PWM定时器和WatchDog定时器

- PWM定时器
- PWM定时器控制寄存器
- PWM定时器应用实例
- WatchDog定时器
- WatchDog定时器控制寄存器
- WatchDog定时器程序实例

BACK

9.1 PWM定时器

9.1.1 PWM定时器概述

在Exynos 4412处理器中，共有5个32位的具有脉冲宽度调制(Pulse Width Modulation，PWM)功能的定时器，这些定时器都可产生内部中断信号给ARM子系统。另外，定时器0、1、2、3具有脉冲宽度调制功能，并可驱动其对应的I/O口。其中，定时器0有可选的死区(dead-zone)产生功能，用以支持大电流设备；定时器4是内置的，没有外部引脚。

PWM定时器的特点：

- 5个32位定时器；
- 2个8位的预分频器对PLCK进行第一次分频，5个时钟分频器和多路复用器进行第二次分频；
- 可编程选择PWM通道；
- 4个独立的PWM通道，可编程进行占空比和极性控制；

PWM定时器的特点：

- 提供静态配置方式，在PWM没有启动时使用；
- 提供动态配置方式，在PWM运行期间使用；
- 支持自动重载模式和触发脉冲模式；
- 两个PWM输出具有死区发生器；

9.1.2 PWM定时器工作原理

5个定时器都采用APB-PCLK作为时钟源。通过预分频器进行第一级分频，其中定时器0和1共用一个可编程8位的预分频器0，定时器2、3、4共用另外一个预分频器1。之后，通过定时器各自的时钟分频器进行第二级分频，时钟分频器有5种分频输出(1/1，1/2，1/4，1/8，1/16)。然后，通过TCNTBn和TCMPBn进行计数和电平翻转。

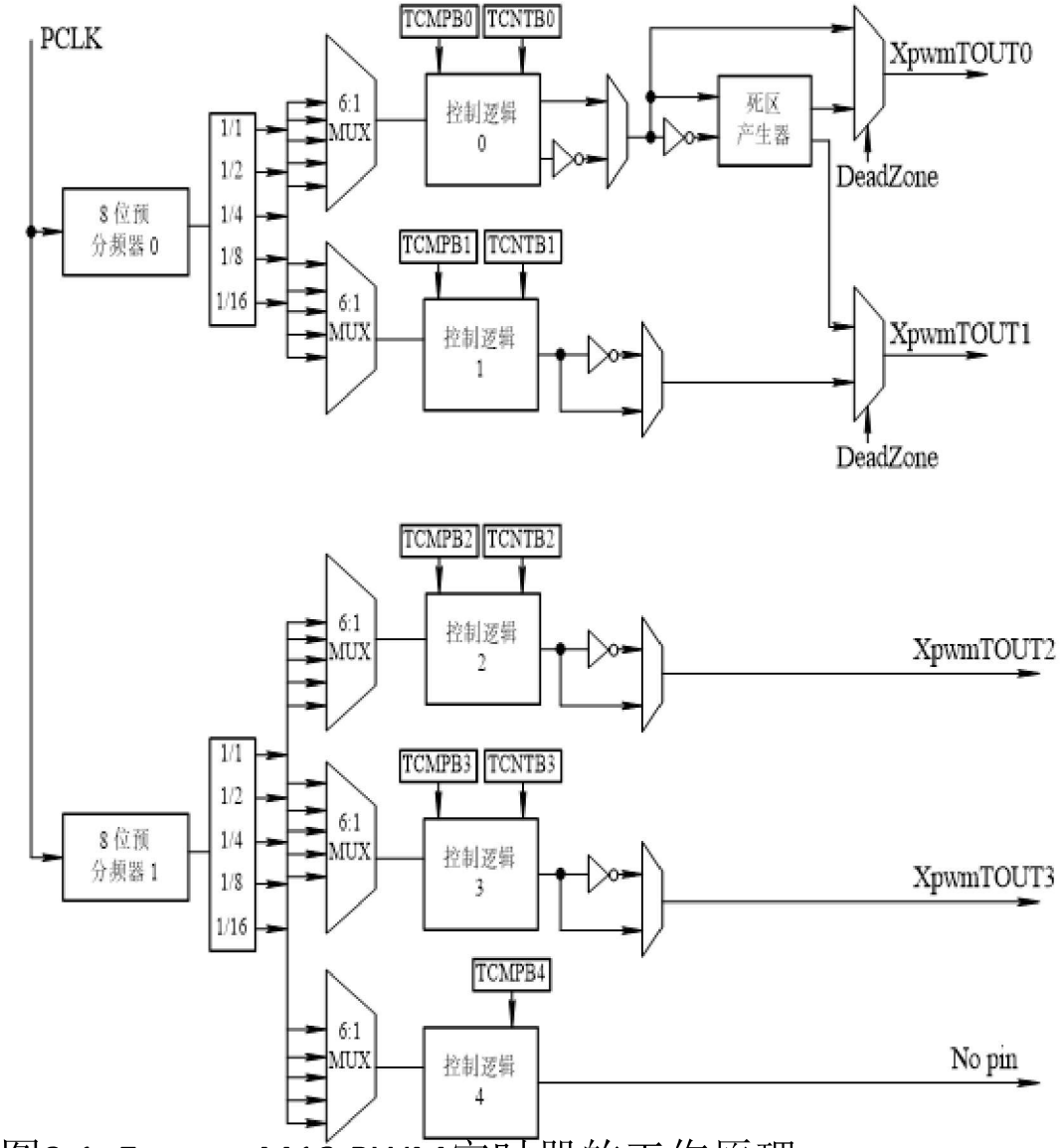

图9.1 Exynos 4412 PWM定时器的工作原理

PWM定时器工作的具体过程：

● 当时钟被使能后，定时器计数缓冲寄存器(TCNTBn)把计数初始值下载到递减计数器(TCNTn)中，定时器比较缓冲寄存器(TCMPBn)把其初始值下载到比较寄存器(TCMPn)中。

●递减计数器从TCNTBn得到初值以后，按其时钟频率进行递减计数。当其值达到0时，产生定时器中断请求并通知CPU该次计时完成。

● TCMPBn的值用于脉冲宽度调制。当定时器的递减计数器的值和比较寄存器的值相等时，PWM输出将改变输出电平的状态。

9.1.3 PWM定时器的死区功能

图9.1中，XpwmTOUT0和XpwmTOUT1可用于电源设备的PWM控制。Exynos 4412具有死区产生器，能够产生如图9.3所示的波形。这个功能允许在一个设备关闭和另外一个设备开启之间插入一个时间间隔。这个时间间隔能够有效防止两个设备的同时启/停。。

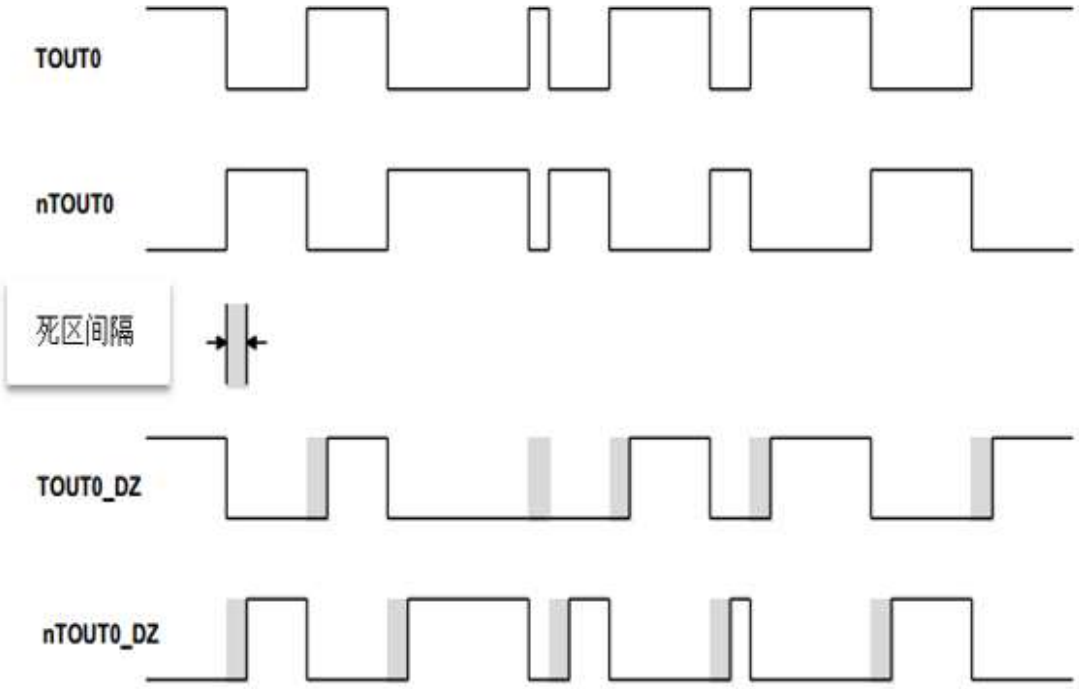

图9.3 死区使能时的波形

9.2 PWM定时器控制寄存器

1. 定时器配置寄存器0(TFCG0)

该寄存器用于配置两个8位预分配器的值和死区长度，

如表9.1所示

名 称	位域	类型	功 能 描 述	复位值
死区长度	[23:16]	RW	决定死区长度，时间单位与定时器0 设置相同，死区长度n = 0~254，计算时用 n+1	0x00
预分频器1	[15:8]	RW	定义定时器2、3、4 预分频值，预分频值=1~255	0x01
预分频器0	[7:0]	RW	定义定时器0、1 预分频值，预分频值=1~255	0x01

表9.1 定时器配置寄存器0(TFCG0)

2. 定时器配置寄存器**1(TFCG1)**

该寄存器用于配置PWM定时器多路开关MUX的输入，

即选择分频器的值，如表9.2所示。

名　称	位域	类型	功 能 描 述	复位值
保留	[31:20]	—	保留	0x000
定时器 4 分频值	[19:16]	RW	0x0 = 1/1，0x1 = 1/2，0x2 = 1/4，0x3 = 1/8，0x4 = 1/16	0x0
定时器 3 分频值	[15:12]	RW	0x0 = 1/1，0x1 = 1/2，0x2 = 1/4，0x3 = 1/8，0x4 = 1/16	0x0
定时器 2 分频值	[11:8]	RW	0x0 = 1/1，0x1 = 1/2，0x2 = 1/4，0x3 = 1/8，0x4 = 1/16	0x0
定时器 1 分频值	[7:4]	RW	0x0 = 1/1，0x1 = 1/2，0x2 = 1/4，0x3 = 1/8，0x4 = 1/16	0x0
定时器 0 分频值	[3:0]	RW	0x0 = 1/1，0x1 = 1/2，0x2 = 1/4，0x3 = 1/8，0x4 = 1/16	0x0

表9.2 定时器配置寄存器1(TFCG1)

定时器的输入频率的计算公式为：

定时器的输入时钟频率 = PCLK/({预分频值 + 1})/{分频值}

其中：

预分频值的取值范围为1～255，按照表9.1；

各个定时器的分频值则按表9.2列出的寄存器TCFG1中各对应位域进行设置；

3. 定时器控制寄存器**(TCON)**

该寄存器用于自动重载、定时器自动更新、定时器启/停、输出翻转、死区启/停等功能的控制，如表9.3所示。

名 称	位域	类型	功 能 描 述	复位值
定时器 4 自动重载开/关	[22]	RW	0＝自动重载关；1＝自动重载开	0x0
定时器 4 手动更新开/关	[21]	RW	0＝无操作；1＝手动更新 TCNTB4	0x0
定时器 4 启动开/关	[20]	RW	0＝停止；1＝启动	0x0
定时器 3 自动重载开/关	[19]	RW	0＝自动重载关；1＝自动重载开	0x0
定时器 3 输出翻转器开/关	[18]	RW	0＝翻转器关；1＝TOUT_3 翻转器开	0x0
定时器 3 手动更新开/关	[17]	RW	0＝无操作；1＝手动更新 TCNTB3	0x0
定时器 3 启动开/关	[16]	RW	0＝停止；1＝启动	0x0
定时器 2 自动重载开/关	[15]	RW	0＝自动重载关；1＝自动重载开	0x0
定时器 2 输出翻转器开/关	[14]	RW	0＝翻转器关；1＝TOUT_2 翻转器开	0x0
定时器 2 手动更新开/关	[13]	RW	0＝无操作；1＝手动更新 TCNTB2	0x0
定时器 2 启动开/关	[12]	RW	0＝停止；1＝启动	0x0
定时器 1 自动重载开/关	[11]	RW	0＝自动重载关；1＝自动重载开	0x0
定时器 1 输出翻转器开/关	[10]	RW	0＝翻转器关；1＝TOUT_1 翻转器开	0x0
定时器 1 手动更新开/关	[9]	RW	0＝无操作；1＝手动更新 TCNTB1	0x0
定时器 1 启动开/关	[8]	RW	0＝停止；1＝启动	0x0
死区使能	[4]		0＝不使能；1＝使能	0x0
定时器 0 自动重载开/关	[3]	RW	0＝自动重载关；1＝自动重载开	0x0
定时器 0 输出翻转器开/关	[2]	RW	0＝翻转器关；1＝TOUT_0 翻转器开	0x0
定时器 0 手动更新开/关	[1]	RW	0＝无操作；1＝手动更新 TCNTB0	0x0

表9.3 定时器控制寄存器(TCON)

4. 定时器计数缓冲寄存器**(TCNTBn，n = 0～4)**

该类寄存器用来预装PWM定时器的计数初值，如表9.4所示。

名 称	位域	类型	功 能 描 述	复位值
定时器 n 计数缓冲寄存器	[31:0]	RW	预装定时器 n(n = 0~4)计数初值	0x00000000

表9.4 定时器比较缓冲寄存器

5. 定时器比较缓冲寄存器**(TCMPBn，n = 0～3)**

该类寄存器用来更改PWM波形的电平状态，从而更改PWM的占空比，如表9.5所示。

名　称	位域	类型	功 能 描 述	复位值
定时器比较缓冲寄存器	[31:0]	RW	预装定时器 n(n=0~3)比较器的初值	0x00000000

表9.5 定时器比较缓冲寄存器

9.3 PWM定时器应用实例

以产生定时器0中断为例，介绍定时器0的配置过程。

/* 1. 相关寄存器的定义，关联各自的物理地址 */

```
#include "int.h"
#include "stdio.h"
#define          PWMTIMER_BASE      (0x 139D0000)
#define    TCFG0     ( *((volatile unsigned long *)(PWMTIMER_BASE+0x00)) )
#define    TCFG1     ( *((volatile unsigned long *)(PWMTIMER_BASE+0x04)) )
#define    TCON      ( *((volatile unsigned long *)(PWMTIMER_BASE+0x08)) )
#define    TCNTB0     ( *((volatile unsigned long *)(PWMTIMER_BASE+0x0C)) )
#define    TCMPB0    ( *((volatile unsigned long *)(PWMTIMER_BASE+0x10)) )
#define    TCNTO0     ( *((volatile unsigned long *)(PWMTIMER_BASE+0x14)) )
#define    TINT_CSTA  ( *((volatile unsigned long *)(PWMTIMER_BASE+0x44)) )
#define ulong unsigned long
void pwm_stop(void);
void timer_request(void);
void irq_handler(void);
void timer_init(ulong utimer,ulong uprescaler,ulong udivider,ulong utcntb,ulong utcmpb);
void irs_timer( );
int counter = 0;                          //用于记录中断发生的次数
```

/* 2. 定义各个功能函数 */

```
void pwm_stop(void)                    //停止timer0
{
        TCON &= ~0x1;
}
void irs_timer( )                      // timer0中断的中断处理
函数
{
TINT_CSTAT |= (0x1<<5);                //清timer0的中断状态寄存器
    printf("Timer0IntCounter = %d \r\n",counter++);     //打印中断发生次数
    intc_clearvectaddr( ); //清除vic相关的中断
}
```

```
void timer_request(void)
{
    printf("\r\n##############Timer test#############\r\n");
    pwm_stop( );
        //禁止timer0
    counter = 0;
    intc_setvectaddr(NUM_TIMER0,irs_timer); //设置timer0的中断
处理函数，未列出函数原型
    intc_enable(NUM_TIMER0);                    //使能
timer0中断，未列出函数原型
    //设置timer0的参数: 定时器0, 预分配值(65+1), 分频值
4(0b0100),     tcntb=62500, tcmpb=0
    timer_init(0,66,4,62500,0);

}
```

9.4 WatchDog定时器

9.4.1 看门狗原理

看门狗(WatchDog)原理上就是一个定时器。定时器对时钟脉冲进行计数，当定时器溢出时，产生复位信号，使得整个系统复位。在正常的程序或嵌入式系统中，需要定期对看门狗进行复位，使其重新计数，这样定时器不会溢出而导致复位系统，从而保证系统的正常运行。

看门狗的作用就是防止系统因意外而“跑飞”；而在可能导致整个系统瘫痪时，保证系统能够在无人监守的情况下仍然能够复位，正常运行。

9.4.2 Exynos 4412看门狗控制

Exynos 4412处理器的看门狗模块如图9.4所示，包括一个预分频因子、一个四分频的分频器和一个16位的计数器。输入时钟为PCLK，它经过两级分频(预分频和分频)，将分频后的时钟作为该定时器的输入时钟。当计数器计满后可以产生中断或者复位信号。

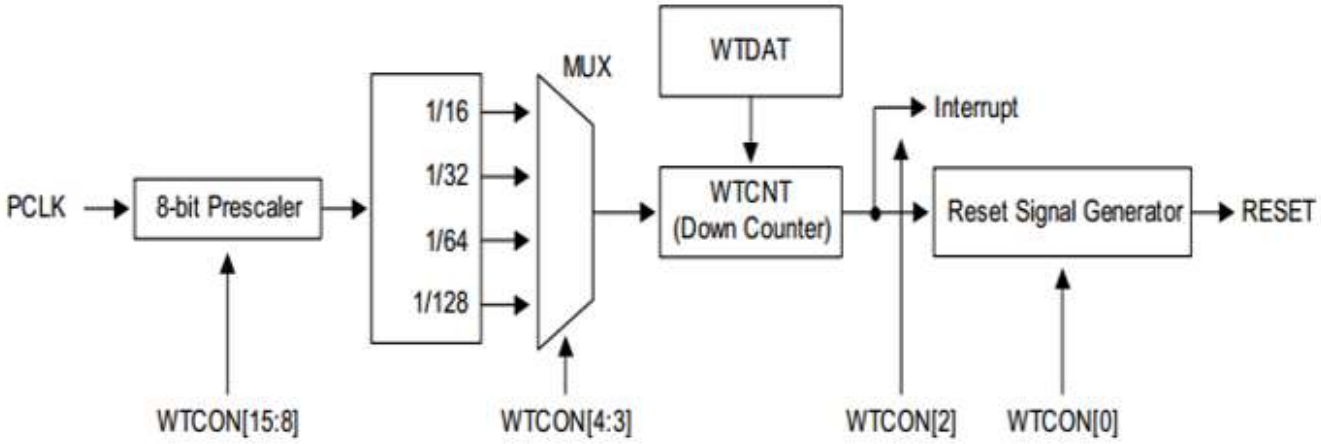

图9.4 Exynos 4412处理器的看门狗模块

看门狗定时器计数值的计算公式如下：

1. 输入到计数器的时钟周期

t_WatchDog = 1/(PCLK / (预分频值 + 1) / 分频值)

其中：预分频器(Prescaler，取值范围为0～254)及分频因子(Division factor，取值为16、32、64、128)的值由用户在WTCON(看门狗时钟控制寄存器)中设置。

(2) 看门狗的定时周期为

T = 计数值(WTCNT初值 - WTCNT当前值) * t_WatchDog

其中，WTCNT为看门狗计数寄存器，用来设置计数多少个时钟周期，乘以时钟周期则是定时的总时间。

9.5 WatchDog定时器控制寄存器

1. 看门狗时钟控制寄存器**(WTCON)**

该寄存器用于配置用户是否启用看门狗定时器、预分频值、4个分频比的选择、是否允许中断产生、是否允许复位操作等。

名称	位域	类型	功 能 描 述	复位值
预分频	[15:8]	RW	设置预分频值(0~2^8-1)	0x0
WDT timer	[5]	RW	0=禁止；1=使能	0x0
分频器	[4:3]	RW	00=16；01=32；10=64；11=128	0x0
中断使能	[2]	RW	0=禁止；1=使能	0x0
复位使能	[0]	RW	0=禁止；1=使能	0x0

表9.6 看门狗时钟控制寄存器(WTCON)

2. 看门狗数据寄存器**(WTDAT)**

WTDAT用于指定定时时间。在初始化看门狗操作后，看门狗寄存器的值不能被自动装载到看门狗计数器(WTCNT)中。

名　称	位域	类型	功能描述	复位值
计数重载值	[15:0]	RW	看门狗重载数值	0x8000

表9.7 看门狗数据寄存器(WTDAT)

3. 看门狗计数寄存器**(WTCNT)**

WTCNT用于设定看门狗定时器工作时计数器的当前计数值。初始化看门狗操作后，看门狗数据寄存器的值不能被自动装载到看门狗计数寄存器中。

名 称	位域	类型	功 能 描 述	复位值
计数值	[15:0]	RW	看门狗当前计数寄存器	0x8000

表9.8 看门狗计数寄存器(WTCNT)

9.6 WatchDog定时器程序实例

1. 看门狗程序设计思路

一般流程如下：

(1) 设置看门狗中断操作，包括全局中断和看门狗中断的使能以及看门狗中断向量的定义。如果只是进行复位操作，这一步可以不用设置。

(2) 对看门狗控制寄存器进行设置，包括设置预分频因子、分频器的分频值、中断使能和复位使能等。

(3) 对看门狗数据寄存器(WTDAT)和看门狗计数寄存器(WTCNT)进行设置。

(4) 启动看门狗定时器。

2. 看门狗程序设计

```
#include "int.h"
#include "stdio.h"
#define WDT_BASE        (0x10060000)
#define WTCON           ( *((volatile unsigned long
*)(WDT_BASE+0x00)) )
#define WTDAT           ( *((volatile unsigned long
*)(WDT_BASE+0x04)) )
#define WTCNT           ( *((volatile unsigned long
*)(WDT_BASE+0x08)) )
#define WTCLRINT        ( *((volatile unsigned long *)(WDT_BASE+0x0C)) )
#define ulong unsigned long
void isr_wtd(void);
void wtd_operate(ulong uenreset, ulong uenint, ulong uselectclk, ulong
uenwtd, ulong                    uprescaler, ulong uwtdat, ulong
uwtcnt);
```

```
void wtd_test(void)
{
    printf("\r\n\r\n##############WatchDog test##############\r\n");
    //设置看门狗中断的中断处理函数
    intc_setvectaddr(NUM_WDT,isr_wtd);
    intc_enable(NUM_WDT);                    //
使能看门狗中断
    wtd_operate(0,1,0,1,100,100000000,100000000);    //测试看门
狗的定时功能
}
```

```
//看门狗中断处理函数
void isr_wtd( )
{
    static int wtdcounter=0;          //记录中断发生次数
    printf("%d\r\n",++wtdcounter);
    WTCLRINT = 1;                     //看门狗相关中断清除
    intc_clearvectaddr();
    //VIC相关中断清除
    if(wtdcounter==5)
    {
        //看门狗reset
        printf("waiting system reset\r\n");
        wtd_operate(1,1,0,1,100,100000000,100000000);
    }
}
```

```
void wtd_operate(ulong uenreset, ulong uenint, ulong uselectclk, ulong
uenwtd, ulong uprescaler, ulong uwtdat, ulong uwtcnt)
{
    WTDAT = uwtdat;
    WTCNT = uwtcnt;
    /*uenreset: 是否使能reset
     *uenint: 是否使能中断
     *uselectclk: 分频系数
     *uenwtd:是否启动定时器
     *uprescaler:预分频系数
     */
    WTCON =
(uenreset<<0)|(uenint<<2)|(uselectclk<<3)|(uenwtd<<5)|((uprescaler)<<8);
}
```

问题与思考：

1．PWM波形的特点是什么？

2．在控制系统中，看门狗的作用是什么？

3．编程实现占空比为2：1、波形周期为9 ms的PWM波形。

4．编程实现：1 s内不对看门狗实现喂狗操作，看门狗就会自动复位。