ECAP 使用指南

概述

增强型捕获（ECAP）模块用于需要准确计时外部事件的情况，以及对外部输入的信号的占空比以及周期信息进行解码。

本手册使用范围：

适用范围	
SPC1125 系列	SPC1125，SPC1128
SPC1168 系列	SPC1155， SPC1156， SPC1158， SPC1168，SPD1148， SPD1178， SPD1188， SPD1163，SPM1173
SPC2168 系列	SPC2168，SPC2165，SPC2166，SPC1198
SPC1169 系列	SPC1169，SPD1179，SPD1176
SPC2188 系列	SPC1185，SPC2188

目录

图片列表

表格列表

版本历史

版本	日期	作者	状态	变更
A/0	2023-09-01	X.He	Outdated	1. 首次发布。
C/0	2024-08-02	Lm.Zhou	Released	1. 修改为全系列通用文档。

术语或缩写

术语或缩写	描述
MCU	Microcontroller Unit，微控制器单元
ECAP	Enhanced capture，增强型捕获单元

1 特性

增强型捕获模块（ECAP）用于对外部输入事件的时序关系进行计时，应用场景包括：

- 测量旋转机械的转速；
- 测量脉冲信号的周期及占空比；
- 对于占空比编码系统的解码；

ECAP 单元有以下模块特点：

- 基于 32bit 的计时器（时间基准），最高时钟频率同 CPU。
- 灵活的输入捕获引脚：任意 GPIO 均可配置为捕获引脚；
- 4 个时间标签捕获寄存器，支持绝对时间戳捕获和相对时间戳捕获；
- 4 个事件均可独立选择边沿极性（上升/下降沿）；
- 与外部事件同步的 4 级序列器；
- 4 个事件均可支持中断：
  - 单轮捕获模式（即一次性捕获模式）：在依次完成最多 4 次捕获后停止；
  - 持续捕获模式：在最多 4 个捕获事件之间持续轮流执行；
- 可被配置为辅助 PWM（APWM）模式，产生简单的 PWM 波形；
- 对于上述资源，部分型号 MCU 仅用于单个输入引脚，部分型号 MCU 即可用于单个输入引脚，可用于双输入引脚，如下表 1-1 所示：

表 1-1：MCU 的 ECAP 模块是否支持双引脚输入

ECAP 支持单引脚输入的产品型号	ECAP 支持双引脚输入的产品型号
SPC1125 系列，SPC1168 系列，SPC2168 系列	SPC1169 系列，SPC2188 系列
SPC1169 系列，SPC2188 系列	

2 功能描述

ECAP 单元可连续捕获 4 个事件的时间戳信息，具备两种模式，捕获模式和 APWM 模式。

捕获模式，即捕获某个事件，并记录事件发生时的时间戳。

在捕获模式下 ECAP 根据是否会循环更新 CAPx 寄存器的数值其捕获事件的行为又可分为连续捕获和单次捕获。

- 连续捕获：计数器从 0 开始连续递增计数，当事件触发时，会将计数器当成时间戳按事件次序依次锁存到 CAPx，且新的对应事件的时间戳会覆盖旧的 CAPx 数值。连续模式又可根据是否在每个事件发生时复位计数值。
- 单次捕获：计数器从 0 开始连续递增计数，当事件触发时，会将计数器当成时间戳按事件次序依次锁存到 CAPx，但当事件发生的个数达到预设数值时，将停止计数，并且也不会进一步更新 CAPx 的数值。

在捕获模式下，ECAP 记录事件发生时机的时间戳计数器，其计数方式又分为绝对时间戳计数和相对时间戳计数。

- 绝对时间戳计数：计数器从 0 开始连续递增计数，当最后事件触发时，计数器将会复位，从 0 开始计数。
- 相对时间戳计数：计数器从 0 开始连续递增计数，当有事件触发时，计数器将会复位，从 0 开始计数。

计数器的这两种计数方式，在连续捕获和单次捕获中均可使用。

对于表 1-1 中支持 ECAP 双引脚输入的 MCU，具备双通道模式。此模式下，可捕获单通道信号也可捕获双通道信号。捕获双通道信号时，CAP0 和 CAP2 作为捕获一个输入信号边沿事件的一对寄存器，CAP1 和 CAP3 作为另一个输入信号边沿事件的一对寄存器，捕获单通道信号时 CAP0、CAP1、CAP2、CAP3 相互之间没有成对的关系。

APWM 模式，即辅助 PWM 模式。作为 APWM 模式时，ECAP 可以当作 PWM 功能进行输出波形，CAP0 作为 APWM0 的周期寄存器，CAP1 作为 APWM 的比较寄存器，CAP2 和 CAP3 作为 CAP0 和 CAP1 的影子寄存器，在 CNT=PRD 时会更新 CAP0 和 CAP1。

3 功能实例

3.1 单通道连续捕获

3.1.1 实例 1：通过绝对时间计算 PWM 频率

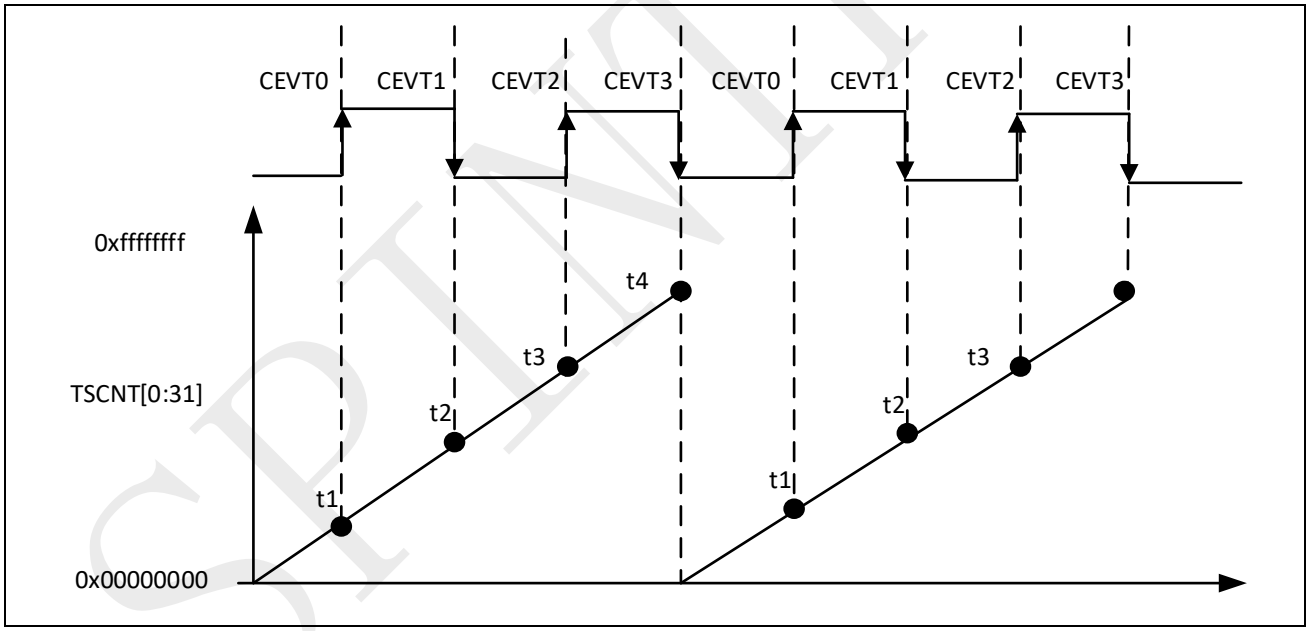
3.1.1.1 功能需求

使用连续捕获模式，以绝对时间戳计数方式，对 PWM 进行频率计算。

3.1.1.2 功能实现

1. 确定捕获信息：以绝对时间戳方式计数，对 PWM 波进行连续捕获。可获取图 3-1 所示信息，包括：
   a) 波形的周期 T= t3 – t1 或 T= t4– t2；
   b) 波形高占比时间 Th = t2 – t1 或 Th = t4 – t3；
   c) 波形为低占时间 TL = t3-t2；

图 3-1：连续捕获（绝对时间戳计数）



- 产生待计算频率的 PWM 波：

  初始化 PWM 时钟；

  配置 PWM 外设，产生单通道 PWM 输出；

- 配置 ECAP 功能：

  初始化 ECAP 为捕获模式，初始化函数默认捕获事件为 CAP0 上升沿捕获、CAP1 下降沿捕获、CAP2 上升沿捕获、CAP3 下降沿捕获；

- 设置被捕获的引脚为输入功能，将被捕获的引脚的 PWM 信号送入 ECAP；

- 使能被捕获引脚的滤波功能并设置滤波窗口，减少干扰带来的误差；
  a) 使能 ECAP_CEVT3，当事件 3（CAP3 捕获下降沿）发生，ECAP 的计数器将进行复位，重新开始计数；
- 使能 EVT3 事件中断，当事件 3 发生将会产生中断，然后在中断服务函数中通过时间戳寄存器 CAPx 进行计算波形的频率；
- 频率计算公式为：

$$\frac{ECAP\_CLK}{((t2-t0)+(t3-t1))/2}$$

以上实现步骤的示例代码可参考 SDK 提供的 Demo，如表 3-1。

表 3-1：实例 1 代码路径

MCU 产品型号	代码路径
所有型号	SDK 目录\0_Examples\ECAP_Continue_Absolute

3.1.2 实例 2：通过相对时间计算 PWM 频率

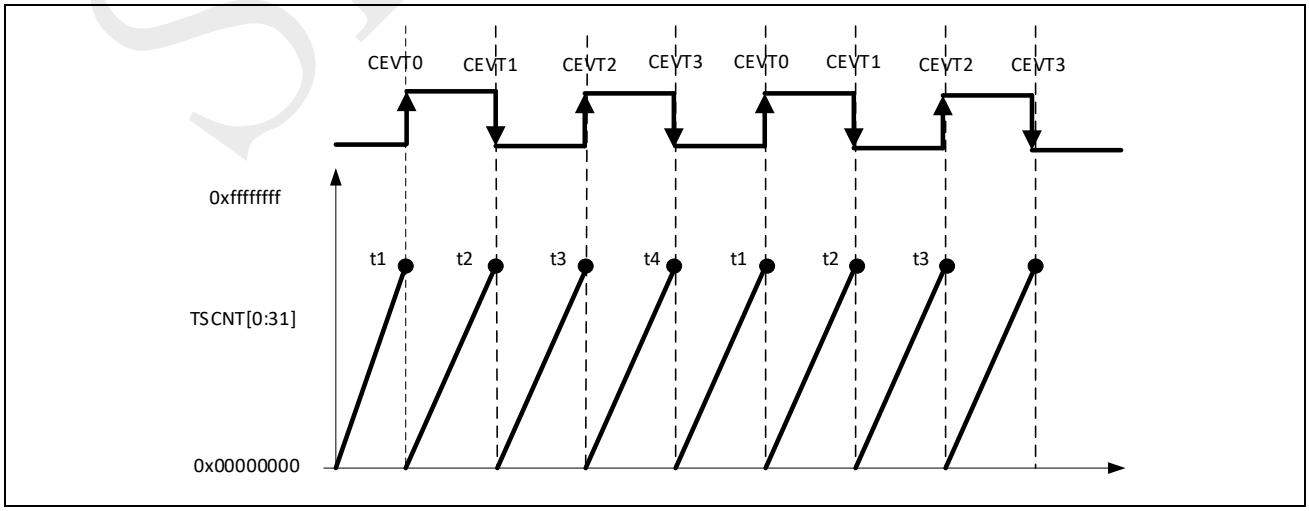
3.1.2.1 功能需求

使用连续捕获模式，以相对时间戳计数方式，对 PWM 进行频率计算。

3.1.2.2 功能实现

1. 确定捕获信息：以相对时间戳方式计数，对 PWM 波进行连续捕获。可获取图 3-2 所示信息，包括：
   a) 波形的周期 T=t1（或 t3）+t2（或 t4）；
   b) 波形高占比时间 T_H = t2 或 T_H = t4；
   c) 波形为低占时间 T_L = t1 或 T_L = t3;

图 3-2：连续捕获（相对时间戳计数）



- 产生待计算频率的 PWM 波；
- 初始化 PWM 时钟；
- 配置 PWM 外设，产生单通道 PWM 输出；
- 配置 ECAP 功能：
  a) 初始化 ECAP 为捕获模式，初始化函数默认捕获事件为 CAP0 上升沿捕获、CAP1 下降沿捕获、CAP2 上升沿捕获、CAP3 下降沿捕获；
  b) 设置被捕获的引脚为输入功能，将被捕获的引脚的 PWM 信号送入 ECAP；
  使能被捕获引脚的滤波功能并设置滤波窗口，减少干扰带来的误差；
  c) 使能 ECAP_CEVT0~3，当事件发生，ECAP 的计数器都进行复位，重新开始计数；
  d) 使能 EVT3 事件中断，当事件发生将会产生中断，然后在中断服务函数中通过时间戳寄存器 CAPx 进行计算波形的频率；
- 频率计算公式为：

$$\frac{ECAP\_CLK}{[(t1+t2)+(t2+t3)]/2}$$

以上实现步骤的示例代码可参考 SDK 提供的 Demo，如表 3-2。

表 3-2：实例 2 代码路径

MCU 产品型号	代码路径
所有型号	SDK 目录\0_Examples\ECAP_Countinue_Delta

3.2 单通道单次捕获

3.2.1 实例 3：通过绝对时间计算 PWM 频率

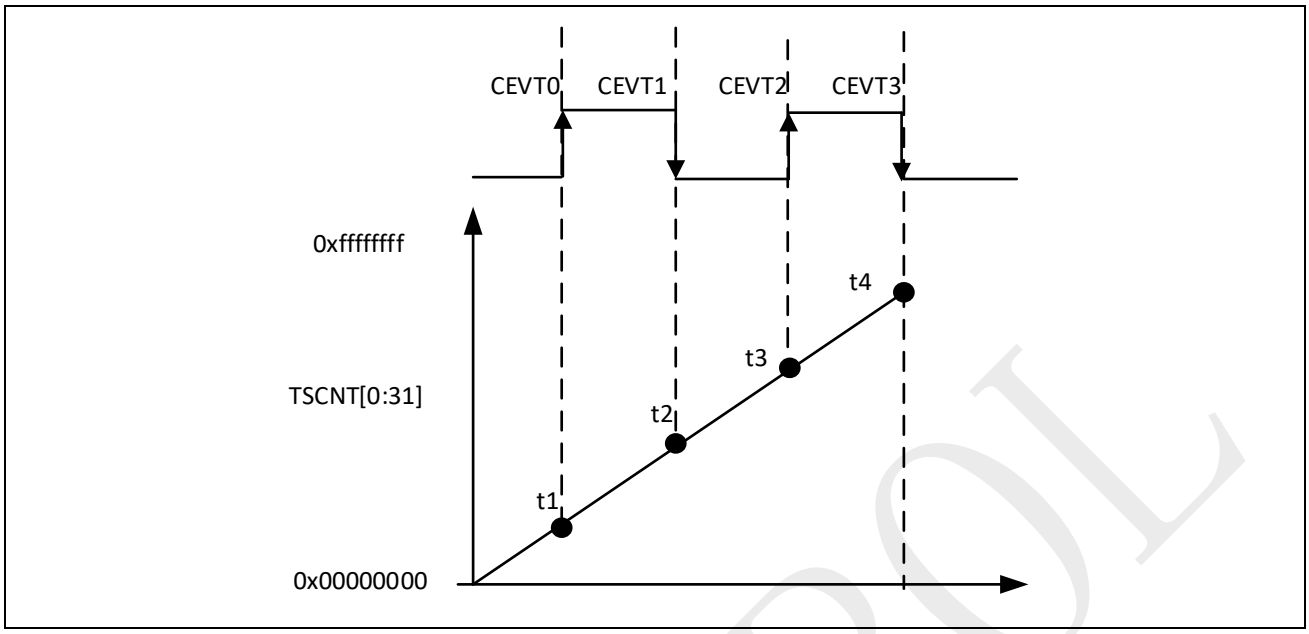
3.2.1.1 功能需求

使用单次捕获模式，以绝对时间戳计数方式，对 PWM 进行频率计算。

3.2.1.2 功能实现

1. 确定捕获信息：以相对时间戳方式计数，对 PWM 波进行单次捕获。可获取图 3-3 所示信息，包括：
   a) 波形的周期 T= t3 – t1 或 T= t4– t2；
   b) 波形高占比时间 T_H = t2 – t1 或 T_H = t4 – t3；
   c) 波形为低占时间 T_L = t3-t2；

图 3-3：单次捕获（绝对时间戳计数）



- 产生待计算频率的 PWM 波：
- 初始化 PWM 时钟；
- 配置 PWM 外设，产生单通道 PWM 输出；
- 配置 ECAP 功能：
- 初始化 ECAP 为捕获模式，初始化函数默认捕获事件为 CAP0 上升沿捕获、CAP1 下降沿捕获、CAP2 上升沿捕获、CAP3 下降沿捕获；
- 配置 ECAP 为单次模式；
- 设置被捕获的引脚为输入功能，将被捕获的引脚的 PWM 信号送入 ECAP；
- 使能被捕获引脚的滤波功能并设置滤波窗口，减少干扰带来的误差；

  a) 使能 ECAP_CEVT3，当事件 3（CAP3 捕获下降沿）发生，ECAP 的计数器将进行复位，重新开始计数；
- 使能 EVT3 事件中断，当事件 3 发生将会产生中断，然后在中断服务函数中通过时间戳寄存器 CAPx 进行计算波形的频率；
- 频率计算公式为：

$$\frac{ECAP\_CLK}{[(t2-t0)+(t3-t1)]/2}$$

以上实现步骤的示例代码可参考 SDK 提供的 Demo，如表 3-3。

表 3-3：实例 3 代码路径

MCU 产品型号	代码路径
所有型号	SDK 目录\0_Examples\ECAP_Oneshot_Absolute

注意：	在单次模式中，调用 ECAP_OneshotReArm()函数会复位 ECAP 的所有捕获事件，使之为下一次单次捕获模式做好准备。

3.2.2 实例 4：通过相对时间计算 PWM 频率

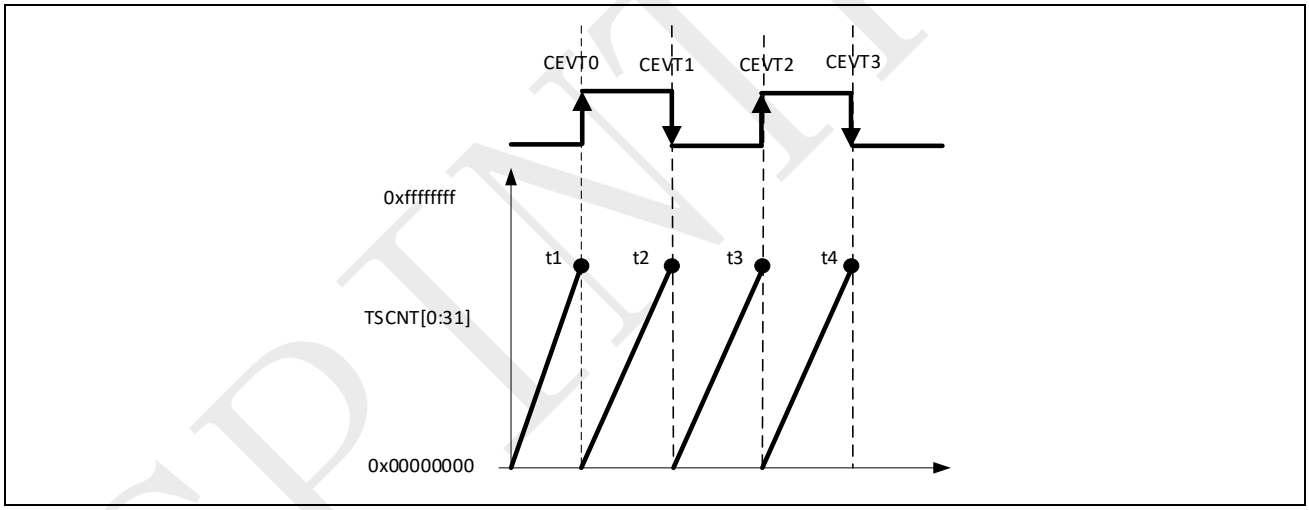
3.2.2.1 功能需求

使用单次捕获模式，以相对时间戳计数方式，对 PWM 进行频率计算。

3.2.2.2 功能实现

1. 确定捕获信息：以相对时间戳方式计数，对 PWM 波进行单次捕获。可获取图 3-4 所示信息，包括：
   a) 波形的周期 T=t1（或 t3）+t2（或 t4）；
   b) 波形高占比时间 Th = t2 或 Th = t4；
   c) 波形为低占时间 TL = t1 或 TL = t3；

图 3-4：单次模式（相对时间戳计数）



1. 产生待计算频率的 PWM 波：

   初始化 PWM 时钟；

   配置 PWM 外设，产生单通道 PWM 输出；

2. 配置 ECAP 功能：

   初始化 ECAP 为捕获模式，初始化函数默认捕获事件为 CAP0 上升沿捕获、CAP1 下降沿捕获、CAP2 上升沿捕获、CAP3 下降沿捕获；

- 配置 ECAP 为单次模式
  a) 设置被捕获的引脚为输入功能，将被捕获的引脚的 PWM 信号送入 ECAP；

     使能被捕获引脚的滤波功能并设置滤波窗口，减少干扰带来的误差；
  b) 使能 ECAP_CEVT0~3，当事件发生，ECAP 的计数器都进行复位，重新开始计数；

c) 使能 EVT3 事件中断，当事件发生将会产生中断，然后在中断服务函数中通过时间戳寄存器 CAPx 进行计算波形的频率；

- 频率计算公式为：

$$\frac{ECAP\_CLK}{[(t1+t2)+(t2+t3)]/2}$$

单次模式(相对时间戳计数)

```
#include <stdio.h>

#include "spc2188.h"


/* This macro is used to get the PWM period with a specified frequency */
#define PWMPeriod(u32PWMFreqHz)
((CLOCK_GetModuleClock(PWM_MODULE))/u32PWMFreqHz)/2;
#define PWM_FREQ                                10000
/* 10kHz PWM */
#define ECAP_PIN                                PIN_GPIO40
/* ECAP PIN */
#define ECAP_PIN_GPIO_FUNC                       PIN_GPIO40_GPIO40
/* ECAP PIN act as GPIO function */
#define ECAP_PIN_PWM_FUNC                        PIN_GPIO40_PWM2A
/* ECAP PIN act as PWM function */
/*usd the count to calculate the period*/
#define CNTTOFREQ(x)
(CLOCK_GetModuleClock(ECAP_MODULE) / (x))


uint32_t TStamp0, TStamp1, TStamp2, TStamp3;
uint32_t u32PWMPeriod;


int main(void)
{
    CLOCK_InitWithRCO(CLOCK_CPU_MAX);

    Delay_Init();

    /*
     * Init the UART
     */
    PIN_SetChannel(PIN_GPIO62, PIN_GPIO62_UART0_TXD);
    PIN_SetChannel(PIN_GPIO63, PIN_GPIO63_UART0_RXD);
    UART_Init(UART0, 38400);

    printf("Enter the test\n");

    /* Init PWM2, PWM freq will be set as 10kHz in this function */
    PWM_InitSingleChannel(PWM2, PWM_CHA, PWM_FREQ);

    /* Set PWM2A output 50% duty waveform */
    u32PWMPeriod = PWMPeriod(PWM_FREQ);
    PWM_SetCMPA(PWM2, u32PWMPeriod / 2);

    /* Start PWM2 */
    PWM_RunCounter(PWM2);

    /* ECAP Init */
```

单次模式(相对时间戳计数)

```
    ECAP_CaptureModeInit(ECAP0, ECAP_PIN);

    /* Set ECAP as onshot mode */
    ECAP_EnableOneshotMode(ECAP0);

    /* Can divide the input signal to observe higher frequency signals */
    ECAP_SetEventDiv(ECAP0, 1);

    /* RE-arm ECAP event count register */
    ECAP_OneshotReArm(ECAP0);

    /* Set the count reset back to zero when event3 happened */
    ECAP_EnableEventResetCounter(ECAP0, ECAP_CEVT0);
    ECAP_EnableEventResetCounter(ECAP0, ECAP_CEVT1);
    ECAP_EnableEventResetCounter(ECAP0, ECAP_CEVT2);
    ECAP_EnableEventResetCounter(ECAP0, ECAP_CEVT3);

    /* Enable EVT3 overflow Interrupt */
    ECAP_EnableInt(ECAP0, ECAP_INT_CEVT3);

    /* Enable global interrupt of EACP */
    NVIC_EnableIRQ(ECAP0_IRQn);

    /* Select ECAP_PIN as the channel A output of PWM2 */
    PIN_SetChannel(ECAP_PIN, ECAP_PIN_PWM_FUNC);
    PIN_EnableInputChannel(ECAP_PIN, ECAP_PIN_GPIO_FUNC);

    /* Enable Pin input deglitch filter */
    PIN_EnableDeglitch(ECAP_PIN);

    /* The smaller the value, the more accurate the measurement result, but the
more susceptible to interference */
    PIN_SetDeglitchWindow(DGCLKCTL_DIV_(0x0));

    /* Start ECAP */
    ECAP_RunCounter(ECAP0);

    while (1)
    {

    }
}


void ECAP0_IRQHandler(void)
{
    uint32_t cnt;

    TStamp0 = ECAP->CAP0;
    TStamp1 = ECAP->CAP1;
    TStamp2 = ECAP->CAP2;
    TStamp3 = ECAP->CAP3;

    cnt = TStamp2 + TStamp3;
    printf("Fre = %dHz\n", CNTTOFREQ(cnt));

    /* Clear CEVT3 */
    ECAP_ClearInt(ECAP, ECAP_INT_CEVT3);
    ECAP_ClearInt(ECAP, ECAP_INT_GLOBAL);
    ECAP_OneshotReArm(ECAP);
}
```

[1] 示例代码适用于 SPC2188。其它系列产品的示例代码会根据实际需求进行补充。

注意：	在单次模式中，调用 ECAP_OneshotReArm()函数会复位 ECAP 的所有捕获事件，使之为下一次单次捕获模式做好准备。

3.3 双通道捕获

对于表 1-1 中支持 ECAP 双引脚输入的 MCU 型号，可以参考本章节实例。

3.3.1 实例 5：通过绝对时间计算 PWM 占空比与频率

3.3.1.1 功能需求

使用连续捕获模式，以绝对时间戳计数方式，对 PWM 双通道波形进行频率计算。

3.3.1.2 功能实现

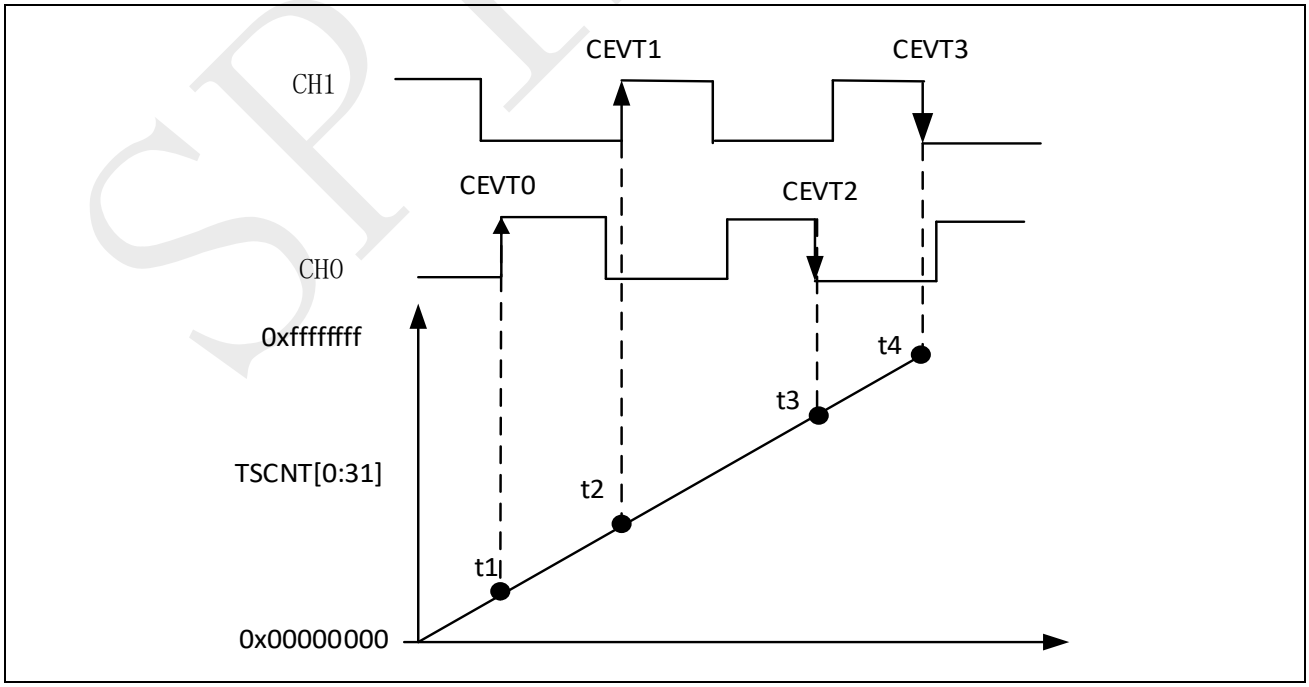
确定捕获信息：以绝对时间戳方式计数，对 PWM 双通道波形进行连续捕获。可获取图 3-5 图 3-5 所示信息，包括：

a) 波形的周期 T=(t2 – t1) + (t4 – t3)；

b) 相位差：

$$[\frac{(t2-t1)}{T}] \times \pi$$

图 3-5：双通道捕获（绝对时间戳计数）



产生待计算频率的 PWM 波：

- 初始化 PWM 时钟；
- 配置 PWM 外设，产生双通道通道 PWM 输出；
- 初始化 ECAP 为双通道模式，并将 PWM 产生的两路信号送进 ECAP 进行捕获。初始化函数默认设置捕获事件为 CAP0 上升沿捕获、CAP1 上升沿捕获、CAP2 下降沿捕获、CAP3 下降沿捕获；
- 设置被捕获的引脚为输入功能，将被捕获的引脚的 PWM 信号送入 ECAP；
- 使能被捕获引脚的滤波功能并设置滤波窗口，减少干扰带来的误差；
- 使能 ECAP_CEVT3，当事件 3（CAP3 捕获下降沿）发生，ECAP 的计数器将进行复位，重新开始计数；
- 使能 EVT3 事件中断，当事件 3 发生将会产生中断，然后在中断服务函数中通过时间戳寄存器 CAPx 进行计算波形的频率；
- 频率计算公式为：

$$\frac{ECAP\_CLK}{(t2-t1)+(t4-t3)}$$

以上实现步骤的示例代码可参考 SDK 提供的 Demo，如表 3-4：

表 3-4：实例 5 代码路径

MCU 产品型号	代码路径
SPC1169、SPD1179、SPD1176、SPC2188、SPC1185	SDK 目录\0_Examples\ ECAP_Continue_Absolute_Dual_Pin

3.4 APWM 模式

3.4.1 实例 6：APWM 模式下输出 PWM

3.4.1.1 功能需求

使用 ECAP 的 APWM 模式，输出 PWM 波形。

- 频率为 10KHz
- 占空比为 50%

3.4.1.2 功能实现

1. 设置 ECAP 为 APWM 模式。

- 设置 APWM 的周期 count 值：

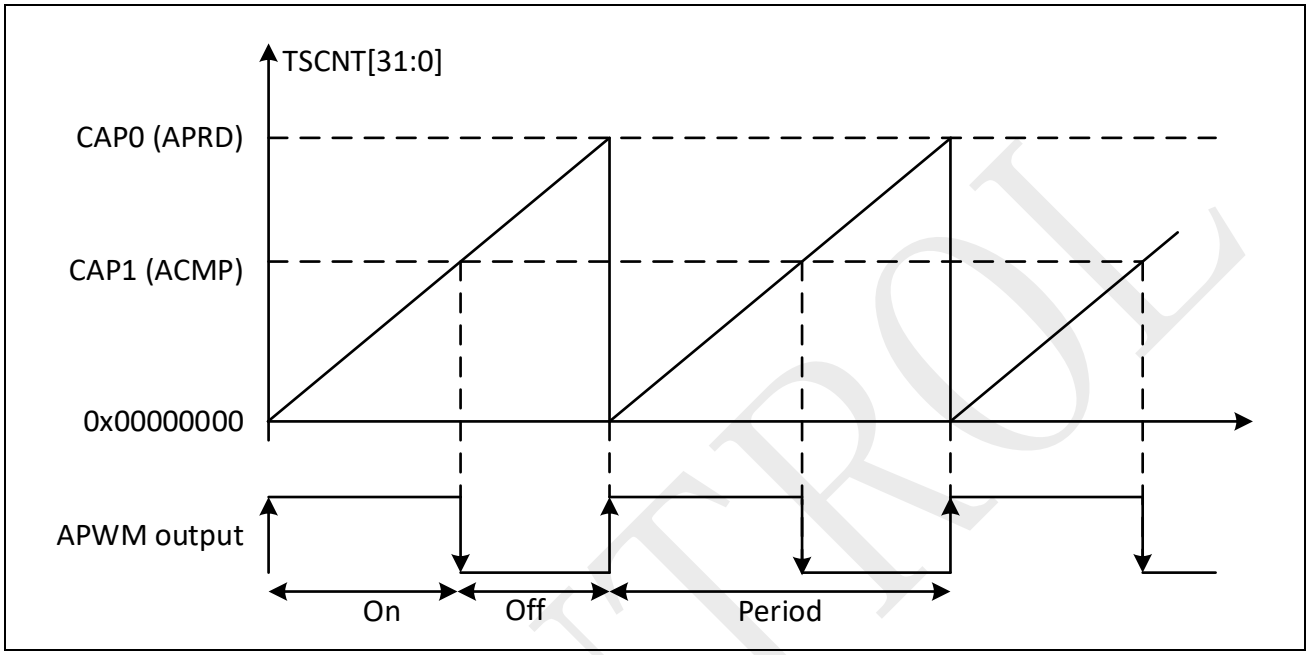
$$count = ECAP\_CLK/PwmFreq$$

- 设置 APWM 的占空比：范围 0~10000，对应 0%~50%。

2. 设置对应功能的引脚作为 APWM 功能；

- ECAP 在 APWM 模式下输出 10KHz 的波形，产生的波形如图 3-6。

图 3-6：APWM 模式



以上实现步骤的示例代码可参考 SDK 提供的 Demo，如表 3-5。

表 3-5：实例 6 代码路径

MCU 产品型号	代码路径
SPC1125，SPC1128，SPC1155，SPC1156，SPC1158，SPC1168，SPD1148，SPD1178，SPD1188，SPD1163，SPM1173，SPC1169，SPD1179，SPD1176，SPC2188，SPC1185	SDK 目录\0_Examples\ ECAP_APWM_Function

[1] SPC2168，SPC2166，SPC1198 等型号产品，目前提供的 SDK 无上述 demo，后续根据需求情况逐渐增加。