

# 第九讲 增强型脉冲捕获模块eCAP

# DSP

主讲：叶剑

电话：13728639620

Email: [yejian@hit.edu.cn](mailto:yejian@hit.edu.cn)

# 第九讲：增强型脉冲捕获模块eCAP



1、脉冲捕获基本原理

2、F28335增强型CAP

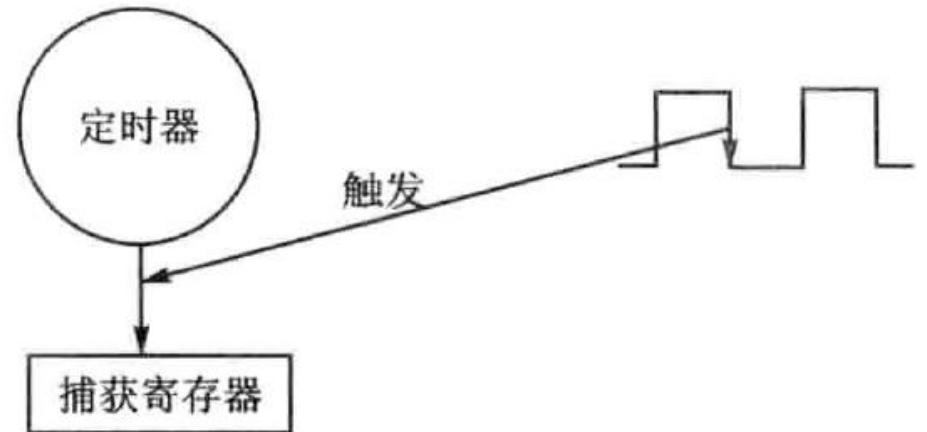
3、eCAP的捕获操作模式

4、eCAP的APWM操作模式

5、实用例子

# 脉冲捕获基本原理

eCAP用于**精确**测量外部信号时序。



典型应用如下：

1. 电机测速；
2. 位置传感器脉冲时间测量；
3. 测量一系列脉冲周期和占空比；
4. 电流/电压传感器的PWM编码信号的解码。

# 应用举例

- ◆ 在低速时为增量编码器估算速度：

问题：低速时，位置变化慢，造成转速估算的误差变大

$$v_k \approx \frac{x_k - x_{k-1}}{\Delta t}$$

解决方法：在固定的位置变化量之间测量时间

$$v_k \approx \frac{\Delta x}{t_k - t_{k-1}}$$

# 第九讲：增强型脉冲捕获模块eCAP

1、脉冲捕获基本原理



2、F28335增强型CAP

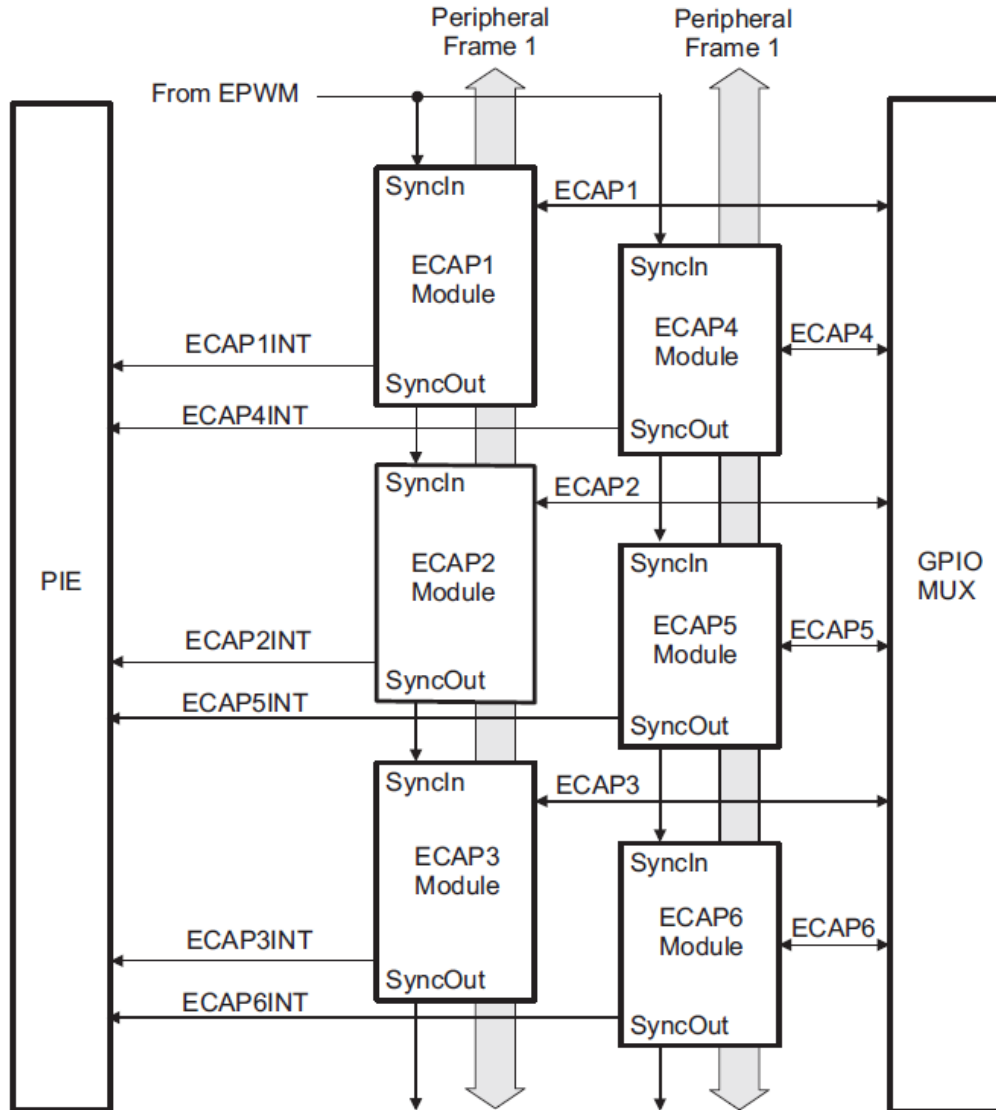
3、eCAP的捕获操作模式

4、eCAP的APWM操作模式

5、CAP寄存器介绍

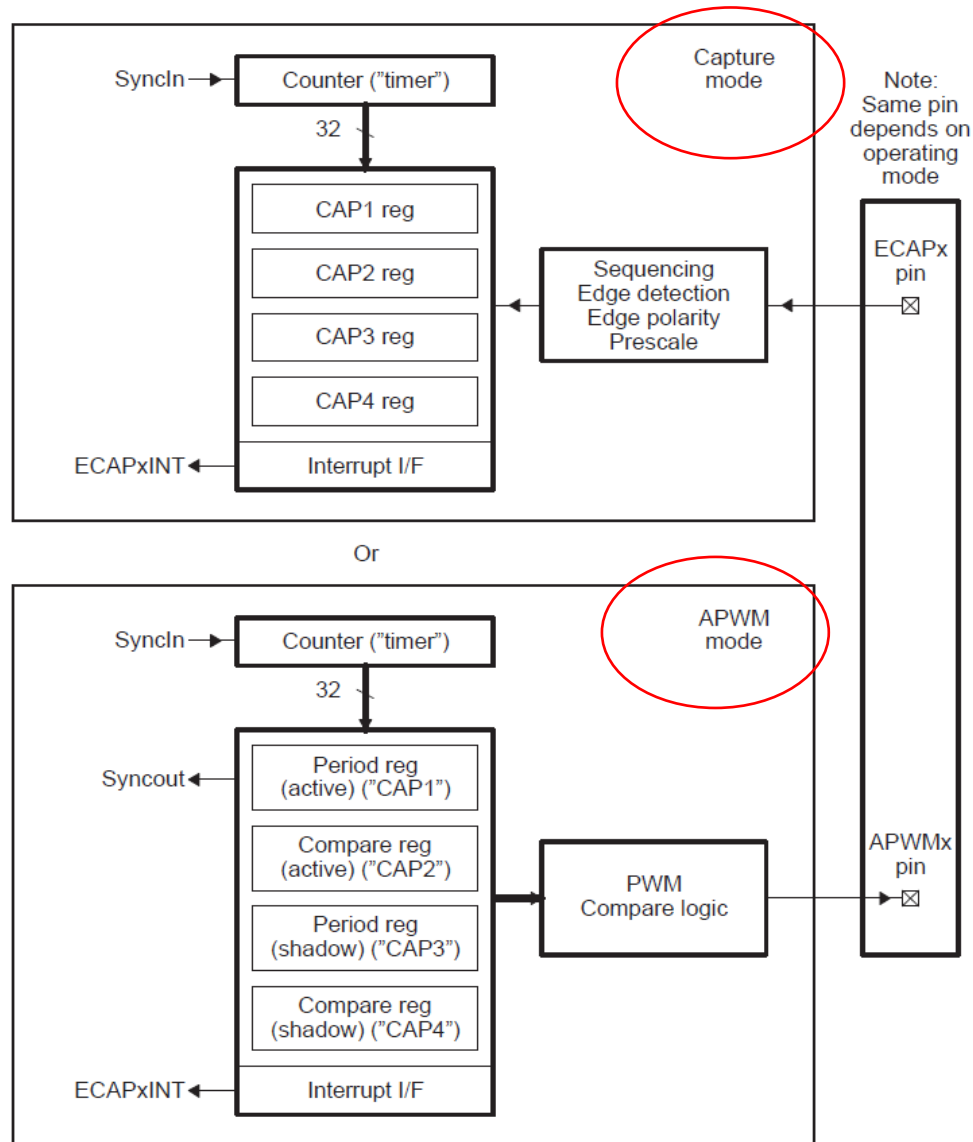
# eCAP模块简介

F28335共有6组eCAP模块



# eCAP模块简介

eCAP模块可以工作在捕获模式或APWM模式



# eCAP模块简介

F28335捕获模块的主要特征如下：

1. 150MHz系统时钟的情况下，32位时基的时间分辨率为6.67ns；
2. 4组32位的时间标识寄存器；
3. 4级捕获事件序列，可以灵活配置捕获事件边沿极性；
4. 四级触发事件均可以产生中断；
5. 软件配置一次捕获可以最多得到4个捕获时间；
6. 可连续循环4级捕获；
7. 绝对时间捕获；
8. 不同模式的时间捕获；
9. 所有捕获都发生在一个输入引脚上；
10. 如果eCAP模块不作捕获使用，可以配置成一个单通道输出的PWM模式。

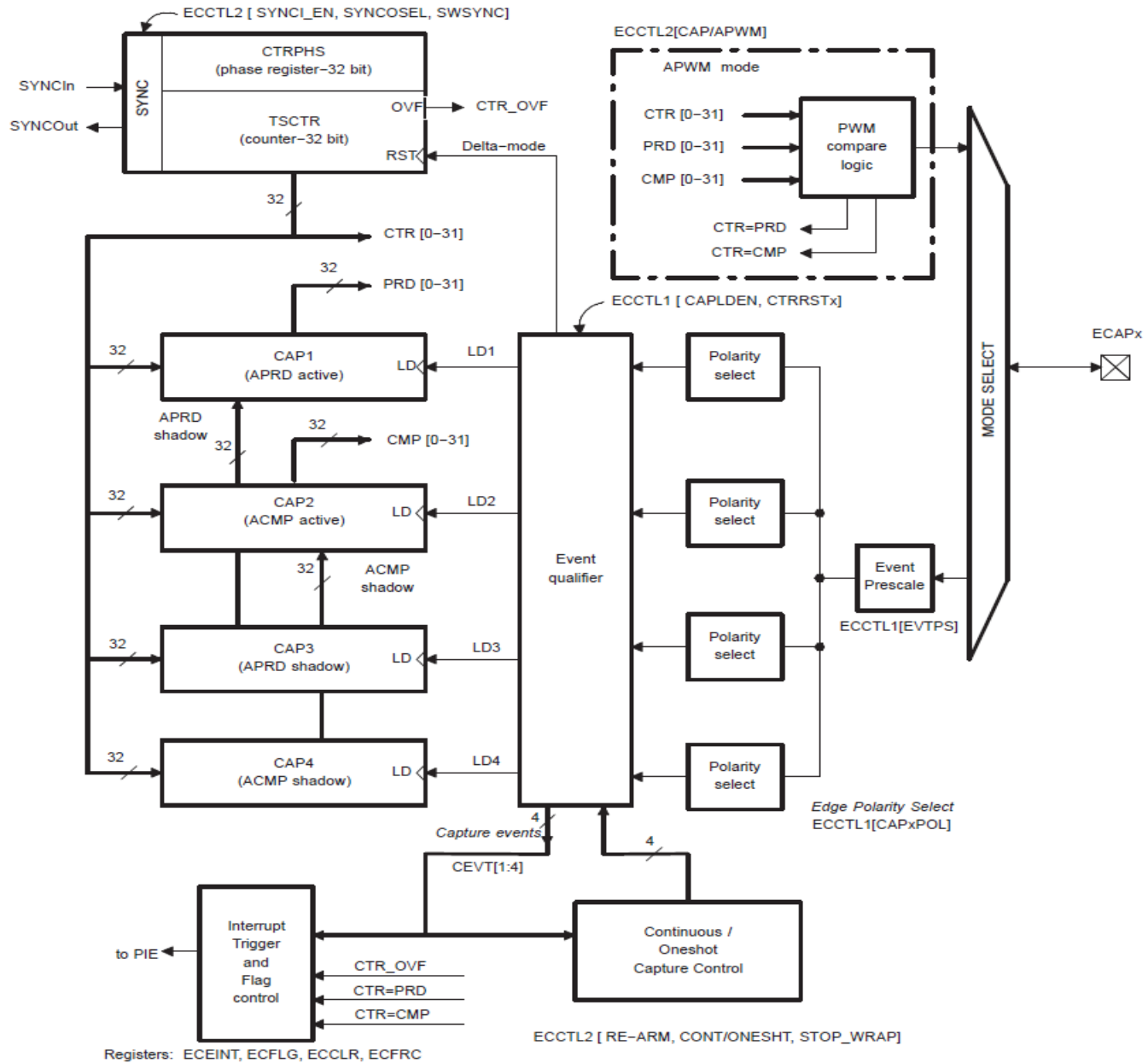


# eCAP模块简介

eCAP模块中一个捕获通道完成一次捕获任务，需要以下关键资源：

- 1、专用捕获输入引脚；
- 2、32位时基（时间标识计数器TSCTR）；
- 3、4组32位的时间标识寄存器(CAP1-CAP4)；
- 4、4级序列器(Modulo4 counter)与外部eCAP引脚的上升/下降沿同步；
- 5、4个事件可独立配置边沿极性；
- 6、输入捕获信号预分频（2-62）；
- 7、一个2位的比较寄存器，一次触发后可以捕获4个时间标签事件；
- 8、采用4级深度的循环缓冲器(CAP1-CAP4)以进行连续捕获；
- 9、4个捕获事件中任意一个都可以产生中断。

# eCAP模块简介



# 第九讲：增强型脉冲捕获模块eCAP

1、脉冲捕获基本原理

2、F28335增强型CAP



3、eCAP的捕获操作模式

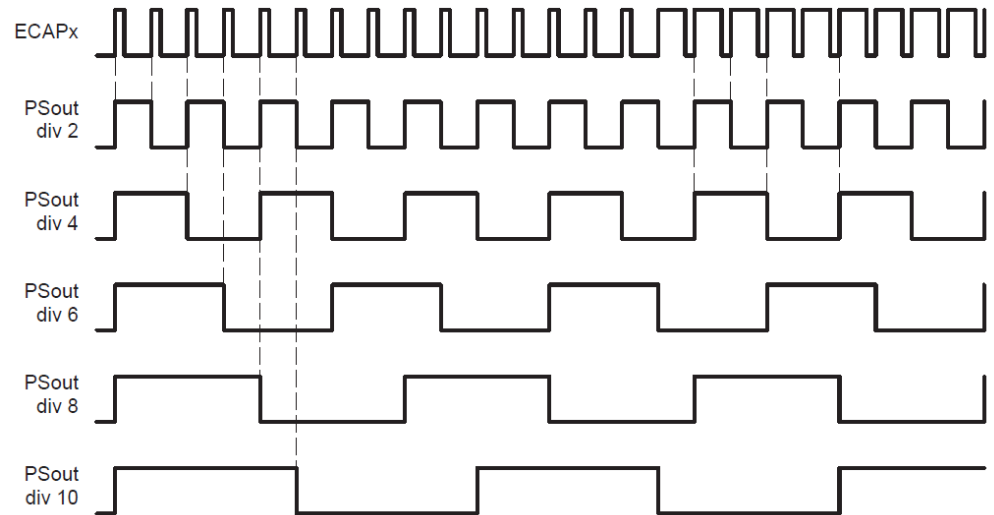
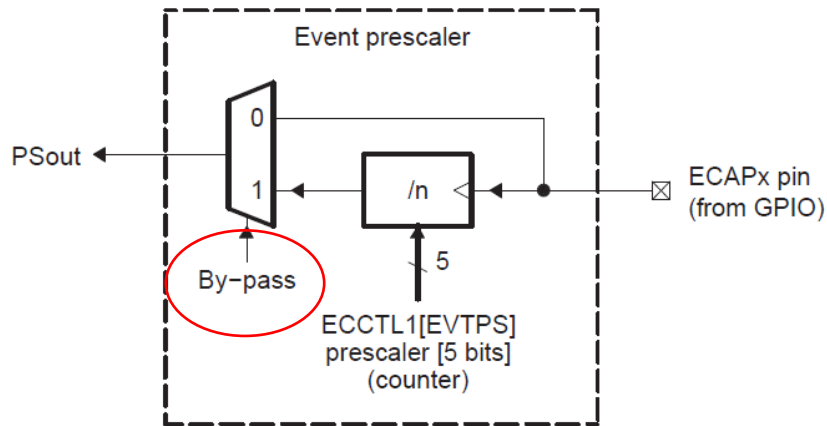
4、eCAP的APWM操作模式

5、实用例子

# eCAP的捕获工作模式

## 输入捕获信号预分频

可以对一个输入的捕捉信号进行分频系数为 $N=2\sim 62$ 的分频，这在输入信号**频率很高**的时候非常有用。



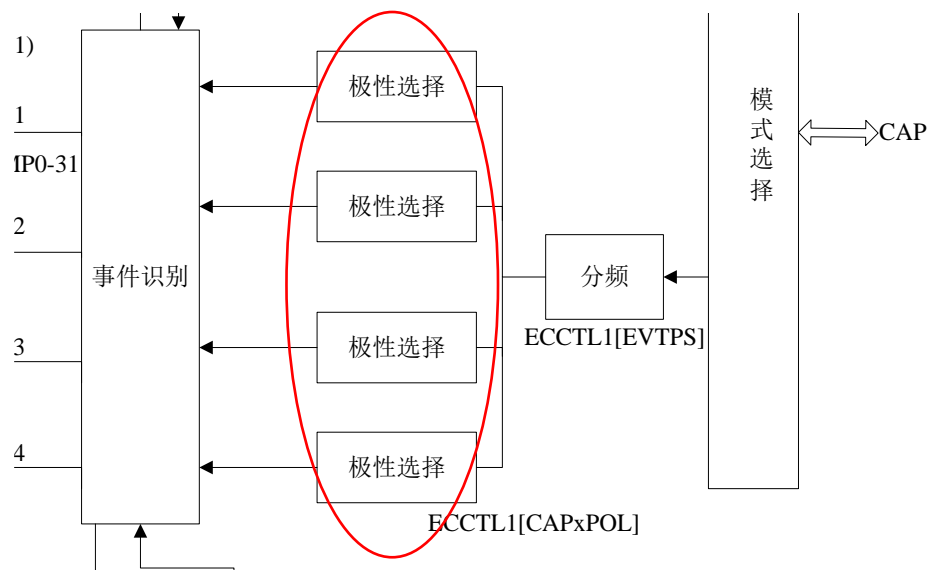
# 输入捕获信号预分频

可以对一个输入的捕捉信号进行分频系数为 $N=2^6$ 的分频，这在输入信号频率很高的时候非常有用。

13:9	PRESCALE	Event Filter prescale select
	00000	Divide by 1 (i.e., no prescale, by-pass the prescaler)
	00001	Divide by 2
	00010	Divide by 4
	00011	Divide by 6
	00100	Divide by 8
	00101	Divide by 10
	...	
	11110	Divide by 60
	11111	Divide by 62

# 边沿极性选择

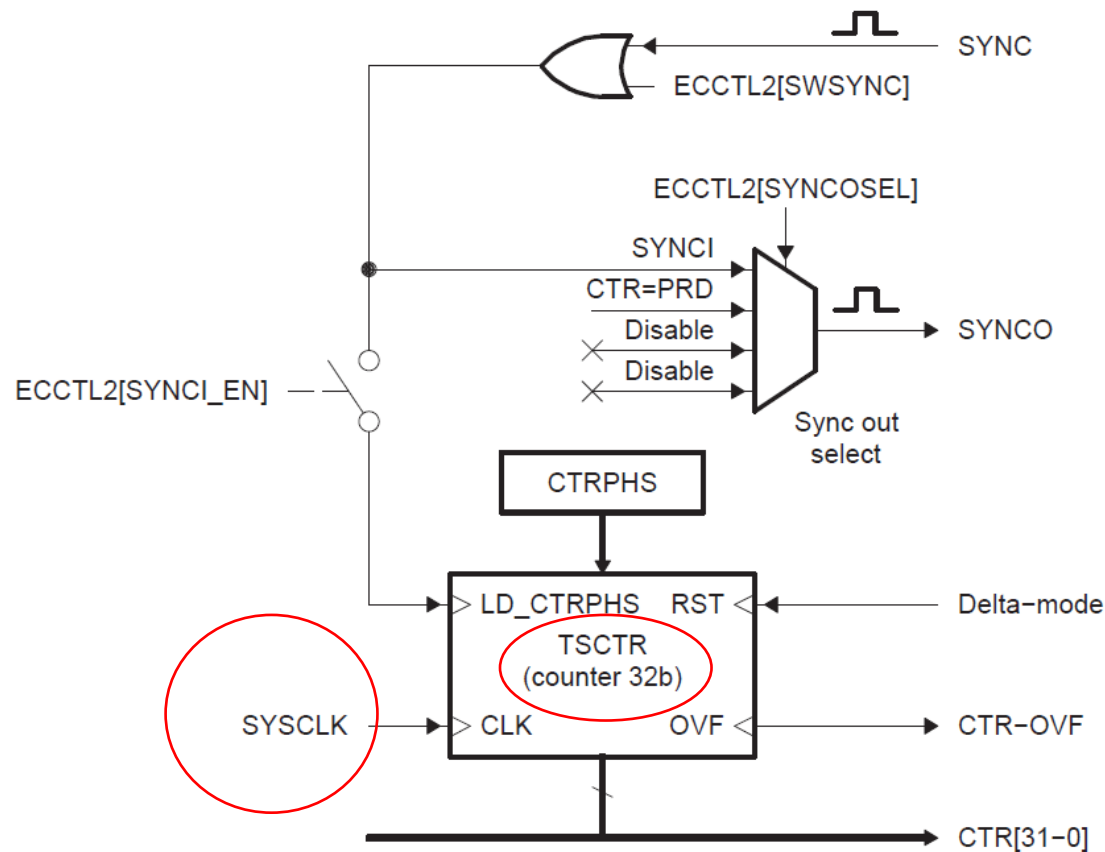
分别将4个捕获事件配置成上升沿或下降沿捕获。



6	CAP4POL		Capture Event 4 Polarity select
		0	Capture Event 4 triggered on a rising edge (RE)
		1	Capture Event 4 triggered on a falling edge (FE)

# 32位计数器与相位控制

1. 32位计数器counter为事件捕获提供**基准时钟**time-base，直接由系统时钟SYSCLK驱动。



# 32位计数器与相位控制

2. 相位寄存器CTRPHS通过**硬件或软件强制**，将多个eCAP计数器**同步**。在APWM模式中，这个相位寄存器在模块之间需要相位差时很有用。

## SWSYNC

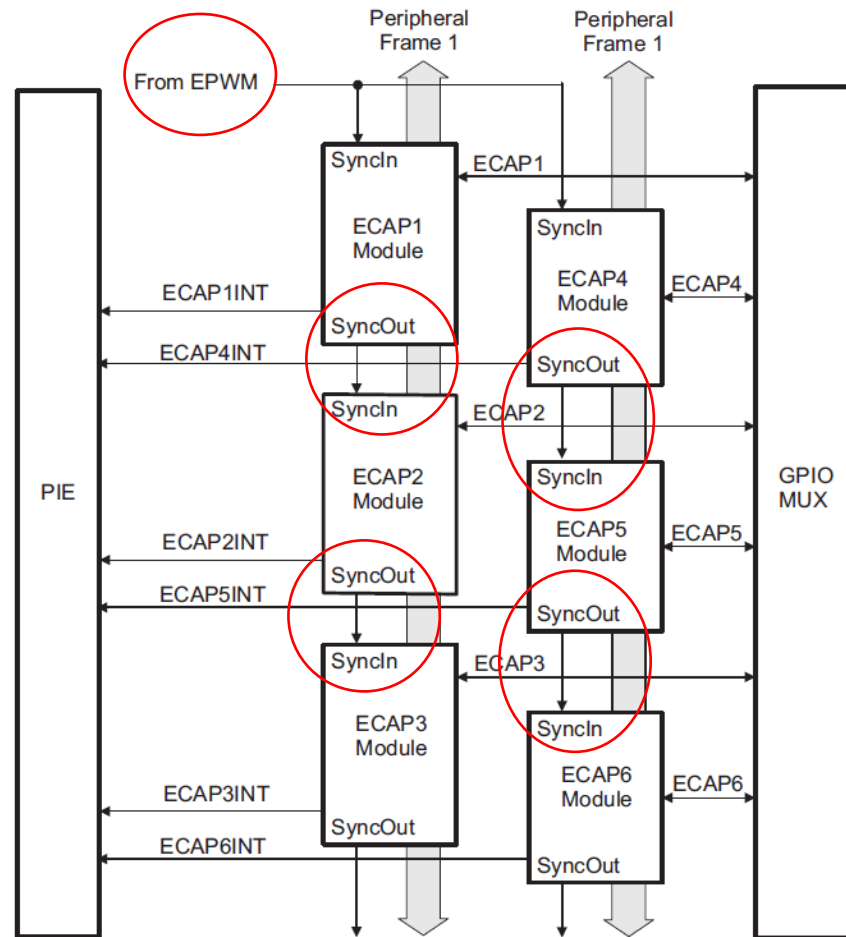
Software-forced  
Counter (TSCTR)  
Synchronizing

## SYNCO\_SEL

Sync-Out Select

## SYNCI\_EN

Counter (TSCTR)  
Sync-In select  
mode





# 32位计数器与相位控制

2. 相位寄存器**CTRPHS**通过**硬件或软件强制**，将多个eCAP计数器**同步**。在APWM模式中，这个相位寄存器在模块之间需要相位差时很有用。

## SWSYNC

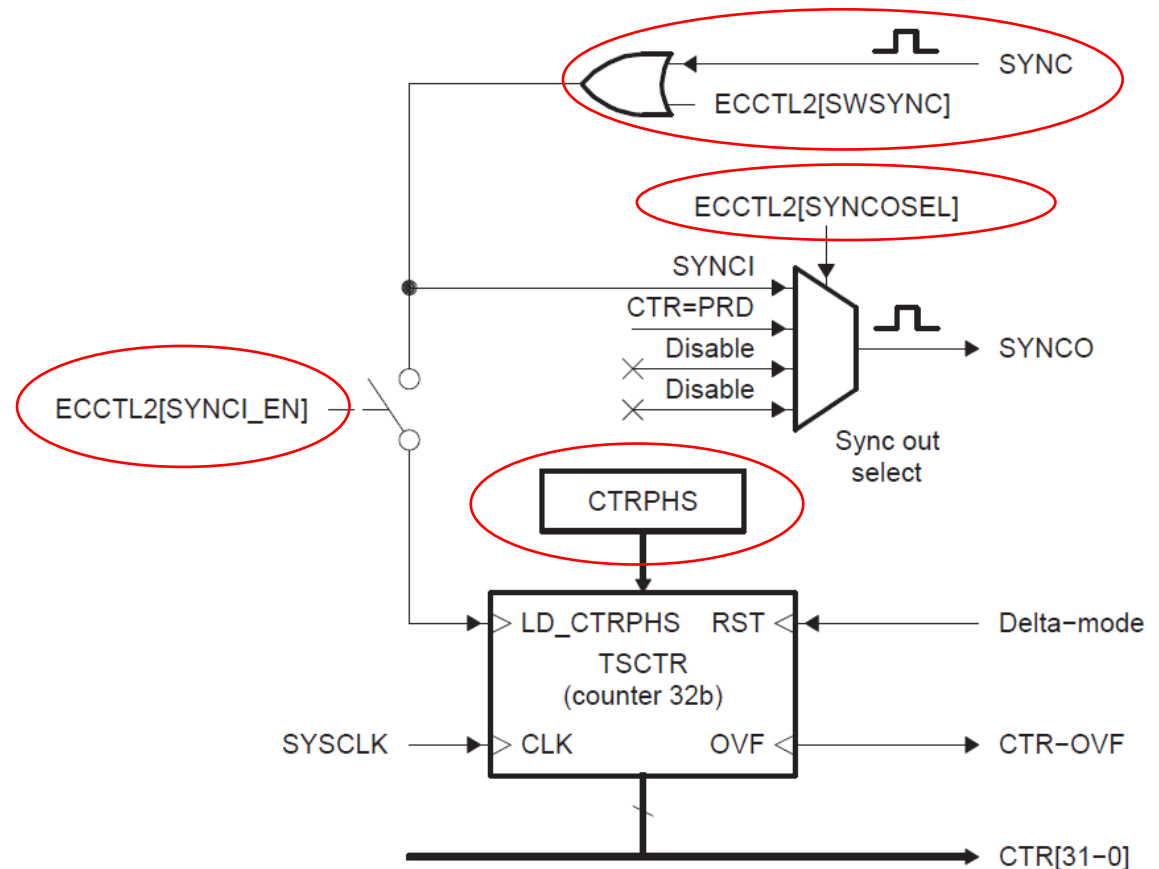
Software-forced  
Counter (TSCTR)  
Synchronizing

## SYNCO\_SEL

Sync-Out Select

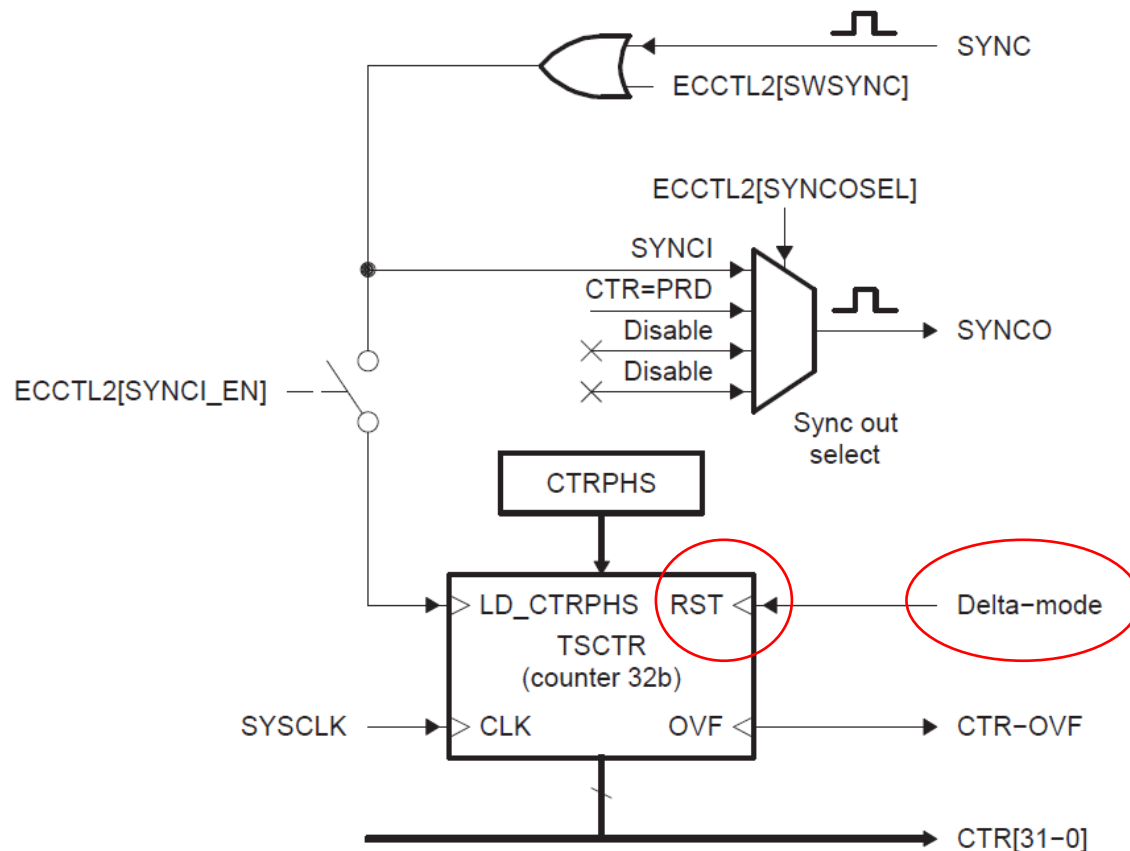
## SYNCI\_EN

Counter (TSCTR)  
Sync-In select  
mode



# 32位计数器与相位控制

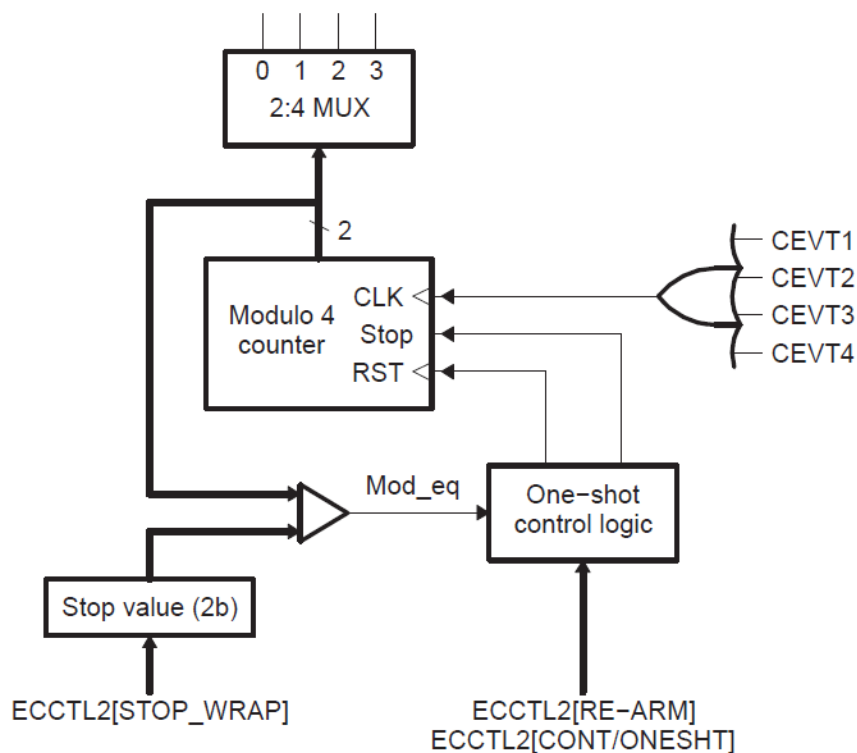
3. 4个装载事件（LD1-LD4）中的任何一个可以用来复位32位计数器，这项功能用来捕获信号边沿间的时间差。





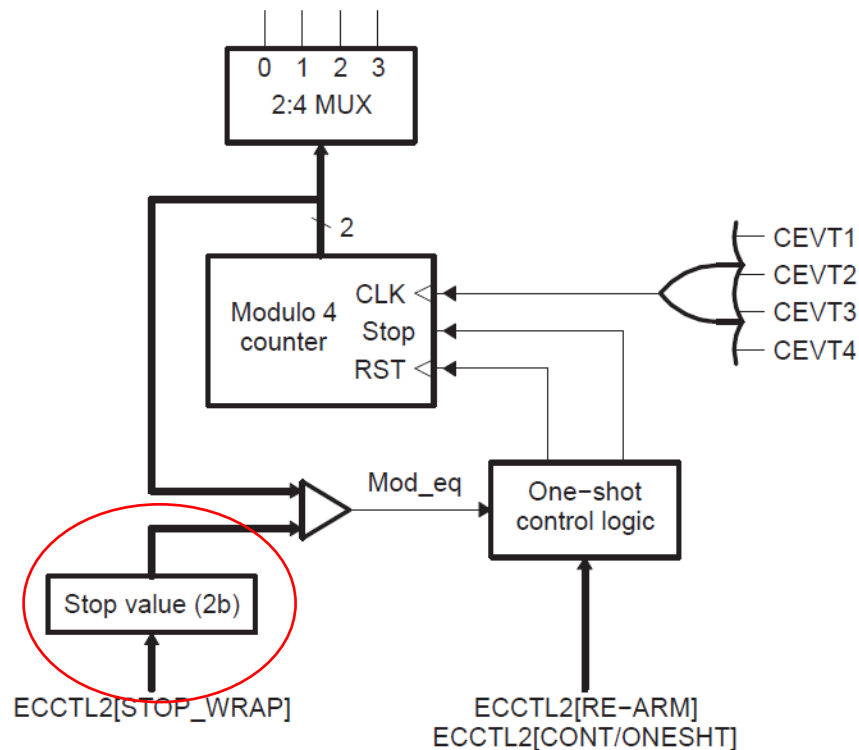
# 连续/单次捕获控制

1. 2位的Mod4计数器在边沿捕获事件（CEVT1-CEVT4）触发时  
递增计数；
2. Mod4计数器循环计数（0→1→2→3→0），直至停止工作；



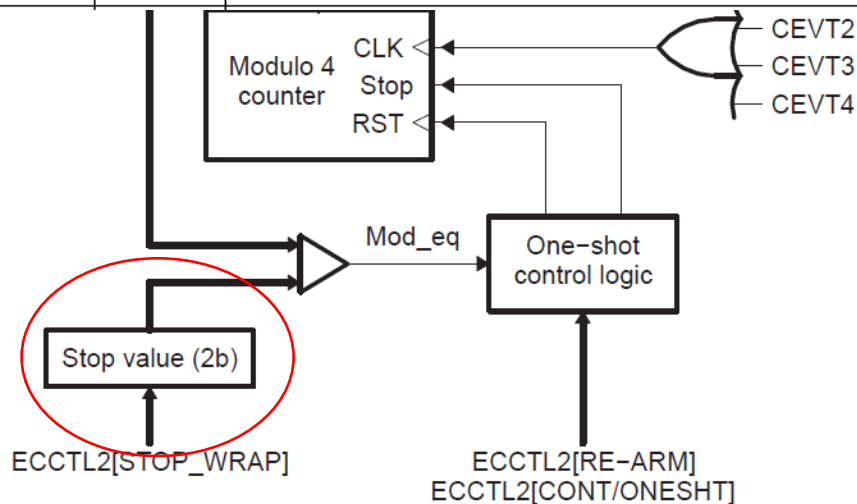
# 连续/单次捕获控制

3. 在**单次模式**下，一个**2位的停止寄存器**与Mod4计数器的输出值进行比较，如果等于停止寄存器的值，Mod4计数器将不再计数，并且阻止CAP1-CAP4寄存器加载数值；
4. 在**连续模式**下，Mod4计数器持续工作（0→1→2→3→0），捕获值按顺序不断的锁存到CAP1-CAP4。



# 连续/单次捕获控制

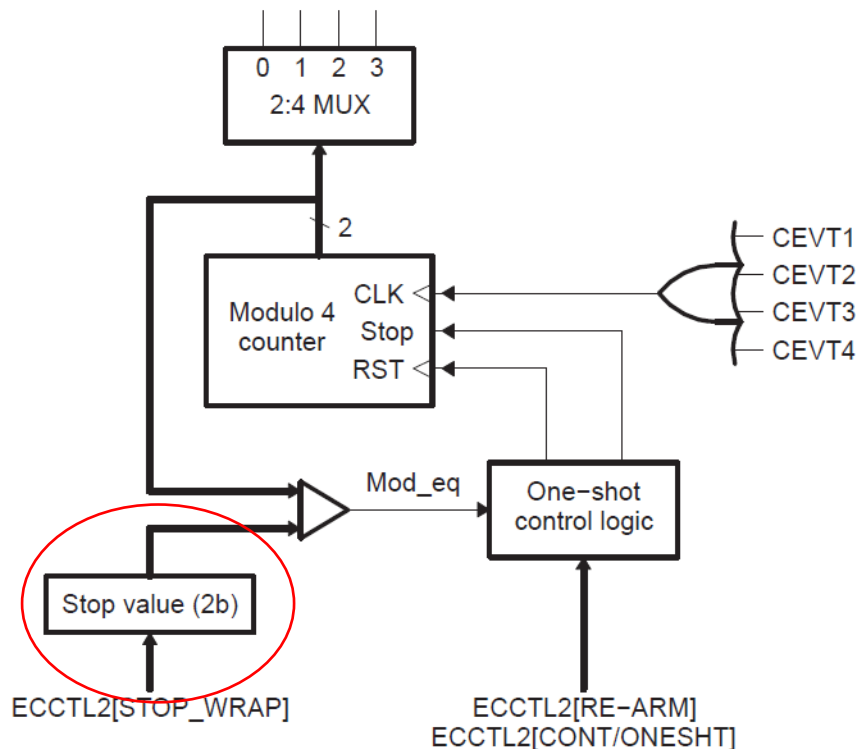
2:1	STOP_WRAP	<p>Stop value for one-shot mode. This is the number (between 1-4) of captures allowed to occur before the CAP(1-4) registers are frozen, i.e., capture sequence is stopped.</p> <p>Wrap value for continuous mode. This is the number (between 1-4) of the capture register in which the circular buffer wraps around and starts again.</p> <p>00 Stop after Capture Event 1 in one-shot mode Wrap after Capture Event 1 in continuous mode.</p> <p>01 Stop after Capture Event 2 in one-shot mode Wrap after Capture Event 2 in continuous mode.</p> <p>10 Stop after Capture Event 3 in one-shot mode Wrap after Capture Event 3 in continuous mode.</p> <p>11 Stop after Capture Event 4 in one-shot mode Wrap after Capture Event 4 in continuous mode.</p> <p>Notes: STOP_WRAP is compared to Mod4 counter and, when equal, 2 actions occur:</p> <ul style="list-style-type: none"> <li>• Mod4 counter is stopped (frozen)</li> <li>• Capture register loads are inhibited</li> </ul> <p>In one-shot mode, further interrupt events are blocked until re-armed.</p>
-----	-----------	--





# CAP1~CAP4寄存器

2. 通过CAPLDEN位可禁止装载功能，在单次模式下，当停止寄存器的值与Mod4计数器的值相等时，对CAPLDEN位清零。

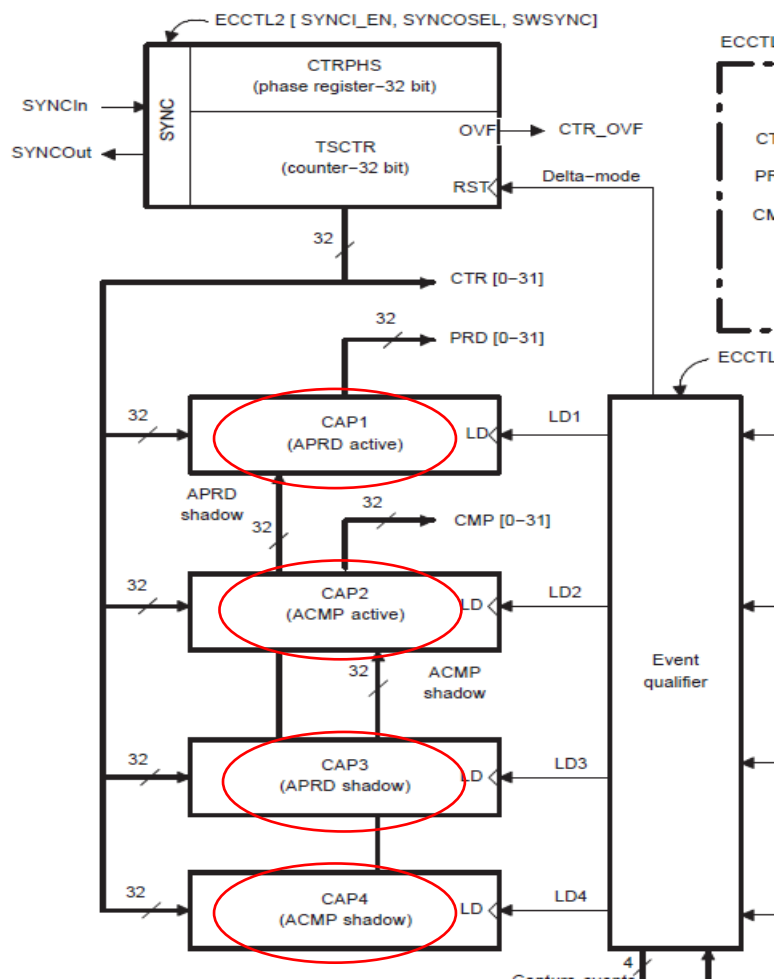


8	CAPLDEN		Enable Loading of CAP1-4 registers on a capture event
		0	Disable CAP1-4 register loads at capture event time.
		1	Enable CAP1-4 register loads at capture event time.



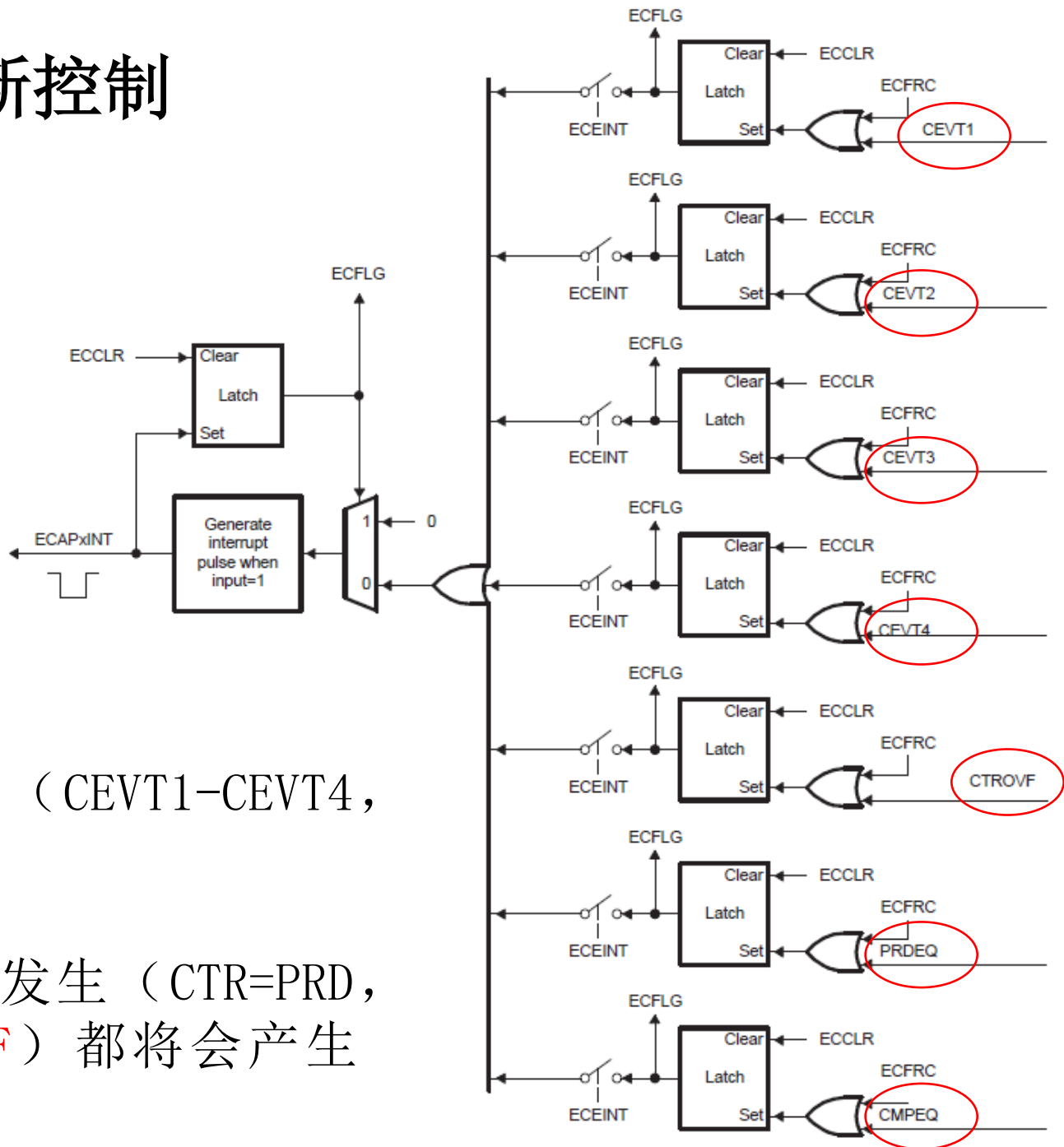
# CAP1~CAP4寄存器

3. 在APWM模式下，CAP1和CAP2分别为**有效的周期寄存器**和**有效的比较寄存器**。CAP3和CAP4分别为CAP1和CAP2对应的**影子寄存器**。



# eCAP中断控制


ECFRC、ECCLR、  
ECFLG、ECEINT  
寄存器



捕捉事件的发生（CEVT1-CEVT4，  
CTROVF）

或者APWM事件的发生（CTR=PRD，  
CTR=CMP，CTROVF）都将会产生  
中断请求。

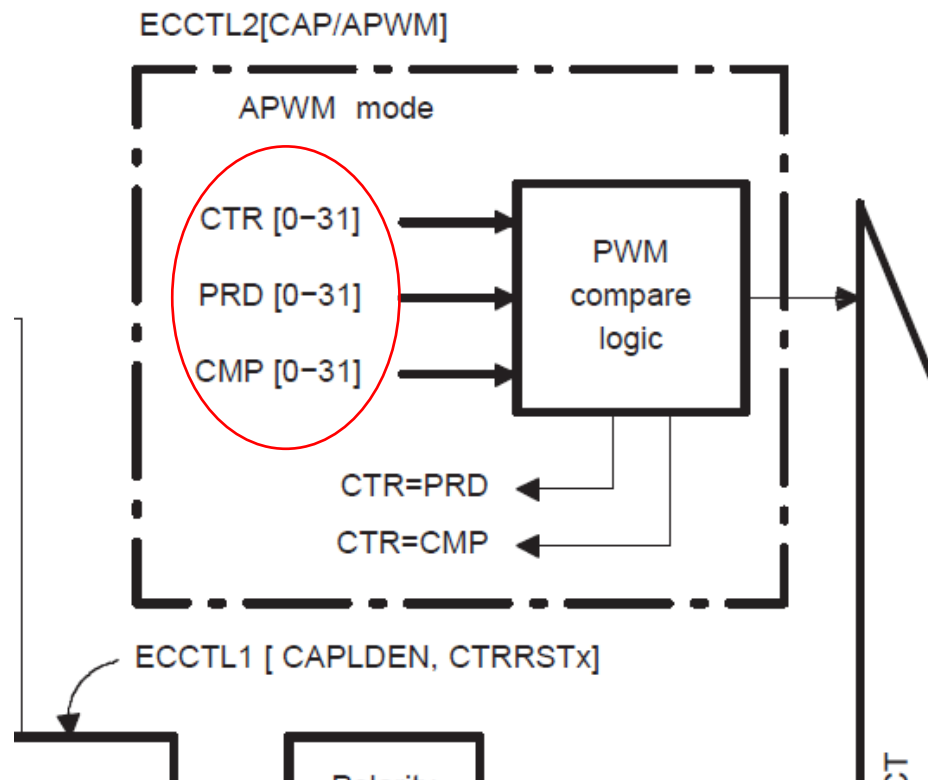
# 第九讲：增强型脉冲捕获模块eCAP

- 1、脉冲捕获基本原理
- 2、F28335增强型CAP
- 3、eCAP的捕获操作模式
-  4、eCAP的APWM操作模式
- 5、实用例子

# eCAP的APWM工作模式

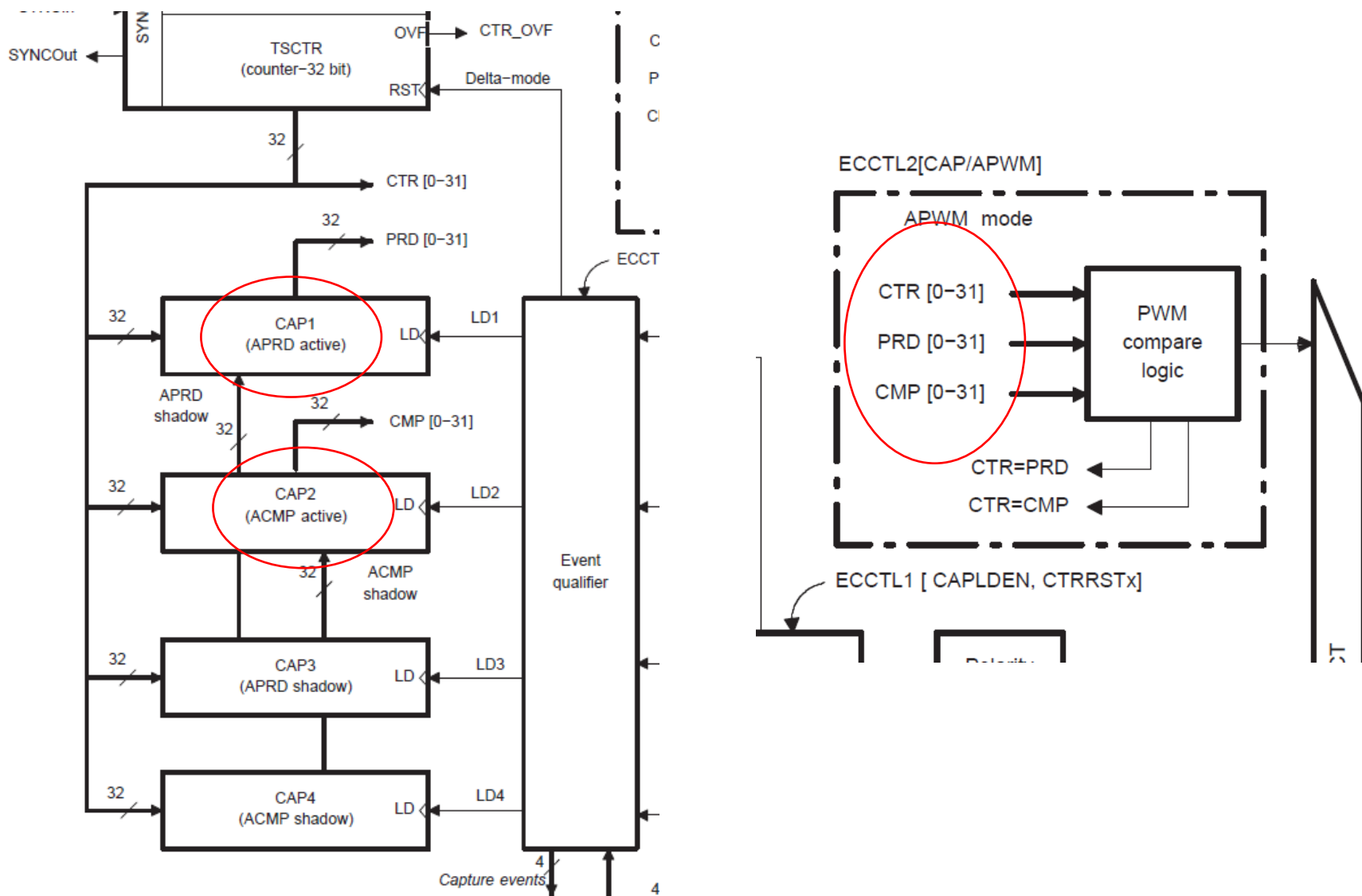
## APWM工作模式的主要特点

1. 时间标识计数器**TSCTR**与2个32位寄存器**APRD**和**ACMP**比较。



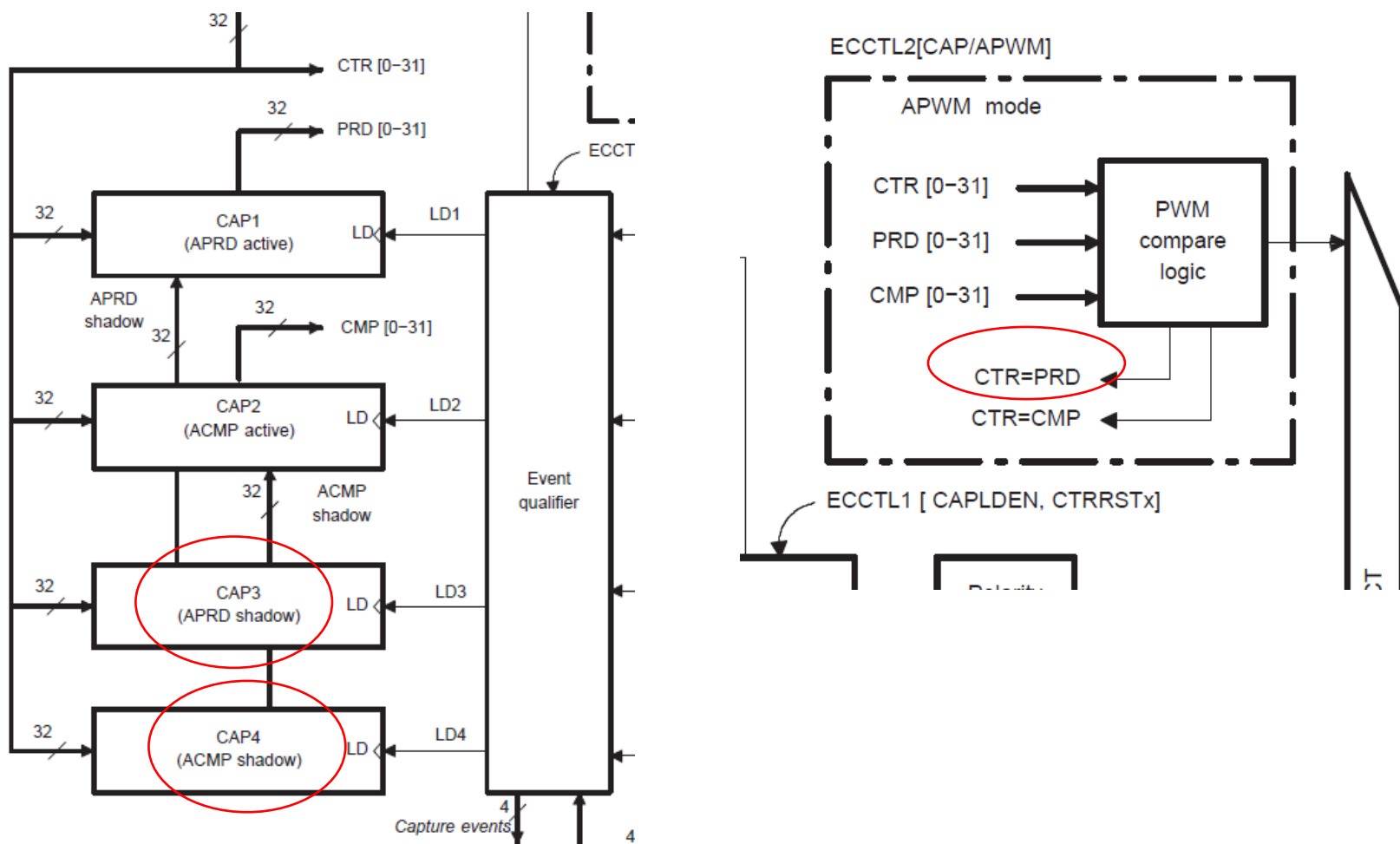
# APWM工作模式的主要特点

2. CAP1和CAP2用作周期和比较寄存器APRD和ACMP。



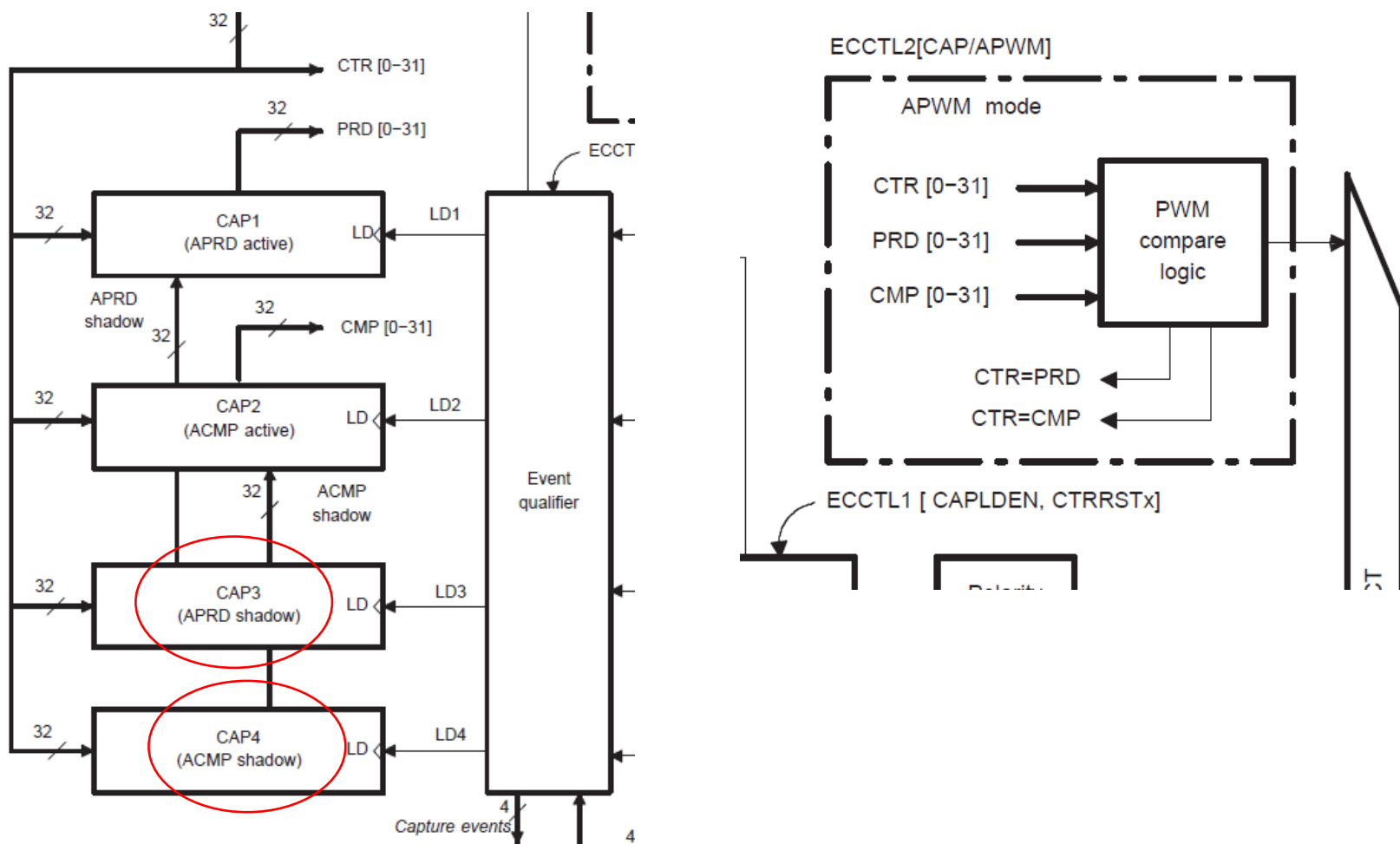
# APWM工作模式的主要特点

3. 通过影子寄存器APRD、ACMP（CAP3/4）实现双缓冲机制。可选择在对CAP3/4写操作或CTR=PRD两种情况下将影子寄存器的值装载到CAP1/2。



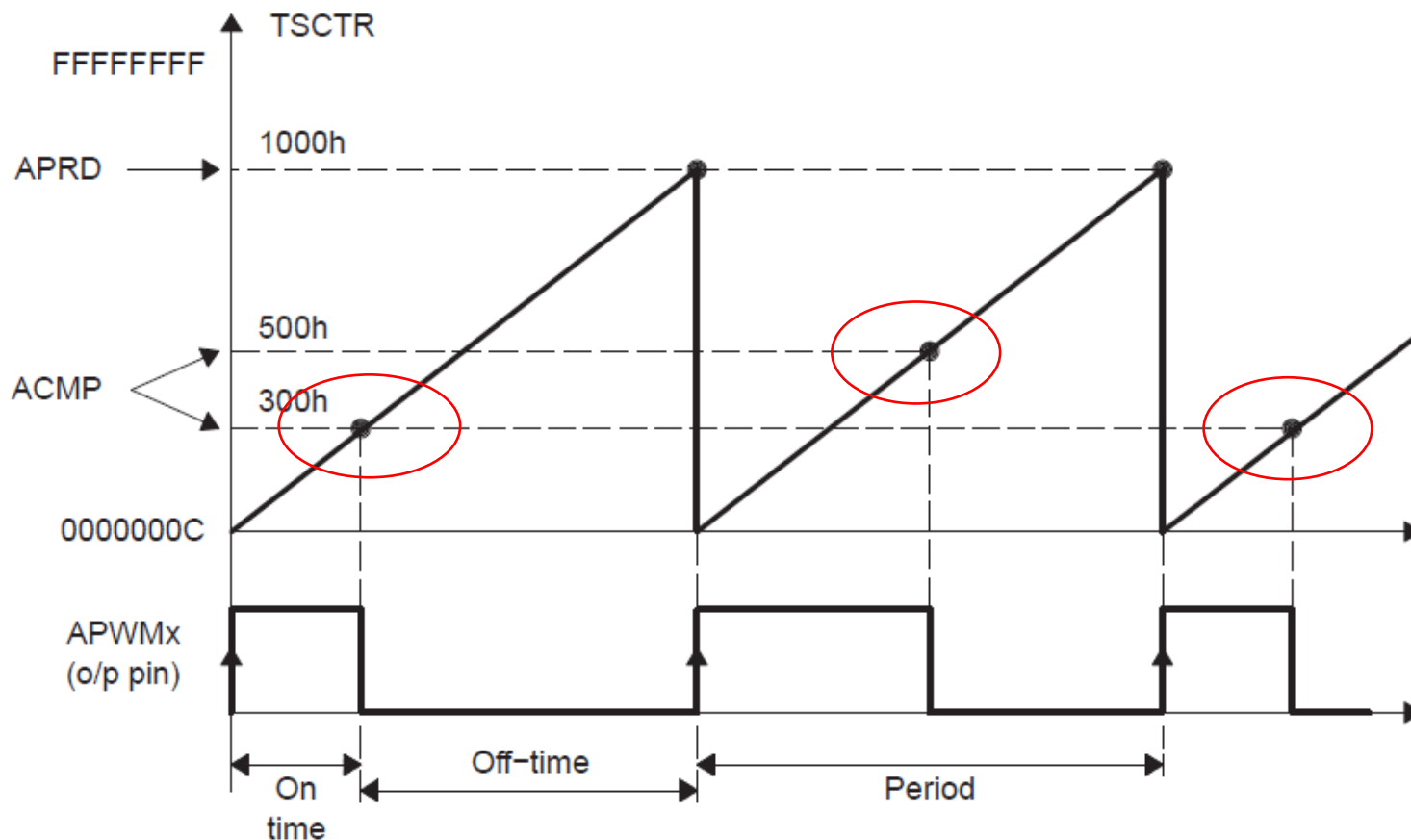
# APWM工作模式的主要特点

4. 在APWM模式下，对CAP1/2进行写操作，将把同样的值分别写入CAP3/4。对CAP3/4进行写操作将启动影子模式。



# APWM工作模式的主要特点

5. 在**初始化时**，必须先将周期和比较值写入CAP1/2，系统自动将同样的值分别写入CAP3/4。**运行时**只需改变CAP3/4的值来改变占空比。





# 第九讲：增强型脉冲捕获模块eCAP

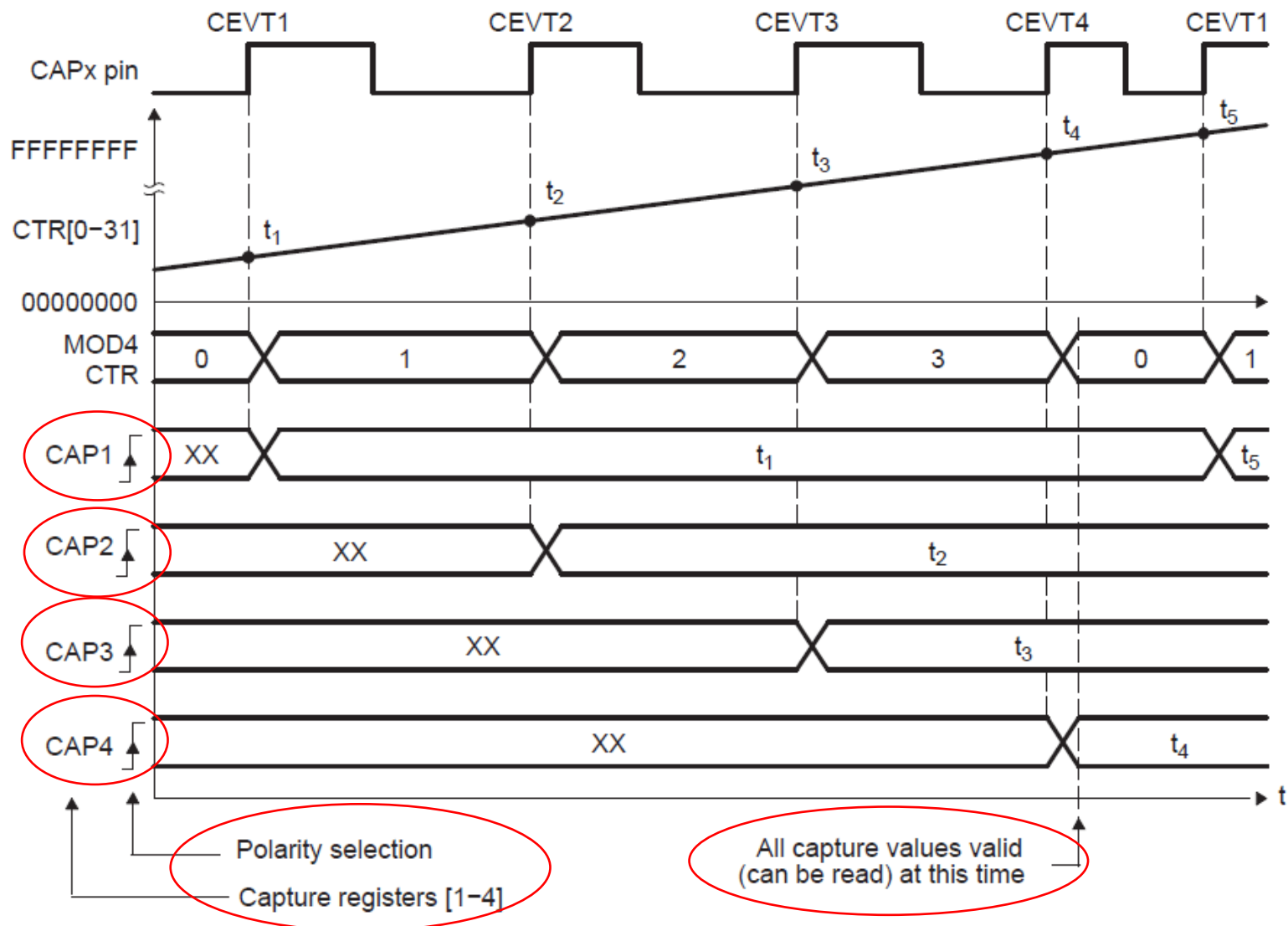
- 1、脉冲捕获基本原理
- 2、F28335增强型CAP
- 3、eCAP的捕获操作模式
- 4、eCAP的APWM操作模式



- 5、实用例子

# 捕获模式下绝对时间获取

## 1. 上升沿触发捕获事件



# 捕获模式下绝对时间获取

## 1. 上升沿触发捕获事件

```
// Code snippet for CAP mode Absolute Time, Rising edge trigger

// Initialization Time
//=====
// ECAP module 1 config
    ECap1Regs.ECCTL1.bit.CAP1POL = EC_RISING;
    ECap1Regs.ECCTL1.bit.CAP2POL = EC_RISING;
    ECap1Regs.ECCTL1.bit.CAP3POL = EC_RISING;
    ECap1Regs.ECCTL1.bit.CAP4POL = EC_RISING;
    ECap1Regs.ECCTL1.bit.CTRRST1 = EC_ABS_MODE;
    ECap1Regs.ECCTL1.bit.CTRRST2 = EC_ABS_MODE;
    ECap1Regs.ECCTL1.bit.CTRRST3 = EC_ABS_MODE;
    ECap1Regs.ECCTL1.bit.CTRRST4 = EC_ABS_MODE;
    ECap1Regs.ECCTL1.bit.CAPLDEN = EC_ENABLE;
    ECap1Regs.ECCTL1.bit.PRESCALE = EC_DIV1;

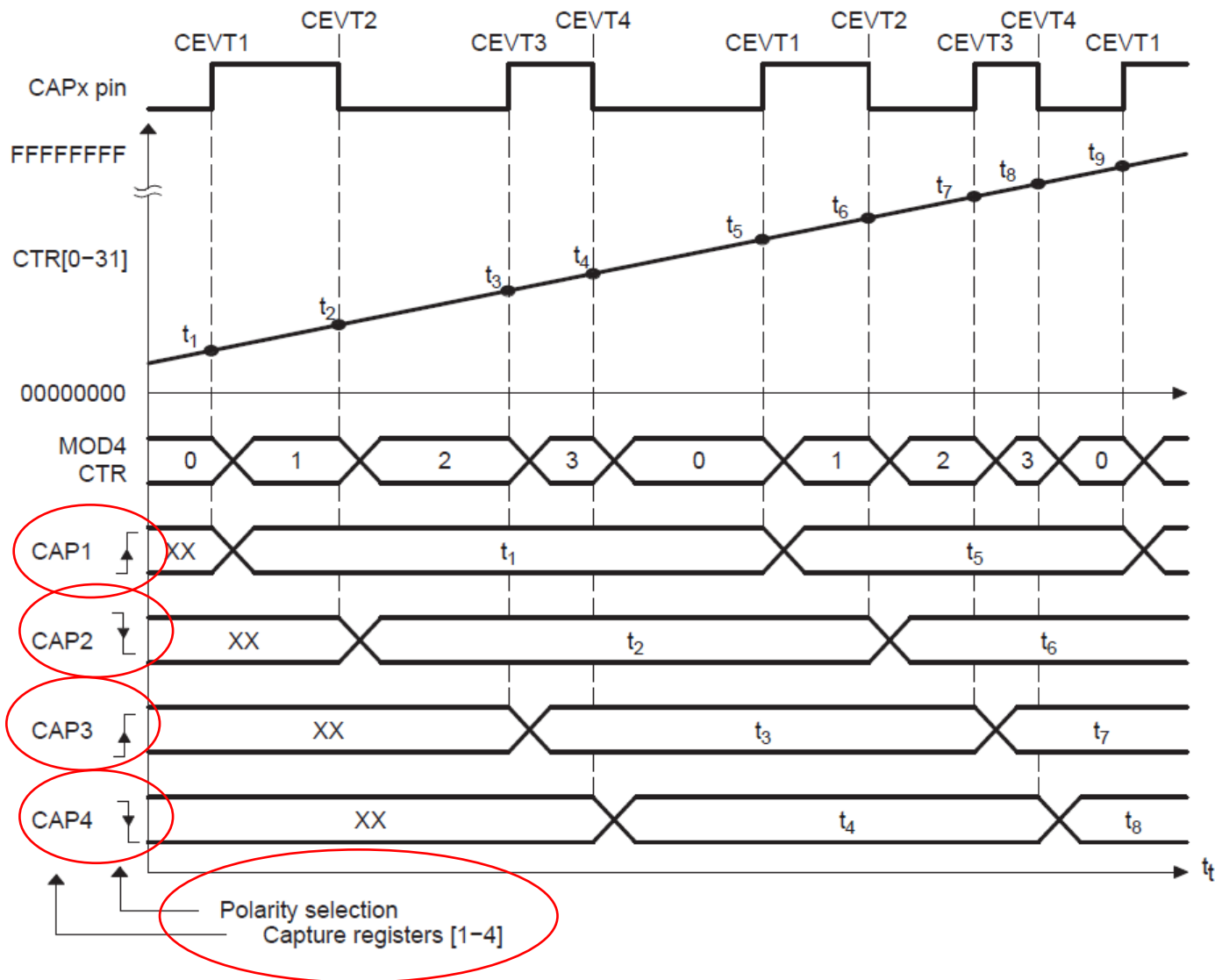
    ECap1Regs.ECCTL2.bit.CAP_APWM = EC_CAP_MODE;
    ECap1Regs.ECCTL2.bit.CONT_ONESHT = EC_CONTINUOUS;
    ECap1Regs.ECCTL2.bit.SYNCO_SEL = EC_SYNCO_DIS;
    ECap1Regs.ECCTL2.bit.SYNCI_EN = EC_DISABLE;
    ECap1Regs.ECCTL2.bit.TSCTRSTOP = EC_RUN;           // Allow TSCTR to run

// Run Time ( e.g. CEVT4 triggered ISR call)
//=====
    TSt1 = ECap1Regs.CAP1;           // Fetch Time-Stamp captured at t1
    TSt2 = ECap1Regs.CAP2;           // Fetch Time-Stamp captured at t2
    TSt3 = ECap1Regs.CAP3;           // Fetch Time-Stamp captured at t3
    TSt4 = ECap1Regs.CAP4;           // Fetch Time-Stamp captured at t4

    Period1 = TSt2-TSt1;             // Calculate 1st period
    Period2 = TSt3-TSt2;             // Calculate 2nd period
    Period3 = TSt4-TSt3;             // Calculate 3rd period
```

# 捕获模式下绝对时间获取

## 2. 上升沿和下降沿都触发捕获事件



# 捕获模式下绝对时间获取

## 2. 上升沿和下降沿都触发捕获事件

```
// Code snippet for CAP mode Absolute Time, Rising & Falling
// edge triggers

// Initialization Time
//=====
// ECAP module 1 config
ECap1Regs.ECCTL1.bit.CAP1POL = EC_RISING;
ECap1Regs.ECCTL1.bit.CAP2POL = EC_FALLING;
ECap1Regs.ECCTL1.bit.CAP3POL = EC_RISING;
ECap1Regs.ECCTL1.bit.CAP4POL = EC_FALLING;
ECap1Regs.ECCTL1.bit.CTRRST1 = EC_ABS_MODE;
ECap1Regs.ECCTL1.bit.CTRRST2 = EC_ABS_MODE;
ECap1Regs.ECCTL1.bit.CTRRST3 = EC_ABS_MODE;
ECap1Regs.ECCTL1.bit.CTRRST4 = EC_ABS_MODE;
ECap1Regs.ECCTL1.bit.CAPLDEN = EC_ENABLE;
ECap1Regs.ECCTL1.bit.PRESCALE = EC_DIV1;

ECap1Regs.ECCTL2.bit.CAP_APWM = EC_CAP_MODE;
ECap1Regs.ECCTL2.bit.CONT_ONESHT = EC_CONTINUOUS;
ECap1Regs.ECCTL2.bit.SYNCO_SEL = EC_SYNCO_DIS;
ECap1Regs.ECCTL2.bit.SYNCI_EN = EC_DISABLE;
ECap1Regs.ECCTL2.bit.TSCTRSTOP = EC_RUN;           // Allow TSCTR to run

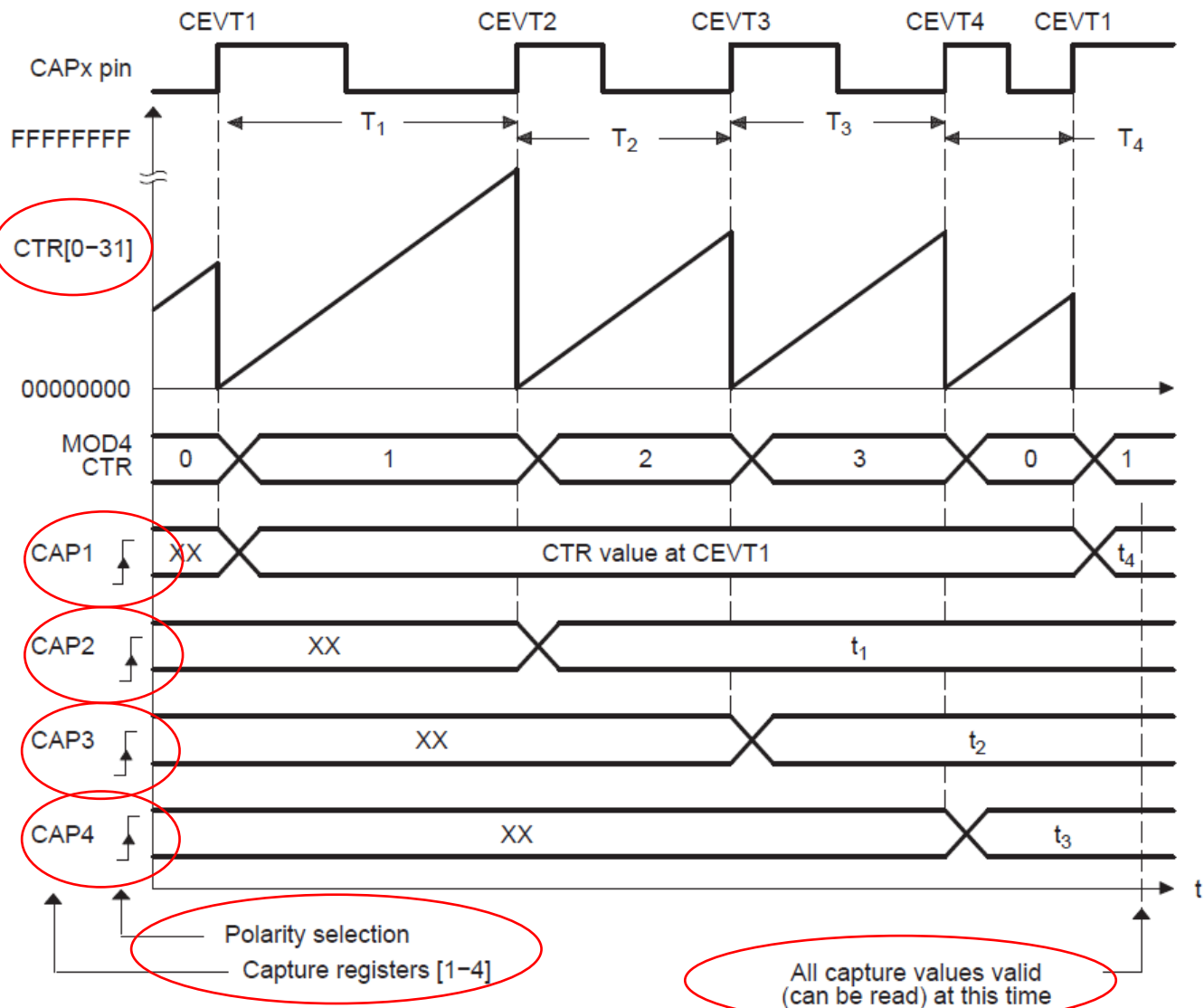
// Run Time ( e.g. CEVT4 triggered ISR call)
//=====
TSt1 = ECap1Regs.CAP1;           // Fetch Time-Stamp captured at t1
TSt2 = ECap1Regs.CAP2;           // Fetch Time-Stamp captured at t2
TSt3 = ECap1Regs.CAP3;           // Fetch Time-Stamp captured at t3
TSt4 = ECap1Regs.CAP4;           // Fetch Time-Stamp captured at t4

Period1 = TSt3-TSt1;             // Calculate 1st period
DutyOnTime1 = TSt2-TSt1;         // Calculate On time
DutyOffTime1 = TSt3-TSt2;        // Calculate Off time
```

# 捕获模式下差分时间 (delta) 获取

## 1. 上升沿触发捕获事件

TSCTR的  
delta-mode  
复位



# 捕获模式下差分时间（delta）获取

## 1. 上升沿触发捕获事件

```
// Code snippet for CAP mode Delta Time, Rising edge trigger

// Initialization Time
//=====
// ECAP module 1 config
    ECap1Regs.ECCTL1.bit.CAP1POL = EC_RISING;
    ECap1Regs.ECCTL1.bit.CAP2POL = EC_RISING;
    ECap1Regs.ECCTL1.bit.CAP3POL = EC_RISING;
    ECap1Regs.ECCTL1.bit.CAP4POL = EC_RISING;
    ECap1Regs.ECCTL1.bit.CTRRST1 = EC_DELTA_MODE;
    ECap1Regs.ECCTL1.bit.CTRRST2 = EC_DELTA_MODE;
    ECap1Regs.ECCTL1.bit.CTRRST3 = EC_DELTA_MODE;
    ECap1Regs.ECCTL1.bit.CTRRST4 = EC_DELTA_MODE;
    ECap1Regs.ECCTL1.bit.CAPLDEN = EC_ENABLE;
    ECap1Regs.ECCTL1.bit.PRESCALE = EC_DIV1;

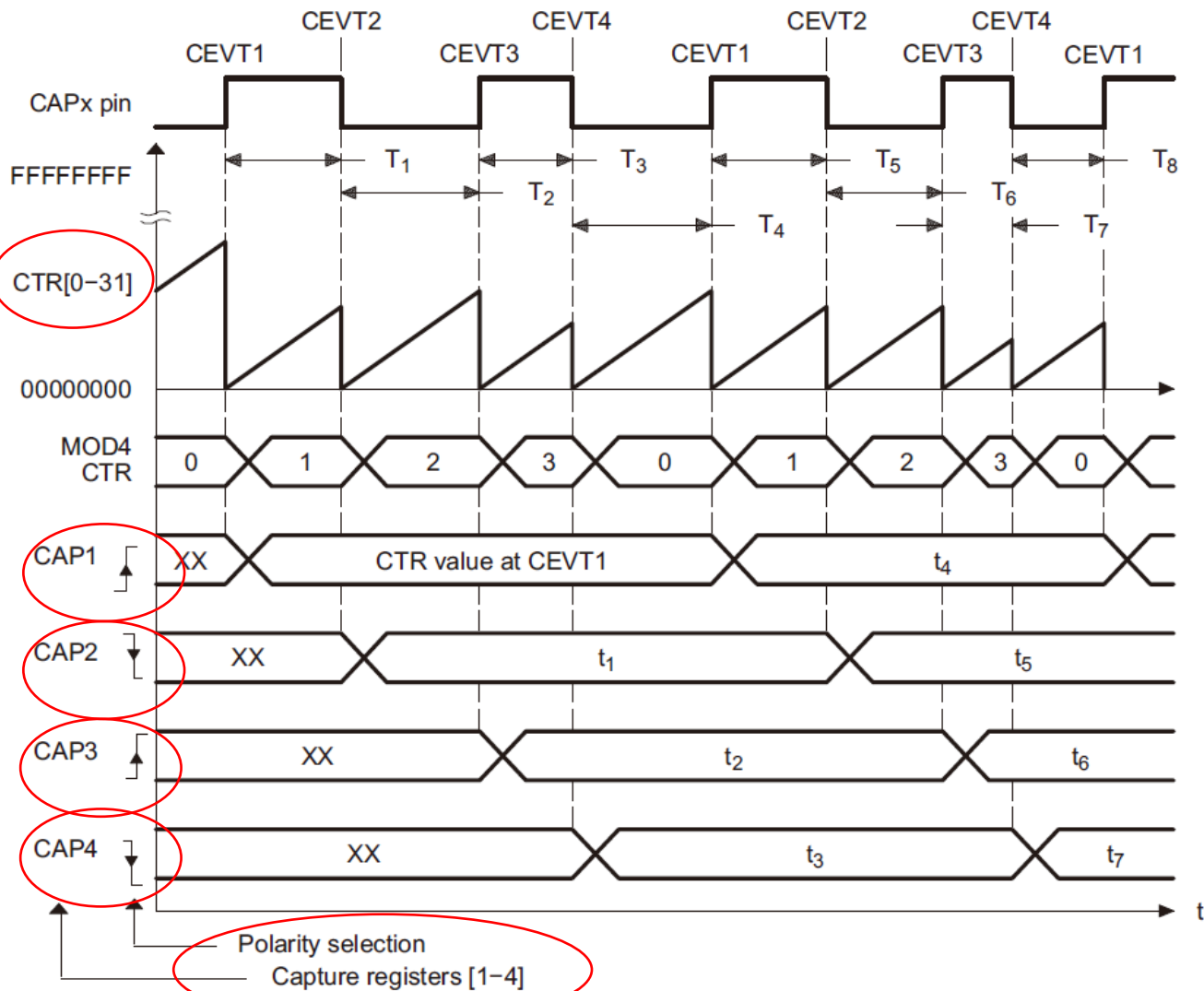
    ECap1Regs.ECCTL2.bit.CAP_APWM = EC_CAP_MODE;
    ECap1Regs.ECCTL2.bit.CONT_ONESHT = EC_CONTINUOUS;
    ECap1Regs.ECCTL2.bit.SYNCO_SEL = EC_SYNCO_DIS;
    ECap1Regs.ECCTL2.bit.SYNCI_EN = EC_DISABLE;
    ECap1Regs.ECCTL2.bit.TSCTRSTOP = EC_RUN;           // Allow TSCTR to run

// Run Time ( e.g. CEVT1 triggered ISR call)
//=====
// Note: here Time-stamp directly represents the Period value.
    Period4 = ECap1Regs.CAP1;    // Fetch Time-Stamp captured at T1
    Period1 = ECap1Regs.CAP2;    // Fetch Time-Stamp captured at T2
    Period2 = ECap1Regs.CAP3;    // Fetch Time-Stamp captured at T3
    Period3 = ECap1Regs.CAP4;    // Fetch Time-Stamp captured at T4
```

# 捕获模式下差分时间 (delta) 获取

## 2. 上升沿和下降沿都触发捕获事件

TSCTR的  
delta-mode  
复位





# 捕获模式下差分时间（delta）获取

## 2. 上升沿和下降沿都触发捕获事件

```
// Code snippet for CAP mode Delta Time, Rising and Falling
// edge triggers

// Initialization Time
//=====
// ECAP module 1 config
ECap1Regs.ECCTL1.bit.CAP1POL = EC_RISING;
ECap1Regs.ECCTL1.bit.CAP2POL = EC_FALLING;
ECap1Regs.ECCTL1.bit.CAP3POL = EC_RISING;
ECap1Regs.ECCTL1.bit.CAP4POL = EC_FALLING;
ECap1Regs.ECCTL1.bit.CTRRST1 = EC_DELTA_MODE;
ECap1Regs.ECCTL1.bit.CTRRST2 = EC_DELTA_MODE;
ECap1Regs.ECCTL1.bit.CTRRST3 = EC_DELTA_MODE;
ECap1Regs.ECCTL1.bit.CTRRST4 = EC_DELTA_MODE;
ECap1Regs.ECCTL1.bit.CAPLDEN = EC_ENABLE;
ECap1Regs.ECCTL1.bit.PRESCALE = EC_DIV1;

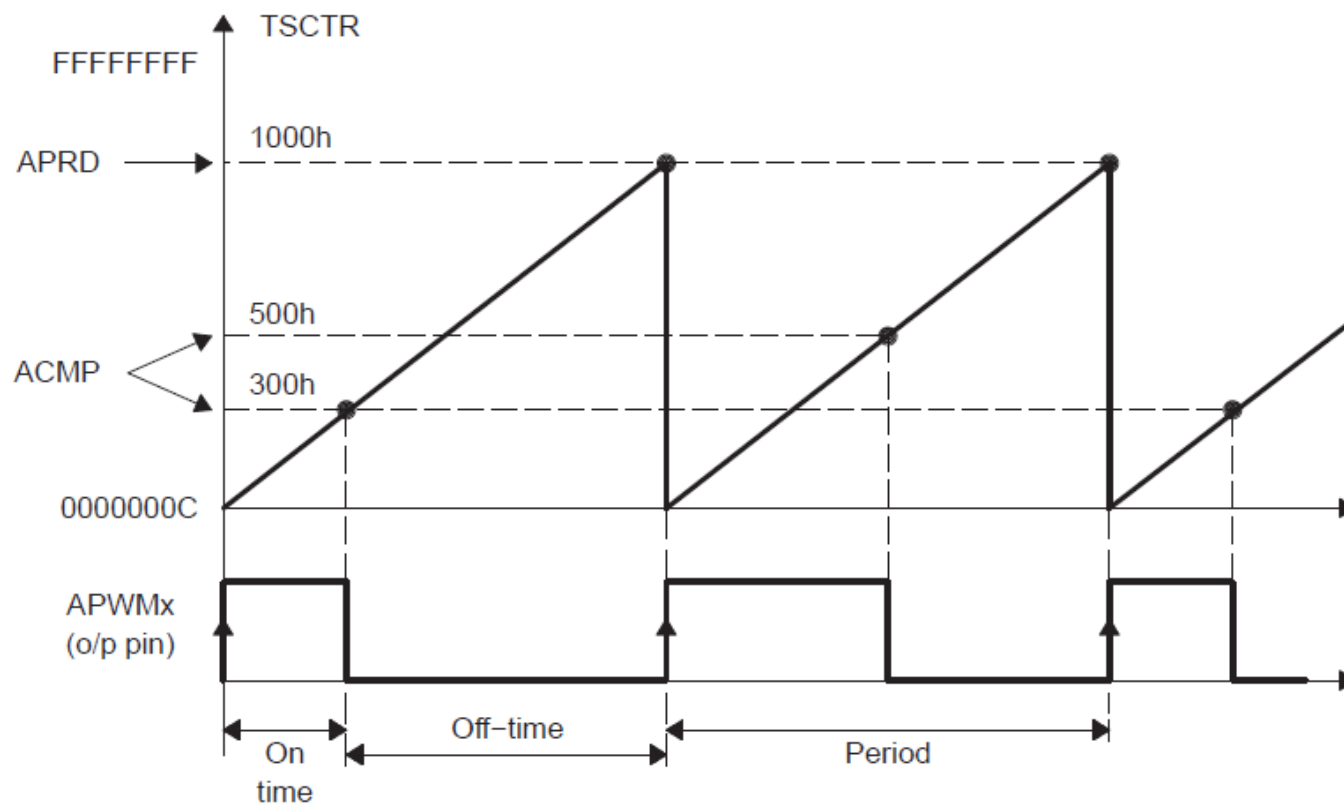
ECap1Regs.ECCTL2.bit.CAP_APWM = EC_CAP_MODE;
ECap1Regs.ECCTL2.bit.CONT_ONESHT = EC_CONTINUOUS;
ECap1Regs.ECCTL2.bit.SYNCO_SEL = EC_SYNCO_DIS;
ECap1Regs.ECCTL2.bit.SYNCI_EN = EC_DISABLE;
ECap1Regs.ECCTL2.bit.TSCTRSTOP = EC_RUN;           // Allow TSCTR to run

// Run Time ( e.g. CEVT1 triggered ISR call)
//=====
// Note: here Time-stamp directly represents the Duty cycle values.
DutyOnTime1 = ECap1Regs.CAP2;    // Fetch Time-Stamp captured at T2
DutyOffTime1 = ECap1Regs.CAP3;   // Fetch Time-Stamp captured at T3
DutyOnTime2 = ECap1Regs.CAP4;    // Fetch Time-Stamp captured at T4
DutyOffTime2 = ECap1Regs.CAP1;   // Fetch Time-Stamp captured at T1

Period1 = DutyOnTime1 + DutyOffTime1;
Period2 = DutyOnTime2 + DutyOffTime2;
```

# APWM模式应用

## 1. 产生单路PWM信号



# APWM模式应用

## 1. 产生单路PWM信号

```
// Code snippet for APWM mode Example 1

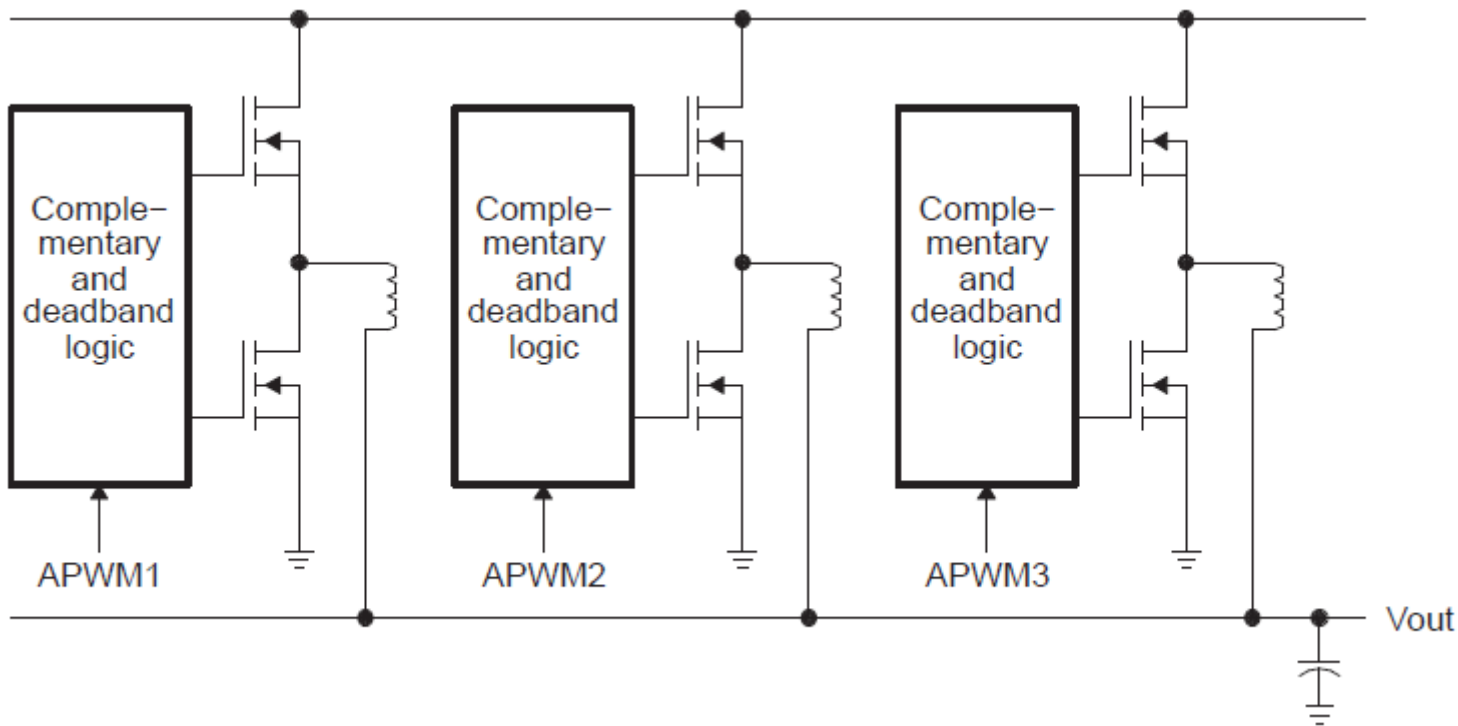
// Initialization Time
//=====
// ECAP module 1 config
ECap1Regs.CAP1 = 0x1000;           // Set period value
ECap1Regs.CTRPHS = 0x0;           // make phase zero
ECap1Regs.ECCTL2.bit.CAP_APWM = EC_APWM_MODE;
ECap1Regs.ECCTL2.bit.APWMPOL = EC_ACTV_HI; // Active high
ECap1Regs.ECCTL2.bit.SYNCI_EN = EC_DISABLE; // Synch not used
ECap1Regs.ECCTL2.bit.SYNCO_SEL = EC_SYNCO_DIS; // Synch not used
ECap1Regs.ECCTL2.bit.TSCTRSTOP = EC_RUN; // Allow TSCTR to run

// Run Time (Instant 1, e.g. ISR call)
//=====
ECap1Regs.CAP2 = 0x300; // Set Duty cycle i.e. compare value

// Run Time (Instant 2, e.g. another ISR call)
//=====
ECap1Regs.CAP2 = 0x500; // Set Duty cycle i.e. compare value
```

# APWM模式应用

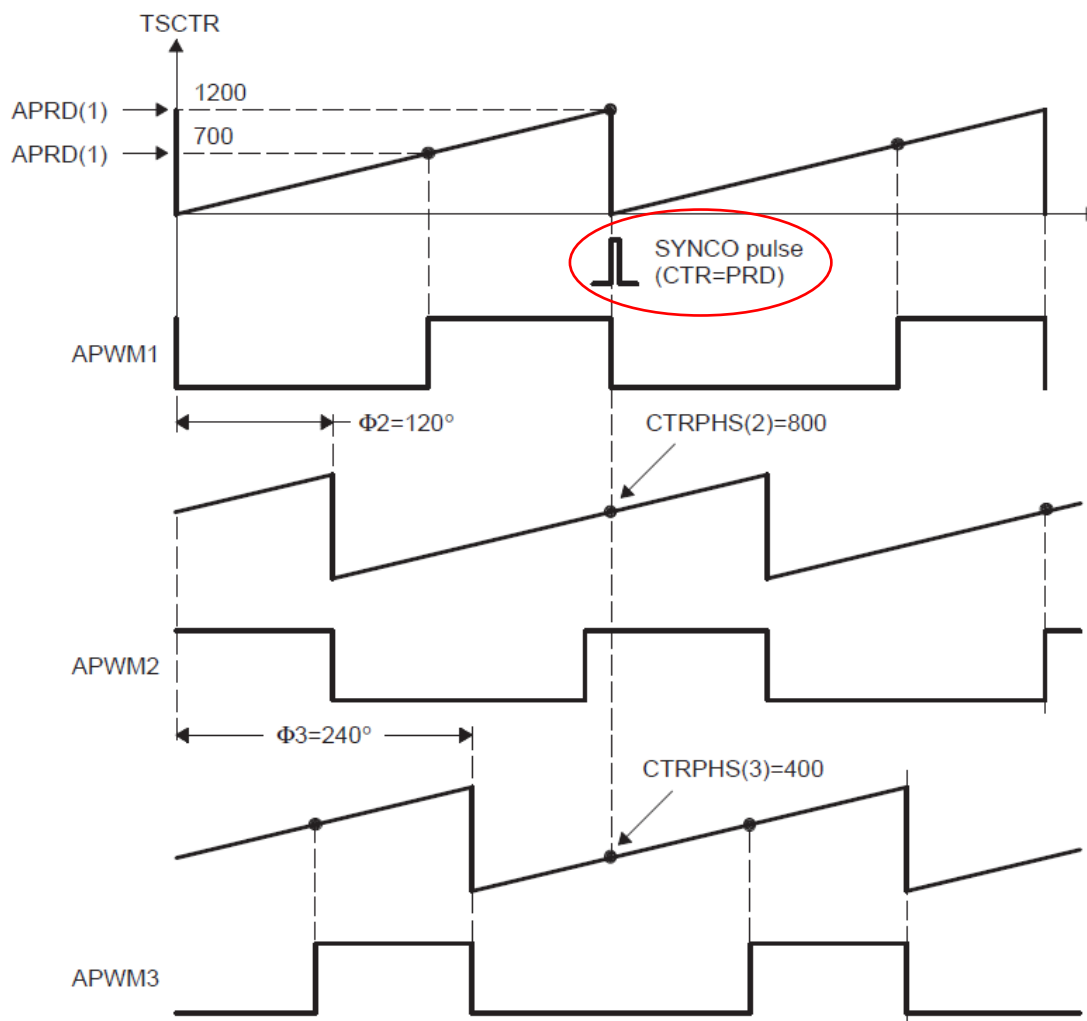
## 2. 利用相位寄存器产生多路PWM信号



3 phase Interleaved DC/DC converter topology

# APWM模式应用

## 2. 利用相位寄存器产生多路PWM信号



# APWM模式应用

## 2. 利用相位寄存器产生多路PWM信号

```
// Code snippet for APWM mode Example 2

// Initialization Time
//=====
// ECAP module 1 config
ECap1Regs.ECCTL2.bit.CAP_APWM = EC_APWM_MODE;
ECap1Regs.CAP1 = 1200; // Set period value
ECap1Regs.CTRPHS = 0; // make eCAP1 reference phase = zero
ECap1Regs.ECCTL2.bit.APWMPOL = EC_ACTV_HI;
ECap1Regs.ECCTL2.bit.SYNCI_EN = EC_DISABLE; // No sync in for Master
ECap1Regs.ECCTL2.bit.SYNCO_SEL = EC_CTR_PRD; // eCAP1 is Master
ECap1Regs.ECCTL2.bit.TSCTRSTOP = EC_RUN; // Allow TSCTR to run

// ECAP module 2 config
ECap2Regs.CAP1 = 1200; // Set period value
ECap2Regs.CTRPHS = 800; // Phase offset = 1200-400 = 120 deg
ECap2Regs.ECCTL2.bit.CAP_APWM = EC_APWM_MODE;
ECap2Regs.ECCTL2.bit.APWMPOL = EC_ACTV_HI;
ECap2Regs.ECCTL2.bit.SYNCI_EN = EC_ENABLE; // slaved off master
ECap2Regs.ECCTL2.bit.SYNCO_SEL = EC_SYNCI; // sync "flow-through"
ECap2Regs.ECCTL2.bit.TSCTRSTOP = EC_RUN; // Allow TSCTR to run

// ECAP module 3 config
ECap3Regs.CAP1 = 1200; // Set period value
ECap3Regs.CTRPHS = 400; // Phase offset = 1200-800 = 240 deg
ECap3Regs.ECCTL2.bit.CAP_APWM = EC_APWM_MODE;
ECap3Regs.ECCTL2.bit.APWMPOL = EC_ACTV_HI;
ECap3Regs.ECCTL2.bit.SYNCI_EN = EC_ENABLE; // slaved off master
ECap3Regs.ECCTL2.bit.SYNCO_SEL = EC_SYNCO_DIS; // "break the chain"
ECap3Regs.ECCTL2.bit.TSCTRSTOP = EC_RUN; // Allow TSCTR to run

// Run Time (Note: Example execution of one run-time instant)
//=====
// All phases are set to the same duty cycle
ECap1Regs.CAP2 = 700; // Set Duty cycle i.e. compare value = 700
ECap2Regs.CAP2 = 700; // Set Duty cycle i.e. compare value = 700
ECap3Regs.CAP2 = 700; // Set Duty cycle i.e. compare value = 700
```

谢谢