Application Note

SPC11X8/SPD11X8 ECAP 单元使用指南

2019 年 8 月 – 版本 1

目录

表格列表

图片列表

1 ECAP 概述

SPC11X8/SPD11X8 内建一个 Enhanced capture (ECAP)单元，用于抓取外部事件发生时的精确时间点。

SPC11X8/SPD11X8 的 ECAP 单元可以用作以下功能：

- 测量旋转机械的转速；
- 测量脉冲信号的周期及占空比；
- 解码加密的占空比信息；

SPC11X8/SPD11X8 的 ECAP 单元有以下特点：

- 基于 32bit 的计时器，且精度可达 5ns（时钟为 200MHz）/10ns（时钟为 100MHz）；
- 4 个事件时间戳寄存器；
- 可以为多大 4 个带时间戳的时间设置事件捕获极性；
- 4 个事件的每个事件都有对应的中断可触发；
- Single shot 抓捕模式下可以抓捕 4 个事件的时间戳；
- Continuous 模式下事件时间戳被存储在 4 深度的循环缓冲区中；
- 捕获绝对时间戳；
- 以上所有特性都基于一个输入管脚；
- 如果 ECAP 单元不使用捕获模式，也可以配置成 PWM 输出。

1.1 宏和函数

在 ECAP 的驱动库中，已经有下列驱动函数可供调动，可大大方便用户使用和理解，如表 1-1 和表 1-2 所示。

表 1-1: ECAP 宏定义列表

宏名	功能及参数说明
ECAP_APWMSetActiveHigh(ECAPx)	当 ECAP 作用 PWM 模式时，设置当 CAPTSCNT 数值等于 CMP 时 ECAP 输出高电平。
ECAP_APWMSetActiveLow(ECAPx)	当 ECAP 作用 PWM 模式时，设置当 CAPTSCNT 数值等于 CMP 时 ECAP 输出低电平。
ECAP_RunCounter(ECAPx)	使能 ECAP 时间计数器。
ECAP_StopCounter(ECAPx)	关闭 ECAP 时间计数器。
ECAP_SetSyncReloadValue(ECAPx, u32Val)	设置 CAPTSCNT 在同步之后的加载数值。U32Val:CNT 加载数值。
ECAP_EnableSync(ECAPx)	使能当同步事件发生时 CAPTSCNT=CAPCNTPHS。
ECAP_DisableSync(ECAPx)	关闭同步事件。

ECAP_ForceSync(ECAPx)	强制 CAPCNTPHS=CAPTSCNT。
ECAP_OneshotReArm(ECAPx)	执行一次 Re-arm 功能。
ECAP_EnableOneshotMode(ECAPx)	使能 One-shot 模式。
ECAP_DisableOneshotMode(ECAPx)	关闭 One-shot 功能。
ECAP_SetEventDiv(ECAPx, u16Div)	设置事件分频。 U16Div：分频系数。
ECAP_EnableEventResetCounter(ECAPx, eCapEvt)	使能 CAPTSCNT 在相应事件发生时 Reset。 eCapEvt：枚举类型 ECAP_EvtEnum。
ECAP_DisableEventResetCounter(ECAPx, eCapEvt)	关闭 CAPTSCNT 在相应事件发生时 Reset。 eCapEvt：枚举类型 ECAP_EvtEnum。
ECAP_SetEventTriggeredOnRisingEdge(ECAPx, eCapEvt)	设置相应事件在上升沿发生。 eCapEvt：枚举类型 ECAP_EvtEnum。
ECAP_SetEventTriggeredOnFallingEdge(ECAPx, eCapEvt)	设置相应事件在下降沿发生。 eCapEvt：枚举类型 ECAP_EvtEnum。
ECAP_SetCounterValue(ECAPx, u32CntVal)	设置计数器 CAPTSCNT 的数值。 u32CntVal：计数数值。
ECAP_EnableCEVT0Int(ECAPx)	使能事件 0 的中断。
ECAP_DisableCEVT0Int(ECAPx)	关闭事件 0 的中断。
ECAP_EnableCEVT1Int(ECAPx)	使能事件 1 的中断。
ECAP_DisableCEVT1Int(ECAPx)	关闭事件 1 的中断。
ECAP_EnableCEVT2Int(ECAPx)	使能事件 2 的中断。
ECAP_DisableCEVT2Int(ECAPx)	关闭事件 2 的中断。
ECAP_EnableCEVT3Int(ECAPx)	使能事件 3 的中断。
ECAP_DisableCEVT3Int(ECAPx)	关闭事件 3 的中断。
ECAP_EnableCounterOverflowInt(ECAPx)	使能计数器 CAPTSCNT 溢出中断。
ECAP_DisableCounterOverflowInt(ECAPx)	关闭计数器 CAPTSCNT 溢出中断。
ECAP_EnableCounterEqualPRDInt(ECAPx)	使能计数器 CAPTSCNT=PRD 中断。

ECAP_DisableCounterEqualPRDInt(ECAPx)	关闭计数器 CAPTSCNT=PRD 中断。
ECAP_EnableCounterEqualCMPInt(ECAPx)	使能计数器 CAPTSCNT=CMP 中断。
ECAP_DisableCounterEqualCMPInt(ECAPx)	关闭计数器 CAPTSCNT=CMP 中断。
ECAP_EnableInt(ECAPx, eIntEvt)	使能相应事件的中断。 eIntEvt：枚举类型 ECAP_IntEnum。
ECAP_DisableInt(ECAPx, eIntEvt)	关闭相应事件的中断。 eIntEvt：枚举类型 ECAP_IntEnum。
ECAP_GetCEVT0IntFlag(ECAPx)	获取事件 0 的中断标记。
ECAP_GetCEVT1IntFlag(ECAPx)	获取事件 1 的中断标记。
ECAP_GetCEVT2IntFlag(ECAPx)	获取事件 2 的中断标记。
ECAP_GetCEVT3IntFlag(ECAPx)	获取事件 3 的中断标记。
ECAP_GetCounterOverflowIntFlag(ECAPx)	获取 CAPTSCNT 溢出的中断标记。
ECAP_GetCounterEqualPRDIntFlag(ECAPx)	获取 CAPTSCNT=PRD 的中断标记。
ECAP_GetCounterEqualCMPIntFlag(ECAPx)	获取 CAPTSCNT=CMP 的中断标记。
ECAP_GetGlobalIntFlag(ECAPx)	获取全局中断标记。
ECAP_IsGlobalIntPending(ECAPx)	获取全局中断标记。
ECAP_GetIntFlag(ECAPx, eIntEvt)	获取相应事件的中断标记。 eIntEvt：枚举类型 ECAP_IntEnum。
ECAP_ForceCEVT0Int(ECAPx)	强制发生事件 0 的中断。
ECAP_ForceCEVT1Int(ECAPx)	强制发生事件 1 的中断。
ECAP_ForceCEVT2Int(ECAPx)	强制发生事件 2 的中断。
ECAP_ForceCEVT3Int(ECAPx)	强制发生事件 3 的中断。
ECAP_ForceCounterOverflowInt(ECAPx)	强制发生 CAPTSCNT 溢出中断。
ECAP_ForceCounterEqualPRDInt(ECAPx)	强制发生 CAPTSCNT=PRD 中断。
ECAP_ForceCounterEqualCMPInt(ECAPx)	强制发生 CAPTSCNT=CMP 中断。

ECAP_ForceInt(ECAPx, eIntEvt)	强制发生相应的事件中断。 eIntEvt：枚举类型 ECAP_IntEnum。
ECAP_ClearCEVT0Int(ECAPx)	清除事件 0 中断。
ECAP_ClearCEVT1Int(ECAPx)	清除事件 1 中断。
ECAP_ClearCEVT2Int(ECAPx)	清除事件 3 中断。
ECAP_ClearCEVT3Int(ECAPx)	清除事件 3 中断。
ECAP_ClearCounterOverflowInt(ECAPx)	清除 CAPTSCNT 溢出中断。
ECAP_ClearCounterEqualPRDInt(ECAPx)	清除 CAPTSCNT=PRD 中断。
ECAP_ClearCounterEqualCMPInt(ECAPx)	清除 CAPTSCNT=CMP 中断。
ECAP_ClearGlobalInt(ECAPx)	清除 ECAP 全局中断。
ECAP_ClearInt(ECAPx, eIntEvt)	清除相应事件的中断。
ECAP_WALLOW(ECAPx)	使能 ECAP 寄存器写操作。
ECAP_WDIS(ECAPx)	关闭 ECAP 寄存器写操作。

表 1-2: ECAP 函数定义列表

函数名	功能及参数说明
ECAP_CaptureModeInit(ECAPx, ePinNum)	初始化 ECAP 作为 Capture 模式，并使能工作。 ePinNum：ECAP 输入的 PIN 脚号。
ECAP_APWMModeInit(ECAPx, ePinNum, u32PwmFreq)	初始化 ECAP 作为 APWM 模式，并使能工作。 ePinNum：APWM 输出的 PIN 脚号。 u32PwmFreq：APWM 输出的频率。
ECAP_APWMSetDuty(ECAPx, u32PwmDuty)	设置 ECAP 作为 APWM 时的占空比。 u32PwmDuty：占空比数值（0~1000000）
ECAP_SetInputPin(ECAPx, ePinNum)	设置一个 GPIO 作为 ECAP 的输入 PIN 脚。 ePinNum：枚举类型 GPIO_PinEnum。
ECAP_SetSyncFromGPIO(ECAPx, ePinNum, ePinLevel)	设置一个 GPIO 作为同步事件源。 ePinNum：枚举类型 GPIO_PinEnum。 ePinLevel：枚举类型 GPIO_LevelEnum。

2 ECAP 实例

2.1 ECAP Absolute Operation In Continue Mode 实例

本示例中，会展示 ECAP 在 Continue 模式下以 Absolute 方式工作的示例。

首先定义要用到的全局变量及宏定义

Example Code

```
/* 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 */

/*usd the count to calculate the period*/
#define CNTTOFREQ(x) (CLOCK_GetModuleClock(ECAP_MODULE) / (x))

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

在 main()中，进行 ECAP 的基本配置

Example Code

```
int main()
{
  FLASH_WALLOW();
  FLASH_SetTiming(200000000);
  /* Disable flash write access after flash operation had done */
  FLASH_WDIS();

  CLOCK_InitWithRCO(CLOCK_HCLK_200MHZ);

  Delay_Init();

  /*
   * Init the UART
   *
   * 1.Set the GPIO44/45 as UART FUNC
   *
   * 2.Enable the UART CLK
   *
   * 3.Set the rest para
```

```
   */
  GPIO_SetPinChannel(GPIO_34,GPIO34_UART_TXD);
  GPIO_SetPinChannel(GPIO_35,GPIO35_UART_RXD);
  CLOCK_EnableModule(UART_MODULE);
  UART_Init(UART,38400);

  /* Init PWM5, PWM freq will be set as 20kHz in this function */
  PWM_SingleChannelInit(PWM5,PWM_CHA,PWM_FREQ);

  /* Set PWM5A output 50% duty waveform */
  u32PWMPeriod = PWMPeriod(PWM_FREQ);
  PWM_SetCMPA(PWM5,u32PWMPeriod / 2);

  /* ECAP Init */
  ECAP_CaptureModeInit(ECAP,GPIO_28);

  /* Select GPIO57 as the channel A output of PWM5 */
  GPIO_SetPinChannel(GPIO_28,GPIO28_PWM5A);

  /* Set the count do not reset back to zero when event happened */
  ECAP_DisableEventResetCounter(ECAP, ECAP_CEVT0);
  ECAP_DisableEventResetCounter(ECAP, ECAP_CEVT1);
  ECAP_DisableEventResetCounter(ECAP, ECAP_CEVT2);
  ECAP_DisableEventResetCounter(ECAP, ECAP_CEVT3);

  /* Only Enable EVT3 INT */
  ECAP_DisableCEVT0Int(ECAP);
  ECAP_DisableCEVT1Int(ECAP);
  ECAP_DisableCEVT2Int(ECAP);

  /* Enable overflow Int */
  ECAP_EnableCounterOverflowInt(ECAP);

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

  /* Start PWM5 */
  PWM_RunCounter(PWM5);

  while(1){ }
}
```

在中断函数中，打印波形频率的数据。

Example Code

```
void ECAP_IRQHandler(void)
{
  uint32_t cnt;

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

  cnt = ((TStamp2 - TStamp0) + (TStamp3 - TStamp1)) / 2;
  printf("Frequency = %dHz\n", CNTTOFREQ(cnt));

  /*
   * Key Point : the follow two step must keep the position as it was and
   *do not change. Reason : if global INT clear after CEVT3 INT, there
   *may be another CEVT3 INT happened, then IRQHandler will enter
   *immediately when CEVT3 clear, this will cause a disordered value in
   * CAP0~3.
   */
  /* Clear global INT of ECAP */
  ECAP_ClearGlobalInt(ECAP);

  /* Clear CEVT3 */
  ECAP_ClearCEVT3Int(ECAP);
}
```

2.2 ECAP Delta Operation In Continue Mode 实例

本示例中，会展示 ECAP 在 Continue 模式下以 Delta 方式工作的示例。

首先定义要用到的全局变量及宏定义

Example Code

```
/* This macro is used to get the PWM period with a specified frequency
*/
#define PWMPeriod(u32PWMFreqHz)
((CLOCK_GetModuleClock(PWM_MODULE))/u32PWMFreqHz)/2;

/* Usd the count to calculate the period */
#define CNTTOFREQ(x) (CLOCK_GetModuleClock(ECAP_MODULE) / (x))

#define PWM_FREQ          10000            /* 10kHz PWM */

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

在 main()中，进行 ECAP 的基本配置

Example Code

```
int main()
{
  FLASH_WALLOW();
  FLASH_SetTiming(200000000);
  /* Disable flash write access after flash operation had done */
  FLASH_WDIS();

  CLOCK_InitWithRCO(CLOCK_HCLK_200MHZ);

  Delay_Init();

  /*
   * Init the UART
   *
   * 1.Set the GPIO44/45 as UART FUNC
   *
   * 2.Enable the UART CLK
   *
   * 3.Set the rest para
   */
  GPIO_SetPinChannel(GPIO_34,GPIO34_UART_TXD);
  GPIO_SetPinChannel(GPIO_35,GPIO35_UART_RXD);
```

```
    CLOCK_EnableModule(UART_MODULE);
    UART_Init(UART,38400);

    /* Init PWM5, PWM freq will be set as 20kHz in this function */
    PWM_SingleChannelInit(PWM5,PWM_CHA,PWM_FREQ);

    /* Set PWM5A output 50% duty waveform */
    u32PWMPeriod = PWMPeriod(PWM_FREQ);
    PWM_SetCMPA(PWM5,u32PWMPeriod / 2);

    /* ECAP Init */
    ECAP_CaptureModeInit(ECAP,GPIO_28);

    /* Select GPIO57 as the channel A output of PWM5 */
    GPIO_SetPinChannel(GPIO_28,GPIO28_PWM5A);

    /* Only Enable EVT0 INT */
    ECAP_DisableCEVT1Int(ECAP);
    ECAP_DisableCEVT2Int(ECAP);
    ECAP_DisableCEVT3Int(ECAP);

    /* Enable overflow INT */
    ECAP_EnableCounterOverflowInt(ECAP);

    /* Enable global INT of EACP */
    NVIC_EnableIRQ(ECAP_IRQn);

    /* Start PWM5 */
    PWM_RunCounter(PWM5);

    while(1)
    {

    }
}
```

在中断函数中，打印波形频率的数据。

Example Code

```
void ECAP_IRQHandler(void)
{
  uint32_t cnt;

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

  cnt = ((TStamp2 + TStamp1) + (TStamp3 + TStamp0)) / 2;
  printf("Frequency = %dHz\n", CNTTOFREQ(cnt));

  /*
   * Key Point : the follow two step must keep the position as it was and do not change.
   *
   * Reason : if global INT clear after CEVT3 INT, there may be another CEVT0 INT happened, then
   * IRQHandler will enter immediately when CEVT0 clear, this will cause a disordered value in
   * CAP0~3.
   */
  /* Clear global INT of ECAP */
  ECAP_ClearGlobalInt(ECAP);

  /* Clear CEVT0 */
  ECAP_ClearCEVT0Int(ECAP);
}
```

2.3 ECAP Absolute Operation In One-shot Mode 实例

本示例中，会展示 ECAP 在 One-shot 模式下以 Absolute 方式工作的示例。

首先定义要用到的全局变量及宏定义

Example Code

```
/* This macro is used to get the PWM period with a specified frequency
*/
#define PWMPeriod(u32PWMFreqHz)
((CLOCK_GetModuleClock(PWM_MODULE))/u32PWMFreqHz)/2;

/* Usd the count to calculate the period */
#define CNTTOFREQ(x) (CLOCK_GetModuleClock(ECAP_MODULE) / (x))

#define PWM_FREQ              10000              /* 10kHz PWM */
#define ECAP_PIN              GPIO_28            /* ECAP PIN */
#define ECAP_PIN_GPIO_FUNC    GPIO28_GPIO28 /* ECAP PIN act as GPIO
function */
#define ECAP_PIN_PWM_FUNC     GPIO28_PWM5A   /* ECAP PIN act as PWM
function */

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

在 main()中，进行 ECAP 的基本配置

Example Code

```
int main()
{
  FLASH_WALLOW();
  FLASH_SetTiming(200000000);
  /* Disable flash write access after flash operation had done */
  FLASH_WDIS();

  CLOCK_InitWithRCO(CLOCK_HCLK_200MHZ);

  Delay_Init();

  /*
   * Init the UART
   *
   * 1.Set the GPIO44/45 as UART FUNC
   *
   * 2.Enable the UART CLK
```

```
   *
   * 3.Set the rest para
   */
  GPIO_SetPinChannel(GPIO_34,GPIO34_UART_TXD);
  GPIO_SetPinChannel(GPIO_35,GPIO35_UART_RXD);
  CLOCK_EnableModule(UART_MODULE);
  UART_Init(UART,38400);

  /* Init PWM5, PWM freq will be set as 20kHz in this function */
  PWM_SingleChannelInit(PWM5,PWM_CHA,PWM_FREQ);

  /* Set PWM5A output 50% duty waveform */
  u32PWMPeriod = PWMPeriod(PWM_FREQ);
  PWM_SetCMPA(PWM5,u32PWMPeriod / 2);

  /* ECAP Init */
  ECAP_CaptureModeInit(ECAP,ECAP_PIN);

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

  /* Set EVT DIV to 1 */
  ECAP_SetEventDiv(ECAP,1);

  /* Set the count do not reset back to zero when event happened */
  ECAP_DisableEventResetCounter(ECAP, ECAP_CEVT0);
  ECAP_DisableEventResetCounter(ECAP, ECAP_CEVT1);
  ECAP_DisableEventResetCounter(ECAP, ECAP_CEVT2);
  ECAP_DisableEventResetCounter(ECAP, ECAP_CEVT3);

  /* ---------------Steps for fixing BUG : BEGIN-----------------------
*/
  /*
   * As discribed in technical reference manual, the ECAP event prescaler
is implemented
   * with existing clock divider IP, so there is extra four event cycles
(regarded as clock
   * cycles to the divider) required to initialize the divider after
power-up. For one shot
   * mode, in which the user cares about exactly each incoming event,
the first 4 events may
   * not work correctly. So we need software work-around to purposely
generate 4 dummy events
   * via GPIO toggling in Gen 3 products to intialize the event prescaler.
```

```
   */

  /*
   * EVT INT are opened in 'ECAP_CaptureModeInit', the 4 dummy events
may cause dummy INT,
   * so disable all EVT INT when senting the 4 dummy event to the ECAP
module.
   */
  ECAP_DisableCEVT0Int(ECAP);
  ECAP_DisableCEVT1Int(ECAP);
  ECAP_DisableCEVT2Int(ECAP);
  ECAP_DisableCEVT3Int(ECAP);

  /* Set the ECAP PIN as the INPUT GPIO function in order to sent 4 dummy
event */
  GPIO_SetPinChannel(ECAP_PIN,ECAP_PIN_GPIO_FUNC);
  GPIO_WritePin(ECAP_PIN,GPIO_LEVEL_LOW);
  GPIO_SetPinDir(ECAP_PIN,GPIO_OUTPUT);

  /* Toggle the ECAP PIN 4 times as 4 events */
  GPIO_TogglePin(ECAP_PIN);
  GPIO_TogglePin(ECAP_PIN);
  GPIO_TogglePin(ECAP_PIN);
  GPIO_TogglePin(ECAP_PIN);

  /* Dummy Event may cause accidental INT, So clear all INT there */
  ECAP->CAPIC.all = 0xFFFFFFFF;

  /* RE-arm ECAP event count register(in order to clear the impact of
the 4 dummy events) */
  ECAP_OneshotReArm(ECAP);
  /* ---------------Steps for fixing BUG : END-----------------------
*/

  /* Enable the EVT INT */
  ECAP_EnableCEVT3Int(ECAP);

  /* Enable overflow INT */
  ECAP_EnableCounterOverflowInt(ECAP);

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

  /* Select ECAP_PIN as the channel A output of PWM5 */
```

```
  GPIO_SetPinChannel(ECAP_PIN,ECAP_PIN_PWM_FUNC);

  /* Start PWM5 */
  PWM_RunCounter(PWM5);

  while(1)
  {

  }
}
```

注意：以上主函数 main 中，为了弥补硬件设计上的一个缺陷，增加了软件 Work-around 解决办法，相关硬件设计缺陷在 TRM（Technical Reference Manual）中有相关描述。

在中断函数中，打印波形频率的数据。

Example Code

```
void ECAP_IRQHandler(void)
{
  uint32_t cnt;

  /* Clear CEVT3 */
  ECAP_ClearCEVT3Int(ECAP);

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

  cnt = ((TStamp2 - TStamp0) + (TStamp3 - TStamp1)) / 2;
  printf("Frequency = %dHz\n", CNTTOFREQ(cnt));

  /*
   * In oneshot mode, the global and CEVT3 INT had not relationship with
each other
   * as 'ECAP_Continue_Absolute'.
   */
  /* Clear global INT of ECAP */
  ECAP_ClearGlobalInt(ECAP);

  /* Rearm the ECAP for next events */
  ECAP_OneshotReArm(ECAP);
}
```

3 修订记录

表 3-1: 文档修订记录

日期	版本	修改内容
2019-08-01	1	初始版本