本章教程通过CH32V103开发板PA8引脚检测PWM脉宽和周期，并通过串口调试助手打印显示。

1、输入捕获简介及相关函数介绍

输入捕获模式是定时器的基本功能之一，其通常用于测量频率或者测量脉宽。

输入捕获可对输入信号的上升沿、下降沿或者双边沿进行捕获，其捕获原理为：当发生并捕获信号跳变沿之后，计数器（CNT）值将被锁存到捕获比较寄存器（CCR）中，将前后两次捕获到的CCR寄存器中的值相减，即可计算出频率或者脉宽。如果捕获脉宽时长超过捕获定时器的周期，会发生溢出，此时需要进行额外处理。

关于定时器输入捕获具体介绍，可参考CH32V103应用手册。输入捕获程序所需函数所在库函数在定时器中断教程中已介绍，本章不做过多介绍。

2、硬件设计

本章教程使用输入捕获中PWM输入模式，需要用到两个开发板，一个用于产生PWM输出，一个用于输入捕获。本章教程使用两个CH32V103开发板，一个直接下载CH32V103 EVT中PWM输出例程，一个直接下载本章教程中例程，然后将两个开发板PA8引脚连接起来。

3、软件设计

本章教程使用PWM输入模式进行输入捕获，其使用两个通道和两个捕获寄存器，具体实现方式为：当使用PWM输入模式时，PWM信号由输入通道TI1进入，配置滤波后的定时器输入1（TI1FP1）为触发信号并设置上升沿捕获。当上升沿的时候捕获比较通道IC1和IC2同时捕获，计数器CNT清零，到了下降沿的时候，IC2捕获，此时计数器CNT的值被锁存到捕获比较寄存器CCR2中，到了下一个上升沿的时候，IC1捕获，计数器CNT 的值被锁存到捕获比较寄存器CCR1中。其中CCR2+1测量的是脉宽， CCR1+1测量的是周期。这里要注意的是

CCR2 和 CCR1 的值在计算占空比和频率的时候都必须加1，因为计数器是从0开始计数的。
使用PWM输入模式实现输入捕获具体程序如下：
capture.h文件

```
#ifndef __CAPTURE_H
#define __CAPTURE_H

#include "ch32v10x_conf.h"

void Input_Capture_Init( u16 arr, u16 psc );
void TIM1_CC_IRQHandler(void);

#endif

```

复制代码
capture.h文件主要是函数的声明。
capture.c文件

```
#include "capture.h"

void TIM1_CC_IRQHandler(void) __attribute__((interrupt("WCH-Interrupt-fast")));

/*********************************************************************************
* Function Name  : Input_Capture_Init
* Description    : Initializes TIM1 input capture.
* Input          : arr: the period value.
*                  psc: the prescaler value.
*                  ccp: the pulse value.
* Return         : None
*********************************************************************************/
void Input_Capture_Init( u16 arr, u16 psc )
{
    GPIO_InitTypeDef GPIO_InitStructure;
    TIM_TimeBaseInitTypeDef TIM_TimeBaseInitStructure;
    TIM_ICInitTypeDef TIM_ICInitStructure;
    NVIC_InitTypeDef NVIC_InitStructure;

    RCC_APB2PeriphClockCmd( RCC_APB2Periph_GPIOA | RCC_APB2Periph_TIM1, ENABLE );//使能GPIOA时钟和TIM1时钟

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
    GPIO_Init( GPIOA, &GPIO_InitStructure);
    GPIO_ResetBits( GPIOA, GPIO_Pin_8 );

```

```
27.     TIM_TimeBaseInitStructure.TIM_Period = arr;                //设置重
    装载值
28.     TIM_TimeBaseInitStructure.TIM_Prescaler = psc;             //设置预
    分频器值
29.     TIM_TimeBaseInitStructure.TIM_ClockDivision = TIM_CKD_DIV1;
    //设置时钟分频因子
30.     TIM_TimeBaseInitStructure.TIM_CounterMode =
    TIM_CounterMode_Up; //设置计数模式，向上计数模式
31.     TIM_TimeBaseInitStructure.TIM_RepetitionCounter =  0x00;
      //设置重复计数器的值，0
32.     TIM_TimeBaseInit( TIM1, &TIM_TimeBaseInitStructure); //初始化
33.
34.     TIM_ICInitStructure.TIM_Channel = TIM_Channel_1;     //配置
    TIM1捕获通道
35.     TIM_ICInitStructure.TIM_ICPrescaler = TIM_ICPSC_DIV1;//配置输
    入捕获预分频器值
36.     TIM_ICInitStructure.TIM_ICFilter = 0x00;             //配置输入捕获筛
    选器，设为0
37.     TIM_ICInitStructure.TIM_ICPolarity = TIM_ICPolarity_Rising; //上
    升沿捕获
38.     TIM_ICInitStructure.TIM_ICSelection = TIM_ICSelection_DirectTI;
    //配置指定输入TI1
39.
40.     TIM_PWMIConfig( TIM1, &TIM_ICInitStructure );
41.
42.     NVIC_InitStructure.NVIC_IRQChannel = TIM1_CC_IRQn;
    //TIM1捕获比较中断
43.     NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 2;//设
    置抢占优先级
44.     NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;       //设置响
    应优先级
45.     NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;          //使能
    通道
46.     NVIC_Init(&NVIC_InitStructure);
47.
48.     TIM_ITConfig( TIM1, TIM_IT_CC1 | TIM_IT_CC2, ENABLE ); //使能
    TIM1捕获中断
49.
50.     TIM_SelectInputTrigger( TIM1, TIM_TS_TI1FP1 );    //选择过滤定时
    器输入1作为触发源
51.     TIM_SelectSlaveMode( TIM1, TIM_SlaveMode_Reset ); //TIM1从
    机模式，所选触发信号（TRGI）的上升沿重新初始化。
52.     TIM_SelectMasterSlaveMode( TIM1,
    TIM_MasterSlaveMode_Enable ); //设置或重置TIM1主从模式，使能定
    时器与从机之间同步
53.     TIM_Cmd( TIM1, ENABLE );  //定时器使能
54. }
55.
56. /*************************************************************************
```

```
57. * Function Name  : TIM1_CC_IRQHandler
58. * Description    : This function handles TIM1  Capture Compare
    Interrupt exception.
59. * Input          : None
60. * Output         : None
61. * Return         : None
62. *******************************************************************************/
63. void TIM1_CC_IRQHandler(void)
64. {
65.     if( TIM_GetITStatus( TIM1, TIM_IT_CC1 ) != RESET )  //若捕获比较
    1发生中断
66.     {
67.         printf( "cycle:%d\r\n", (TIM_GetCapture1(TIM1)+1) ); //打印得
    到的捕获比较1寄存器值，其值加1表示周期
68.     }
69.     if( TIM_GetITStatus( TIM1, TIM_IT_CC2 ) != RESET )  //若捕获比较
    2发生中断
70.     {
71.         printf( "Pulsewidth:%d\r\n", (TIM_GetCapture2(TIM1)+1) ); //打
    印得到的捕获比较2寄存器值，其值加1表示脉宽
72.     }
73.     TIM_ClearITPendingBit( TIM1, TIM_IT_CC1 | TIM_IT_CC2 ); //清除
    TIM1捕获比较1和捕获比较2中断挂起位
74. }
75.
76.
```

复制代码

capture.c文件主要是输入捕获相关函数配置，包括输入捕获相关函数初始化配置和定时器中断服务函数，其具体配置流程如下：

1、使能GPIOA时钟和TIM1时钟

```
1. RCC_APB2PeriphClockCmd( RCC_APB2Periph_GPIOA |
   RCC_APB2Periph_TIM1, ENABLE );//使能GPIOA时钟和TIM1时钟
```

复制代码

2、GPIO初始化配置

```
1. GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8;
2. GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
3. GPIO_Init( GPIOA, &GPIO_InitStructure);
4. GPIO_ResetBits( GPIOA, GPIO_Pin_8 );
```

复制代码

3、根据TIM_TimeBaseInitStruct中指定的参数初始化TIM1时基单元外围设备。

```
1. TIM_TimeBaseInitStructure.TIM_Period = arr;              //设置重装载
   值
```

2. TIM_TimeBaseInitStructure.TIM_Prescaler = psc;　　//设置预分频器值
3. TIM_TimeBaseInitStructure.TIM_ClockDivision = TIM_CKD_DIV1; //设置时钟分频因子
4. TIM_TimeBaseInitStructure.TIM_CounterMode = TIM_CounterMode_Up; //设置计数模式，向上计数模式
5. TIM_TimeBaseInitStructure.TIM_RepetitionCounter = 0x00;　　//设置重复计数器的值，0
6. TIM_TimeBaseInit(TIM1, &TIM_TimeBaseInitStructure); //初始化

复制代码

4、输入捕获结构体初始化，配置IC1捕获通道

1. TIM_ICInitStructure.TIM_Channel = TIM_Channel_1;　//配置TIM1捕获通道
2. TIM_ICInitStructure.TIM_ICPrescaler = TIM_ICPSC_DIV1;//配置输入捕获预分频器值
3. TIM_ICInitStructure.TIM_ICFilter = 0x00;　　//配置输入捕获筛选器，设为0
4. TIM_ICInitStructure.TIM_ICPolarity = TIM_ICPolarity_Rising; //上升沿捕获
5. TIM_ICInitStructure.TIM_ICSelection = TIM_ICSelection_DirectTI; //配置指定输入TI1

复制代码

5、初始化PWM输入模式

1. TIM_PWMIConfig(TIM1, &TIM_ICInitStructure);

复制代码

6、配置中断优先级

1. NVIC_InitStructure.NVIC_IRQChannel = TIM1_CC_IRQn;　　//TIM1捕获比较中断
2. NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 2; //设置抢占优先级
3. NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;　　//设置响应优先级
4. NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;　　//使能通道
5. NVIC_Init(&NVIC_InitStructure); //NVIC初始化

复制代码

7、使能捕获中断

1. TIM_ITConfig(TIM1, TIM_IT_CC1 | TIM_IT_CC2, ENABLE); //使能TIM1捕获中断

复制代码

8、选择输入捕获触发信号以及主从模式

```
1. TIM_SelectInputTrigger( TIM1, TIM_TS_TI1FP1 );                    //
   选择过滤定时器输入1作为触发源
2. TIM_SelectSlaveMode( TIM1, TIM_SlaveMode_Reset );
   //TIM1从机模式，所选触发信号（TRGI）的上升沿重新初始化。
3. TIM_SelectMasterSlaveMode( TIM1, TIM_MasterSlaveMode_Enable
   );//设置或重置TIM1主从模式，使能定时器与从机之间同步
```

复制代码

9、使能定时器

```
1. TIM_Cmd( TIM1, ENABLE );
```

复制代码

10、编写中断服务函数

```
1. void TIM1_CC_IRQHandler(void)
2. {
3.     if( TIM_GetITStatus( TIM1, TIM_IT_CC1 ) != RESET )  //若捕获比较
   1发生中断
4.     {
5.         printf( "cycle:%d\r\n", (TIM_GetCapture1(TIM1)+1) ); //打印得
   到的捕获比较1寄存器值，其值加1表示周期
6.     }
7.     if( TIM_GetITStatus( TIM1, TIM_IT_CC2 ) != RESET )  //若捕获比较
   2发生中断
8.     {
9.         printf( "Pulsewidth:%d\r\n", (TIM_GetCapture2(TIM1)+1) ); //打
   印得到的捕获比较2寄存器值，其值加1表示脉宽
10.    }
11.    TIM_ClearITPendingBit( TIM1, TIM_IT_CC1 | TIM_IT_CC2 ); //清除
   TIM1捕获比较1和捕获比较2中断挂起位
12. }
```

复制代码

capture.c文件主要是进行输入捕获的相关配置以及中断服务函数功能的设置，通过以上配置，即可进行输入捕获并通过中断服务函数打印输出脉宽和周期。

main.c文件

```
1. int main(void)
2. {
3.     USART_Printf_Init(115200);
4.     printf("SystemClk:%d\r\n",SystemCoreClock);
5.
6.     Input_Capture_Init( 0xFFFF, 48000-1 );
7.
8.     while(1);
9. }
```

复制代码

main.c文件主要是相关函数的初始化。

4、下载验证

将编译好的程序下载到开发板并复位，将两个开发板PA8引脚连接起来，打开串口调试助手，可看见在不停打印PWM脉宽和周期，具体如下：