第二次上升沿产生中断时，此时周期测量结束，需要用第一次减去第二次计数值。在第二次产生中断之后，计数器的计数值默认重载为初始化的设置值。

周期测量初始化函数

```
1. void TMA_Get_PulsePeriod(uint16_t clk, TMA_Pulse_t edge)
2. {
3.  uint32_t width;
4.     volatile uint16_t tmp;
5.  GPIO_InitTypeDef GPIO_InitStruct;
6.  TMA_InitTypeDef TIMA_InitStructure;
```

```
7.  ISR_InitTypeDef_t ISR_InitStructure;
8.
9.  GPIO_PinAFConfig(GPIO_PORT0,GPIO_Pin_1,GPIO_P01,GROUP_AF_ODEFAULT);
     //P00 used as TAIO input
10.
11. GPIO_InitStruct.GPIO_Pin = GPIO_Pin_1;
12. GPIO_InitStruct.GPIO_Mode = GPIO_Mode_IN;
13. GPIO_InitStruct.GPIO_Ctrl = GPIO_Control_DIG;
14. GPIO_InitStruct.GPIO_PuPd = GPIO_PuPd_NOPULL;
15. GPIO_Init(GPIO_PORT0,&GPIO_InitStruct);
16.
17. TIMA_InitStructure.TMA_Clk = clk;   //specify the operation clk of
    tim
18. TIMA_InitStructure.TMA_Period  = g_tma0_ta0_value;       //specify
    the number of count clock
19. TIMA_InitStructure.TMA_Mode    = TMA_Mode_PlusePeriod;      // Even
    tCount mode
20. TIMA_InitStructure.TMA_Pulse   = edge;
21. TMA_Init(&TIMA_InitStructure);
22.
23.}
```

- 选择 P01 引脚作为外部输入事件的输入引脚，即使用 P01 的 TAIO 功能
- 选择定时器运行时钟源
- 设置计数值默认给 0xffff
- 设置 TMA 工作在周期测量功能

脉冲周期测量函数

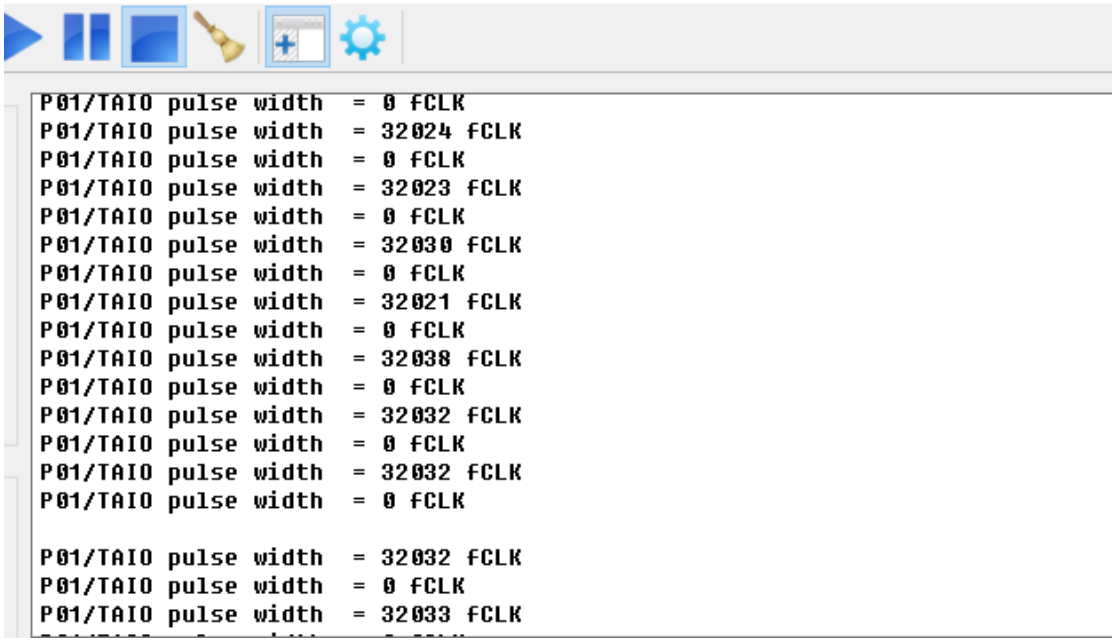
```
1.  uint32_t TMA_GetPulsePeriod(void)
2.  {
3.   uint32_t width=0, period;
4.   static uint32_t tempwidth = 0xffff,tma_underflow_count=0;
5.   static uint8_t time =0;
6.
7.   while(INTC_GetPendingIRQ(TMA_IRQn) == 0);
8.      INTC_ClearPendingIRQ(TMA_IRQn); /* clear INTTMA interrupt flag *
    /
9.
10.     if ((TMA->TACR0 & _20_TMA_UNDERFLOW_OCCUR) != 0U)
11.     {
12.         TMA->TACR0 &= (uint8_t)~_20_TMA_UNDERFLOW_OCCUR;
13.         tma_underflow_count += 1U;
14.     }
```

```
15.  if((TMA->TACR0 & _10_TMA_ACTIVE_EDGE_RECEIVED)) //no active edge oc
    uurs
16.  {
17.   time++;
18.   TMA->TACR0 &= ~_10_TMA_ACTIVE_EDGE_RECEIVED;
19.
20.   if(time == 1)
21.   {
22.    period = TMA->TA0;    // dummy read to update readbuf
23.   }
24.   else
25.   {
26.    if(tma_underflow_count == 0)
27.    {
28.     width = tempwidth - TMA->TA0;
29.    }
30.    else
31.    {
32.      width = tempwidth - TMA->TA0 + 1U + (tempwidth + 1) * tma_under
    flow_count;
33.    }
34.
35.    time = 0;
36.   }
37.  }
38.
39.  return width;
40. }
```

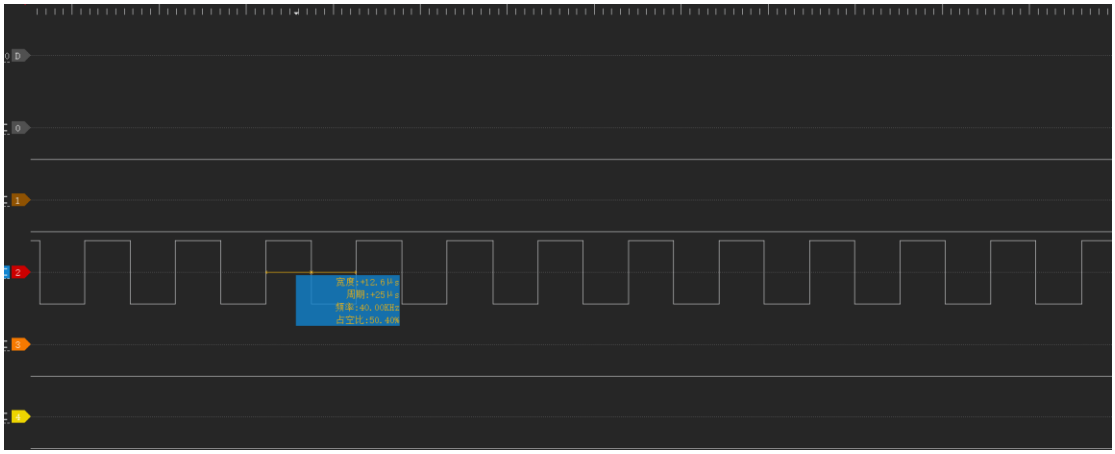
- 产生中断时候，判断中断类型：上升沿/下降沿产生的中断，还是计数器下溢中断
- 检测到有效边沿中断产生之后，判断是第一次还是第二次边沿中断，第二次中断时求出脉冲周期宽度
- 若产生下溢中断，需要结合上一次计数器值，求出周期宽度

# 4. 示例演示

外部产生周期为 2ms 的方波，输入到 P01 引脚，测试结果：

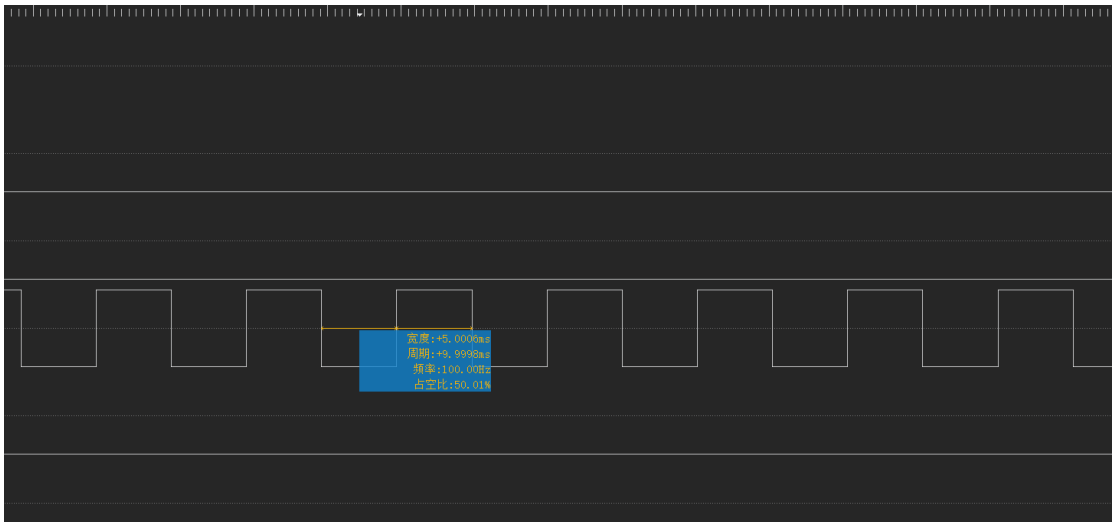



图一 测量脉冲周期结果



图二 产生方波

定时功能，通过在中断中拉高拉低 P31 引脚，观察 P31 状态确定产生 5ms 的定时中断



图三 产生 5ms 的定时中断

普通事件计数，P17 输出波形，输入 P01 引脚，P01 对 P17 上升沿进行计数，计数值达到 10 产生中断，g_tma0_underflow_count 值增加 1