MSP430 单片机简介

Andy Scout

September 14, 2012

简介

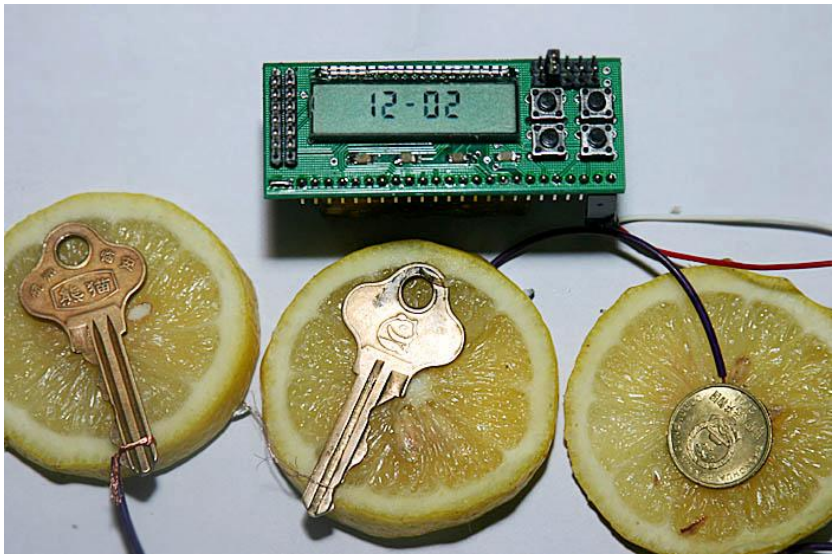
MSP430 单片机是美国德州仪器 (TI) 公司 1996 年开始推向市场的一种 16 位超低功耗、具有精简指令集的混合信号处理器。它将多个不同功能的模拟电路、数字电路模块和微处理器集成在了一个芯片上。

特点

1. 处理能力强
2. 运算速度快
3. 超低功耗
4. 片内资源丰富
5. 方便高效的开发环境

超低功耗

图: 用水果电池供电的某 430 单片机系统



超低功耗

操作模式	说明	CPU (MCLK)	SMCLK	ACLK	RAM 保持	BOR	自动唤醒	中断源
运行状态	CPU、所有时钟与外设均可用。	•	•	•	•	•		定时器、ADC、DMA、UART、WDT、I/0、比较器、外部中断、RTC、串行通信、其他外设
LPM0	CPU 关断，外设时钟可用。		•	•	•	•	•	定时器、ADC、DMA、UART、WDT、I/0、比较器、外部中断、RTC、串行通信、其他外设
LPM1	CPU 关断，外设时钟可用。DCO 被停用，而且 DC 发生器可以禁用。		•	•	•	•	•	定时器、ADC、DMA、UART、WDT、I/0、比较器、外部中断、RTC、串行通信、其他外设
LPM2	CPU 关断，只有一个外设时钟可用。DC 发生器被启用。			•	•	•	•	定时器、ADC、DMA、UART、WDT、I/0、比较器、外部中断、RTC、串行通信、其他外设
LPM3	CPU 关断，只有一个外设时钟可用。DC 发生器被停用。			•	•	•	•	定时器、ADC、DMA、UART、WDT、I/0、比较器、外部中断、RTC、串行通信、其他外设
LPM3.5	无 RAM 保持，RTC 可以启用。（仅限 MSP430F5xx 系列）					•	•	外部中断、RTC
LPM4	CPU 关断，而且所有时钟均被停用。				•	•		外部中断
LPM4.5	无 RAM 保持，RTC 被停用。（仅限 MSP430F5xx 系列）					•		外部中断

时钟系统

MSP430 具有独特的时钟系统，
时钟源主要可分为：[1]

- LFXT1CLK 低频时钟源
- XT2CLK 高频时钟源
- DCOCLK 数字控制 RC 振荡器

时钟输出信号可以分为：

- ACLK 辅助时钟
- MCLK 主系统时钟
- SMCLK 子系统时钟

▸ 时钟源配置示例

[1] MSP430X2XX 系列还有 VLOCLK 时钟源

端口

类型丰富

P1, P2
P3, P4, P5, P6
S 和 COM

功能丰富

I/O
中断能力
其他片内外设功能
驱动液晶

寄存器丰富

各端口共有的寄存器：

- PxDIR I/O 方向寄存器
- PxIN 输入寄存器
- PxOUT 输出寄存器
- PxSEL 功能选择寄存器

P1 和 P2 具有中断功能，还有以下几个与中断有关的寄存器

- PxIE 中断使能寄存器
- PxIFG 中断标志寄存器
- PxIES 中断触发沿选择寄存器

部分芯片的端口还有 PxREN 内部上拉/下拉电阻使能寄存器。

▸ 端口程序示例

定时器

MSP430 主要有一下几类定时器：

1. 看门狗定时器 WDT
2. 基本定时器 BT
3. 定时器 A TimerA
4. 定时器 B TimerB
5. 实时时钟 RTC

看门狗定时器

看门狗一般是防止单片机程序跑飞，检测系统运行的功能。MSP430 的看门狗定时器不仅包含这个功能，他也可以作为内部定时器和其他定时器一样产生定时中断。

看门狗定时器可以工作在以下三种状态：

1. **看门狗模式**
   在这种模式下，如果计数器超过了定时时间，就会产生复位和系统上电清除 (PUC) 信号。看门狗模式下的中断是不可屏蔽的，由受控程序不正常运行引发。
2. **定时器模式**
   在这种模式下，将产生时间的周期性中断。定时器中断是可屏蔽的。

   注意事项：
   - 改变定时时间不同时清除 WDTCNT 将导致不可预测的系统立即复位或中断。定时器时间改变和计数器清除必须在一条指令中完成。
   - 如果先后分别进行清除和改变定时时间，可能立即引起不可预料的系统复位或中断。
   - 在正常工作时，改变时钟源，可能导致 WDTCNT 额外的计数时钟。
3. **低功耗模式**
   当系统不需要 WDT 做看门狗或定时器时，可以关闭 WDT 节约功耗。
   WDTCTL = WDTPW + WDTHOLD;

MSP430 看门狗有两个相关的寄存器：
WDTCNT(计数寄存器) 和 WDTCTL(控制寄存器)。
WDTCNT 不能直接通过软件存取，必须通过 WDTCTL 来控制。
WDTCTL 由两部分组成，高 8 位作为口令，低 8 位是对 WDT 操作的控制命令。高八位必须写入 5AH(宏定义为 WDTPW)，如果写错，系统将复位。

▸ 看门狗示例

定时器 A

定时器 A 具有以下特性：

- 输入时钟有多种选择, 可以是慢时钟、快时钟以及外部时钟。
- 虽然没有自动重载时间常数功能，但产生的定时脉冲或 PWM 信号没有软件带来的误差。
- 不仅能捕获外部事件发生的时间，还可以锁定发生时的电平高低。
- 可实现串行通讯。
- 完善的中断服务功能。
- 4 种计数功能选择。
- 8 种输出方式选择。
- 支持多时序控制。
- DMA 使能。

TimerA 寄存器

TimerA 有丰富的寄存器供用户使用。如下表所列：

表: TimerA 的寄存器

寄存器	缩写	读写类型
TimerA 控制寄存器	TACTL	读/写
TimerA 计数器	TAR	读/写
捕获/比较控制寄存器 0	CCTL0	读/写
捕获/比较寄存器 0	CCR0	读/写
捕获/比较控制寄存器 1	CCTL1	读/写
捕获/比较寄存器 1	CCR1	读/写
捕获/比较控制寄存器 2	CCTL2	读/写
捕获/比较寄存器 2	CCR2	读/写
中断向量寄存器	TAIV	读/写

TimerA 中断

TimerA 中断可以由计数器溢出引起，也可以来自捕获/比较寄存器。
TimerA 使用两个中断向量，一个单独分配个 CCR0，另一个作为共用中断向量分配给定时器和其他捕获/比较寄存器。
CCR0 中断向量具有最高的优先级，CCIFG0 在中断服务时能自动复位。中断向量的名称为 TIMER0_A0_VECTOR。
CCR1-CCRx 与定时器共用另一个中断向量 (TIMER0_A1_VECTOR)，属于多源中断。在多源中断中，TAIV 用于确定中断请求的中断源。

▸ TimerA 中断示例

TimerA 工作模式

TimerA 共有四种工作模式:

1. 停止模式 MC_0
2. 增计数模式 MC_1
3. 连续计数模式 MC_2
4. 增/减计数模式 MC_3

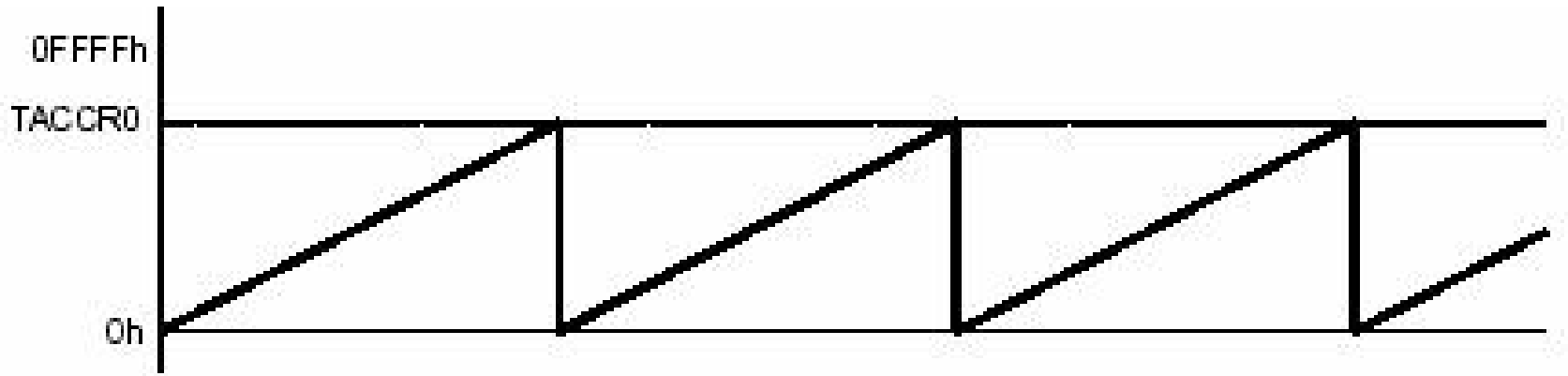
停止模式

停止模式用于定时器暂停，并不发生复位，所有寄存器现行的内容在停止模式结束后都可用。当定时器暂停后重新计数时，计数器将从暂停时的值开始以暂停前的计数方向计数。
例如，停止模式前，Timer_A 工作于增/减计数模式并且处于下降计数方向，停止模式后，Timer_A 仍然工作于增/减计数模式，从暂停前的状态开始继续沿着下降方向开始计数。如果不想这样，则可通过 TACTL 中的 CLR 控制位来清除定时器的方向记忆特性。

增计数模式

捕获/比较寄存器 CCR0 用作 Timer_A 增计数模式的周期寄存器，因为 CCR0 为 16 位寄存器，所以该模式适用于定时周期小于 65 536 的连续计数情况。计数器 TAR 可以增计数到 CCR0 的值，当计数值与 CCR0 的值相等 (或定时器值大于 CCR0 的值) 时，定时器复位并从 0 开始重新计数。

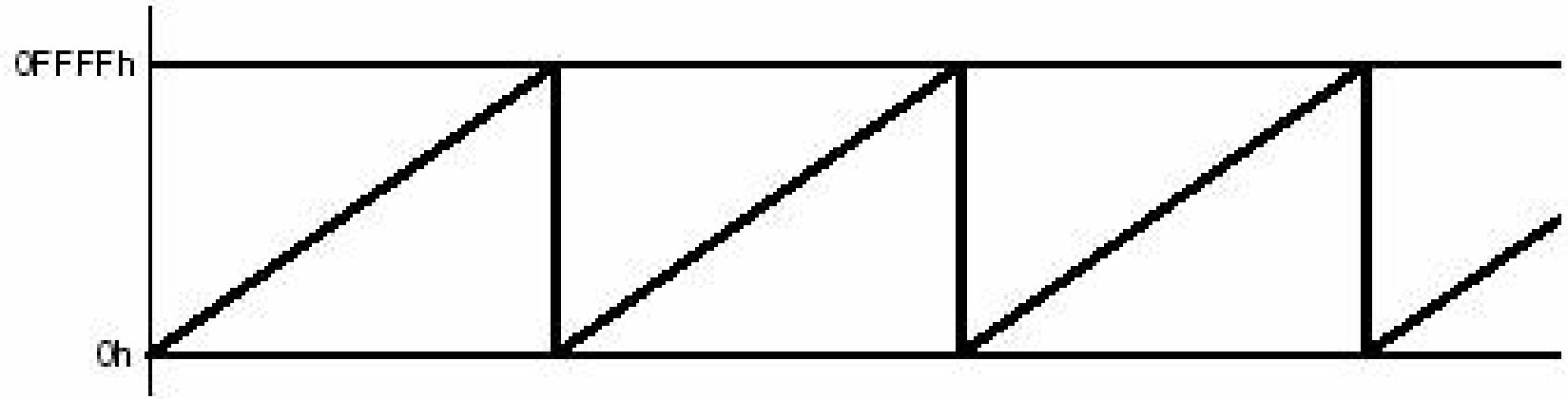
图: 增计数模式的计数过程



连续计数模式

需要 65 536 个时钟周期的定时应用场合常用连续计数模式。定时器从当前值计数到 0FFFFH 后，又从 0 开始重新计数。

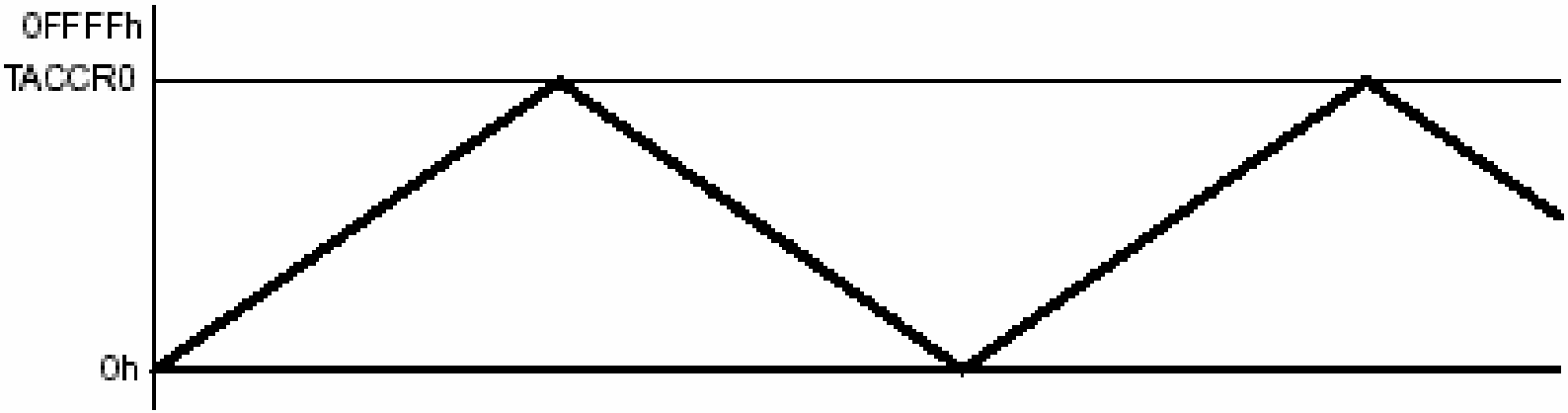
图：连续计数模式的计数过程



增/减计数模式

需要对称波形的情况经常可以使用增/减计数模式，该模式下，定时器先增计数到 CCR0 的值，然后反向减计数到 0。计数周期仍由 CCR0 定义，它是 CCR0 计数器数值的 2 倍。

图：增/减计数模式的计数过程



捕获模式

- CCTLx 中的 CAPx=1, 该模块工作在捕获模式。这时如果在选定的引脚上发生设定的脉冲触发沿 (上升沿、下降沿或任意跳变), 则 TAR 中的值将写入到 CCRx 中。
- 每个捕获/比较寄存器能被软件用于时间标记。可用于各种目的:

测量软件程序所用时间 测量硬件事件之间的时间 测量系统频率

- 当捕获完成后, 中断标志位 CCIFGx 被置位。

输出单元

每个捕获/比较模块包含一个输出单元，用于产生输出信号

Table 12. Timer0_A3 信号接线

输入引脚编号			器件输入信号	模块输入名称	模块区块	模块输出信号	输出引脚编号		
PW20，N20	PW28	RHB32					PW20，N20	PW28	RHB32
P1.0-2	P1.0-2	P1.0-31	TACLK	TACLK	定时器	不适用			
			ACLK	ACLK					
			SMCLK	SMCLK					
PinOsc	PinOsc	PinOsc	TACLK	INCLK					
P1.1-3	P1.1-3	P1.1-1	TA0.0	CCI0A	CCR0	TA0	P1.1-3	P1.1-3	P1.1-1
			ACLK	CCI0B			P1.5-7	P1.5-7	P1.5-5
			V_{SS}	GND				P3.4-15	P3.4-14
			V_{CC}	V_{CC}					
P1.2-4	P1.2-4	P1.2-2	TA0.1	CCI1A	CCR1	TA1	P1.2-4	P1.2-4	P1.2-2
			CAOUT	CCI1B			P1.6-14	P1.6-22	P1.6-21
			V_{SS}	GND			P2.6-19	P2.6-27	P2.6-26
			V_{CC}	V_{CC}				P3.5-19	P3.5-18
	P3.0-9	P3.0-7	TA0.2	CCI2A	CCR2	TA2		P3.0-9	P3.0-7
PinOsc	PinOsc	PinOsc	TA0.2	CCI2B				P3.6-20	P3.6-19
			V_{SS}	GND					
			V_{CC}	V_{CC}					

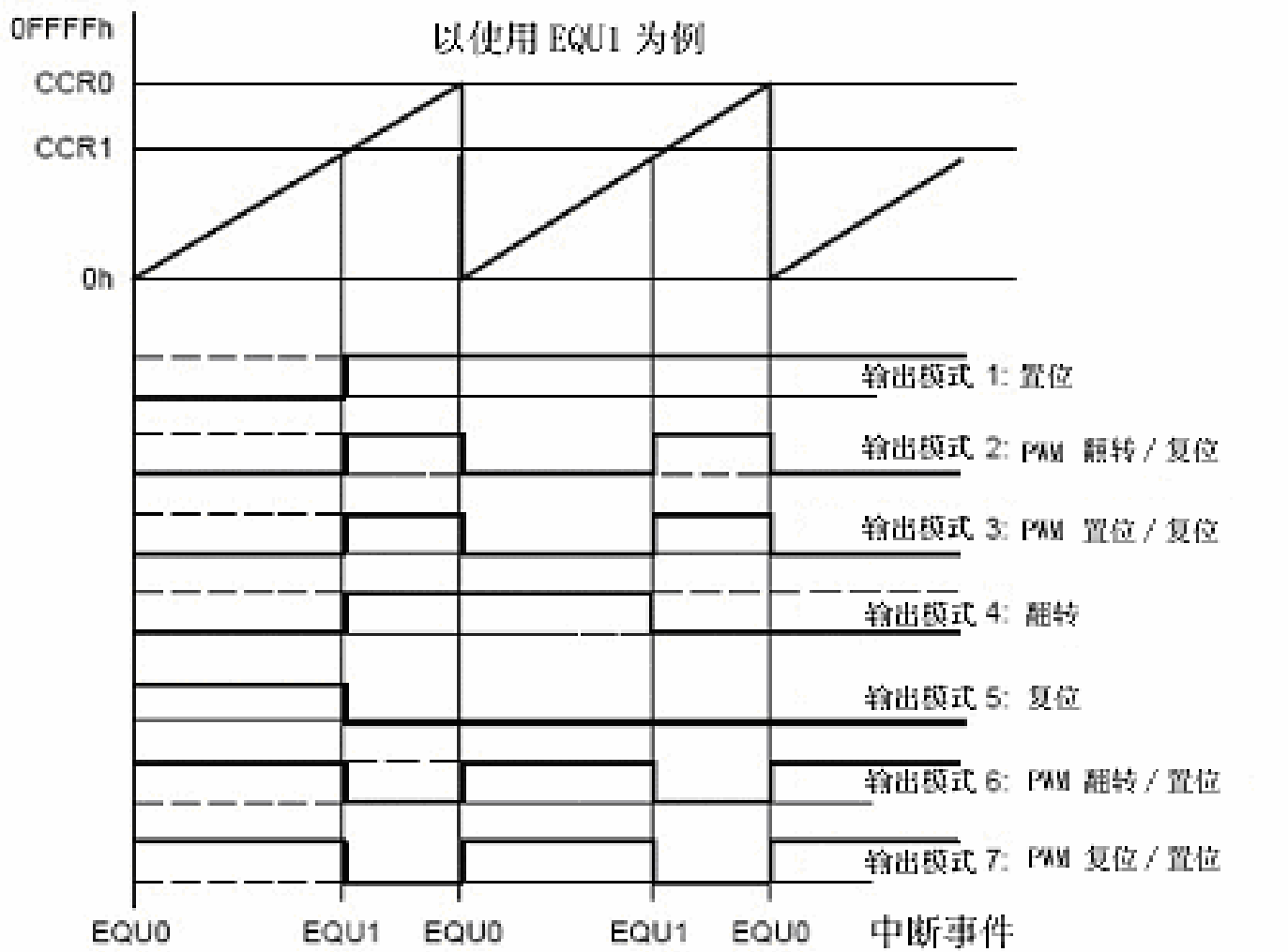
输出模式

- OUTMOD_0 输出模式: 输出信号 OUTx 由每个捕获/比较模块的控制寄存器 CCTLx 中的 OUTx 位定义, 并在写入该寄存器后立即更新。最终位 OUTx 直通。
- OUTMOD_1 置位模式: 输出信号在 TAR 等于 CCRx 时置位, 并保持置位到定时器复位或选择另一种输出模式为止。
- OUTMOD_2 PWM 翻转/复位模式: 输出在 TAR 的值等于 CCRx 时翻转, 当 TAR 的值等于 CCR0 时复位。
- OUTMOD_3 PWM 置位/复位模式: 输出在 TAR 的值等于 CCRx 时置位, 当 TAR 的值等于 CCR0 时复位

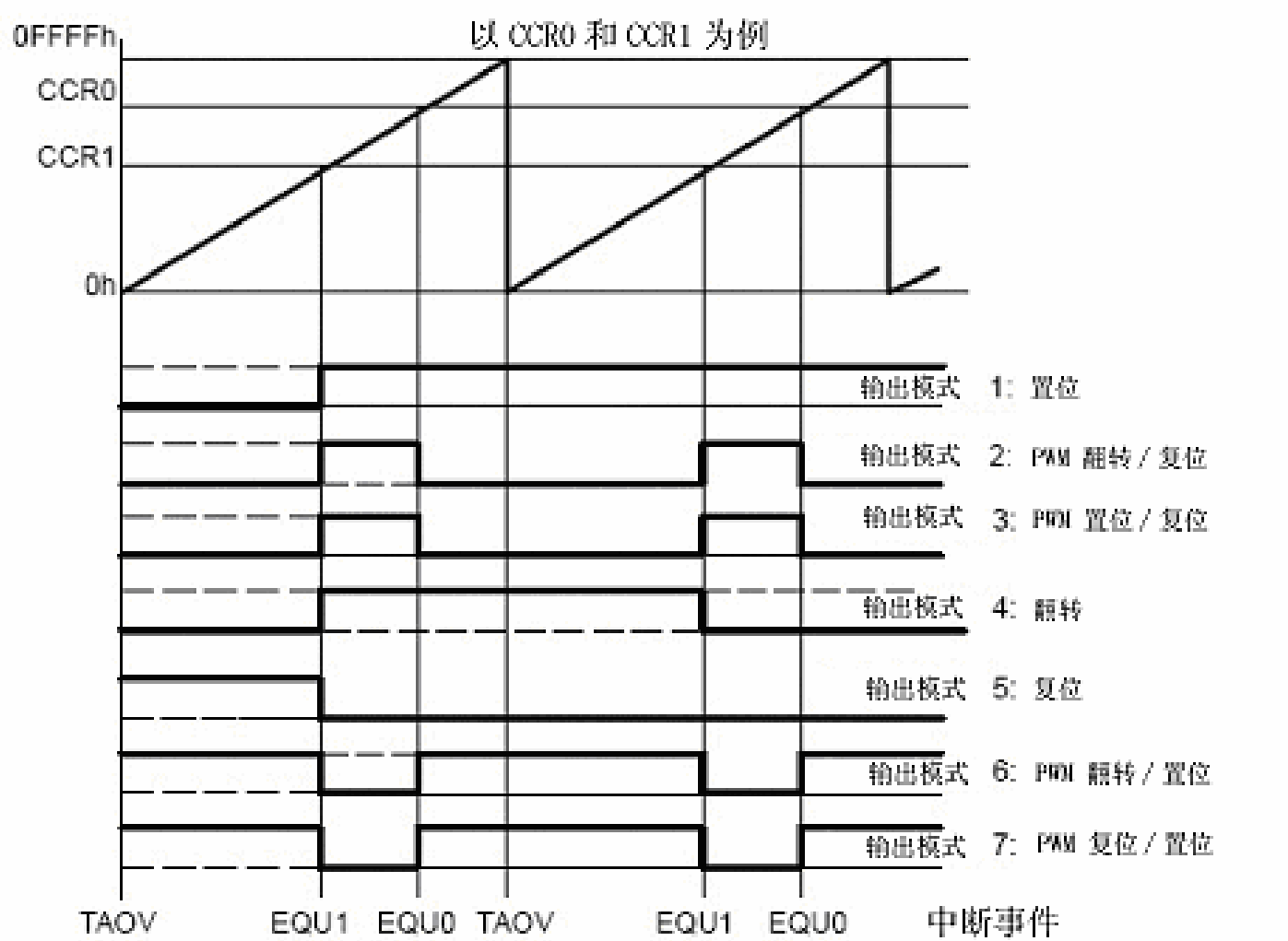
输出模式

- OUTMOD_4 翻转模式: 输出电平在 TAR 的值等于 CCRx 时翻转, 输出周期是定时器周期的 2 倍。
- OUTMOD_5 复位模式: 输出在 TAR 的值等于 CCRx 时复位, 并保持低电平直到选择另一种输出模式。
- OUTMOD_6 PWM 翻转/置位模式: 输出电平在 TAR 的值等于 CCRx 时翻转, 当 TAR 值等于 CCR0 时置位。
- OUTMOD_7 PWM 复位/置位模式: 输出电平在 TAR 的值等于 CCRx 时复位, 当 TAR 的值等于 CCR0 时置位。

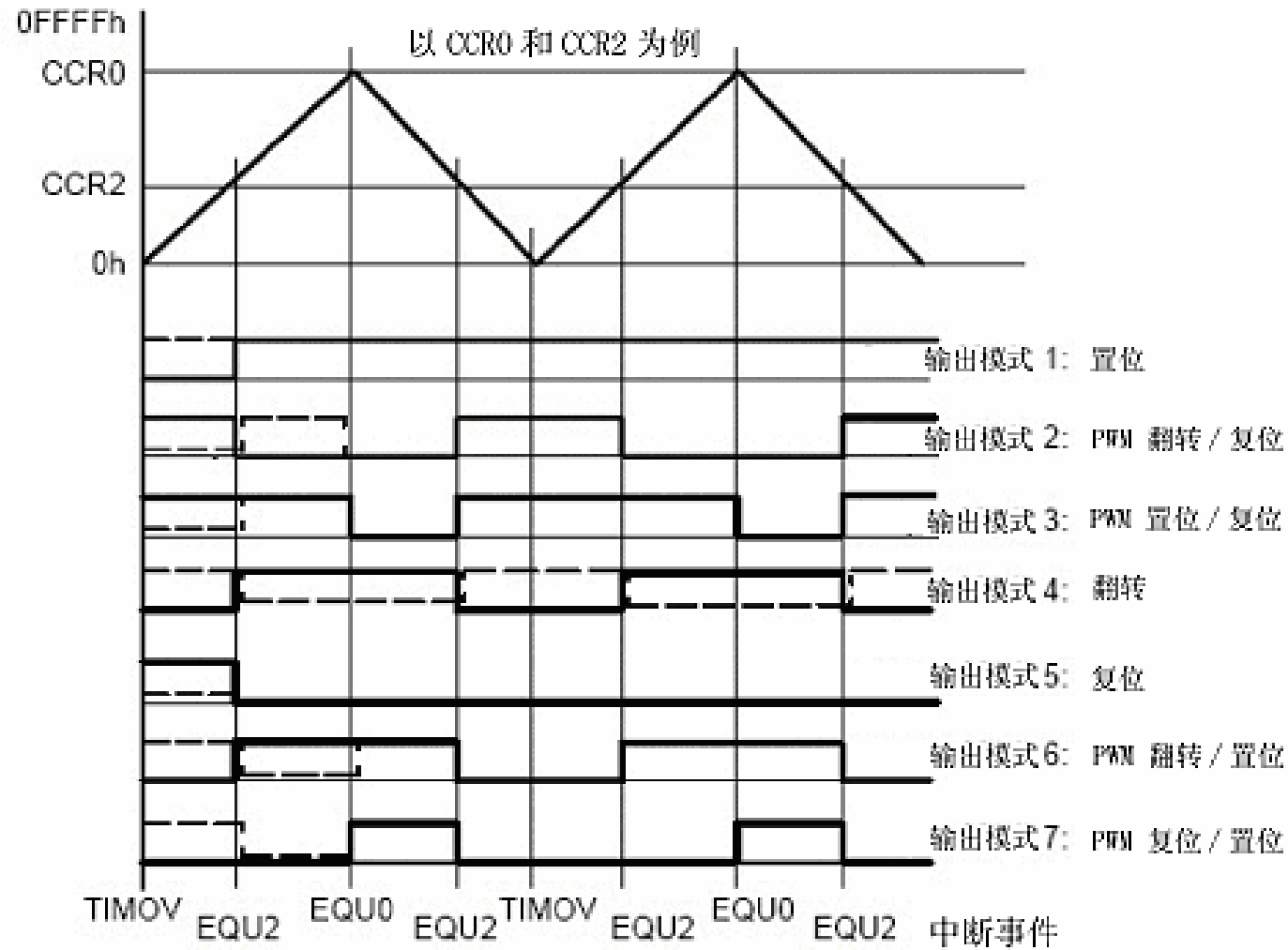
增计数模式输出波形


以使用 EQU1 为例
0FFFFh
CCR0
CCR1
0h
输出模式 1: 置位
输出模式 2: PWM 翻转 / 复位
输出模式 3: PWM 置位 / 复位
输出模式 4: 翻转
输出模式 5: 复位
输出模式 6: PWM 翻转 / 置位
输出模式 7: PWM 复位 / 置位
EQU0
EQU1
EQU0
EQU1
EQU0
中断事件

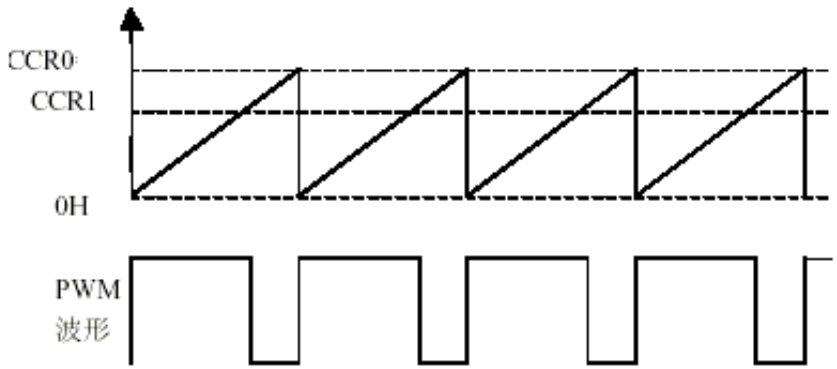
连续计数模式下输出波形



增/减计数模式下输出波形



使用 TimerA 实现 PWM



CCR0 决定了信号的周期，改变 CCR1 即可改变占空比。

▸ PWM 示例

Example

```
1  void ConfigClock(void)
2  {
3      int i;
4      DCOCTL = CALDCO_1MHZ;
5      BCSCTL1 = CALBC1_1MHZ;
6      BCSCTL3 |= LFXT1S_0;    //ACLK = LFXT1 = 32768Hz
7      do {
8          IFG1 &= ~OFIFG;          //清除振荡器失效标志
9          for (i = 0xf;i>0;i--);
10     } while ((IFG1 & OFIFG)!=0);       //如果振荡器失效标志存在
11     BCSCTL2 |= SELM_0 + DIVM_0 + DIVS_0;
12     //MCLK =DCO SMCLK = DCO
13 }
```

返回

当 P1.4 引脚出现下降沿信号，则触发中断，将系统从 LPM4 唤醒，P1.0 信号取反。

Example

```
1  #include <msp430g2553.h>
2  void main(void)
3  {
4      WDTCTL = WDTPW +WDTHOLD; //关闭看门狗
5      P1DIR |= BIT0;           //P1.0 输出
6      P1IE  |= BIT4;           //P1.4 中断允许
7      P1IES |= BIT4;           //P1.4 下降沿触发中断
8      P1IFG &= ~BIT4;          //清除 P1.4 中断标志
9      _BIC_SR(LPM4_bits+GIE); //进入 LPM4 并允许中断
10 }
11 #pragma vector = PORT1_VECTOR
12 __interrupt void Port1_ISR(void)
13 {
14     P1OUT ^= BIT0;           //P1.0 取反
15     P1IFG &= ~BIT4;          //P1.4 中断标志清除
16 }
```

返回

Example

```
1  #include <msp430g2553.h>
2  void main(void){
3      BCSCTL1 |= DIVA_1;       //ACLK/2
4      BCSCTL3 |= LFXT1S_2;     //ACLK = VLO
5      WDTCTL = WDT_ADLY_1000; //WDT 工作在定时器模式
6      IE1 |= WDTIE;            //看门狗中断允许
7      P1DIR = 0xFF;            //P1 输出
8      P1OUT = 0;               //P1 输出 0
9      while(1){
10         int i;
11         P1OUT |= BIT0;       //P1.0 置位
12         for(i= 10000; i>0 ; i--); //软件延时
13         P1OUT &= ~BIT0;          //P1.0 复位
14         _BIS_SR(LPM3_bits + GIE); //进入 LPM3，ACLK 是活动的
15     }
16 }
17 #pragma vector = WDT_VECTOR
18 __interrupt void watchdog_timer(void){
19     _BIC_SR_IRQ(LPM3_bits); //清除 LPM3 唤醒 CPU
20 }
```

返回

Example

```
1  #include <msp430g2553.h>
2  void main(void){
3      WDTCTL = WDTPW + WDTHOLD;
4      TACTL = TASSEL_2 + TACLR + TAIE; //SMCLK, 清除 TAR，允许定时器溢出中断
5      P1DIR |= BIT0;                   //P1.0 输出
6      TACTL |= MC_2;                   //连续计数模式
7      _EINT();                         //允许全局中断
8      while(1){
9          _BIS_SR(CPUOFF);
10     }
11 }
12 #pragma vector = TIMER0_A1_VECTOR
13 __interrupt void TimerA_ISR(void){
14     switch(TAIV){
15         case 2: break;       //CCR1 中断处理没有使用
16         case 4: break;       //CCR2 中断处理没有使用
17         case 10: P1OUT ^= BIT0; // P1.0 翻转;
18                  break
19     }
20 }
```

返回

Example

```
1  #include <msp430x44x.h>
2  void main(void)
3  {
4      WDTCTL = WDTPW +WDTHOLD;
5      TACTL = TASSEL0 + TACLR; //ACLK 清除 TAR
6      CCR0 = 512-1;     //PWM 周期
7      CCTL1 = OUTMOD_7;
8      CCR1 = 384;    //占空比 384/512=0.75
9      CCTL2 = OUTMOD_7;
10     CCR2 = 128;    //占空比 128/512=0.25
11     P1DIR |= BIT2; //P1.2 输出
12     P1SEL |= BIT2; //P1.2 TA1
13     P2DIR |= BIT0; //P2.0 输出
14     P2SEL |= BIT0; //P2.0 TA2
15     TACTL |= MC_0; //TimerA 增计数模式
16     _BIS_SR(LPM3_bits); //进入 LPM3
17 }
```

返回

谢谢大家！

Contact Me: andyhuzhill@gmail.com