第6章 中断与定时技术

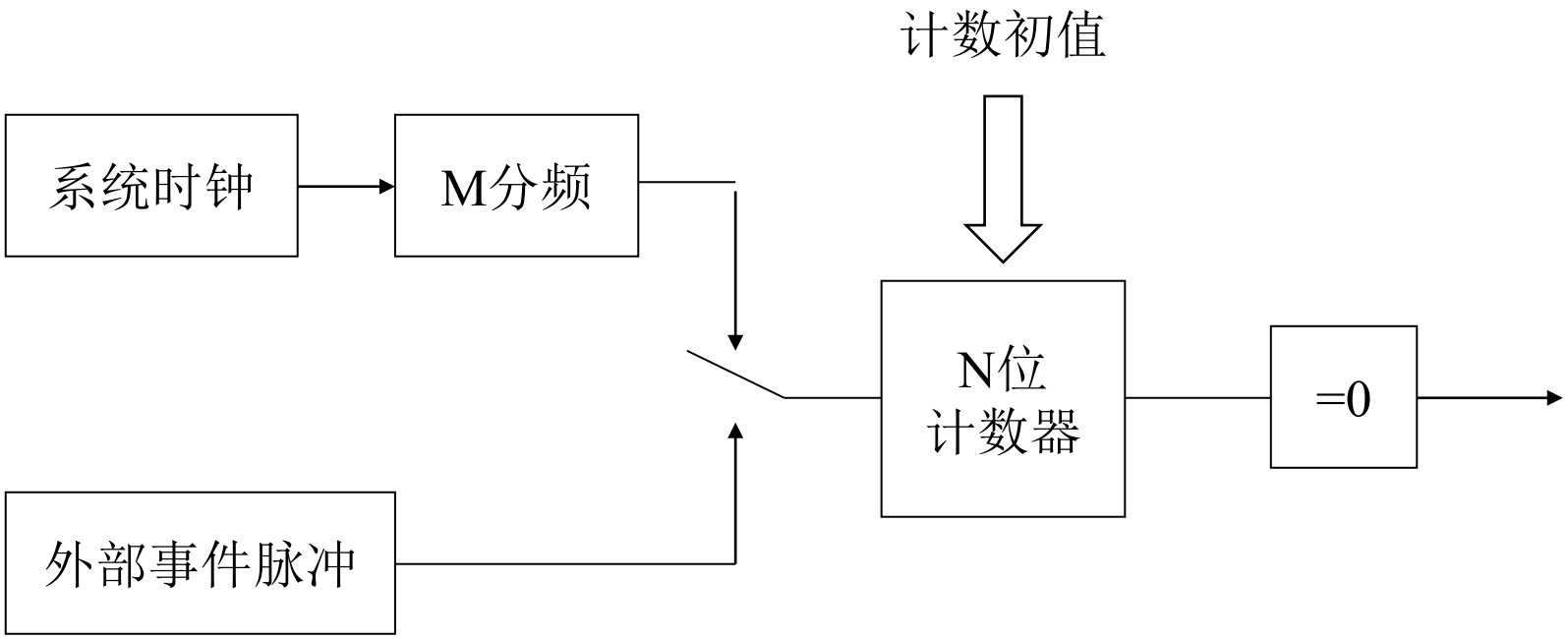
6.3 定时器工作原理

- 定时器是嵌入式系统的常用部件，其主要用作定时功能或计数功能。不同的定时部件在使用上有所差异，但它们的逻辑原理是相同的。
- S3C2410芯片中的定时部件有多个，不同的定时部件有不同的应用。本章主要介绍其中的看门狗定时器、RTC部件和Timer部件的控制原理及应用。

定时部件的一般性原理

- 定时器或计数器的逻辑电路本质上是相同的，它们之间的区别主要在用途上。它们都是主要由带有保存当前值的寄存器和当前寄存器值加1或减1逻辑组成。在应用时，定时器的计数信号是由内部的、周期性的时钟信号承担，以便产生具有固定时间间隔的脉冲信号，实现定时的功能。而计数器的计数信号是由非周期性的信号承担，通常是外部事件产生的脉冲信号，以便对外部事件发生的次数进行计数。因为同样的逻辑电路可用于这两个目的，所以该功能部件通常被称为“**定时/计数器**”。

- 定时/计数器内部工作原理是以一个N位的加1或减1计数器为核心，计数器的初始值由初始化编程设置，计数脉冲的来源有两类：系统时钟和外部事件脉冲。



三要素：计数源、计数方式、结果输出

- 定时工作方式：N位计数器的计数脉冲来源于内部系统时钟，并经过M分频。每个计数脉冲使计数器加1或减1，当N位计数器里的数加到0或减到0时，则会产生一个“**回0信号**”，该信号有效时表示N位计数器中的当前值是0。因为系统时钟的频率是固定的，其M分频后所得到的计数脉冲频率也就是固定的，因此通过对该频率脉冲的计数就转换为定时，实现了定时功能。
- 计数方式：N位计数器的计数脉冲来源于外部事件产生的脉冲信号。有一个外部事件脉冲，计数器加1或减1，直到**N位计数器中的值为0，产生“回0信号”**。
- N位计数器里初始值的计算，在不同的定时部件中其具体的计算公式是不同的。

6.4 S3C2410定时器

S3C2410芯片中的定时部件有多个，主要分为定时器及PWM、看门狗定时器和实时日历RTC。

PWM（Pulse Width Modulation 脉宽调制）：脉冲序列信号的占空比受某参数控制（调制），幅度和周期不变。

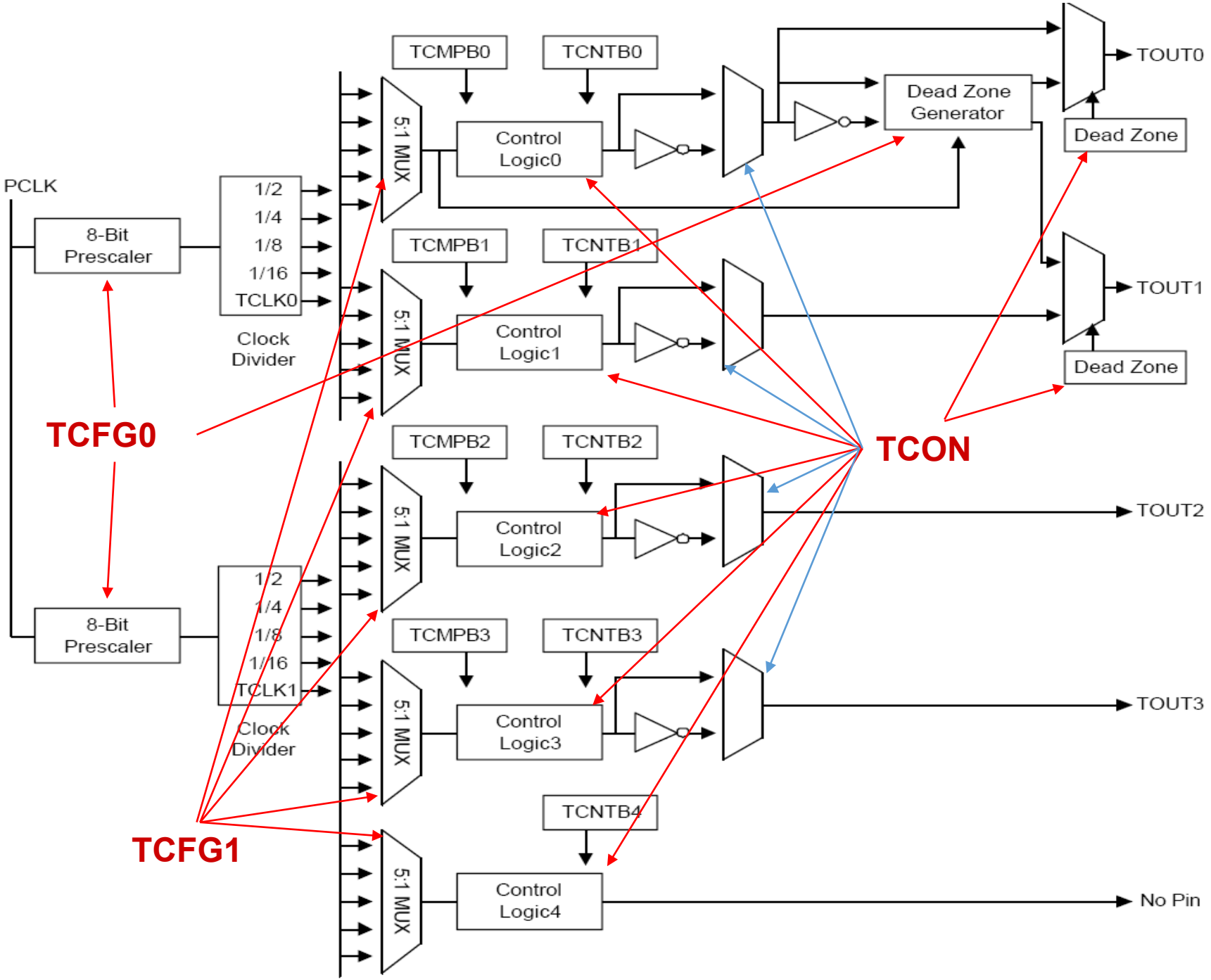
PWM应用：使用脉宽调制方法实现数字信号到模拟信号的变换。广泛应用在从测量、通信到功率控制与变换的许多领域中。

6.4.1 定时器及PWM

Timer部件主要是用于提供定时功能、脉宽调制（PWM: Pulse Width Modulation）功能，对于需要一定频率的脉冲信号、一定时间间隔的定时信号的应用场合，它都能提供应用支持。

S3C2410有5个16位的Timer部件：Timer0~Timer4。

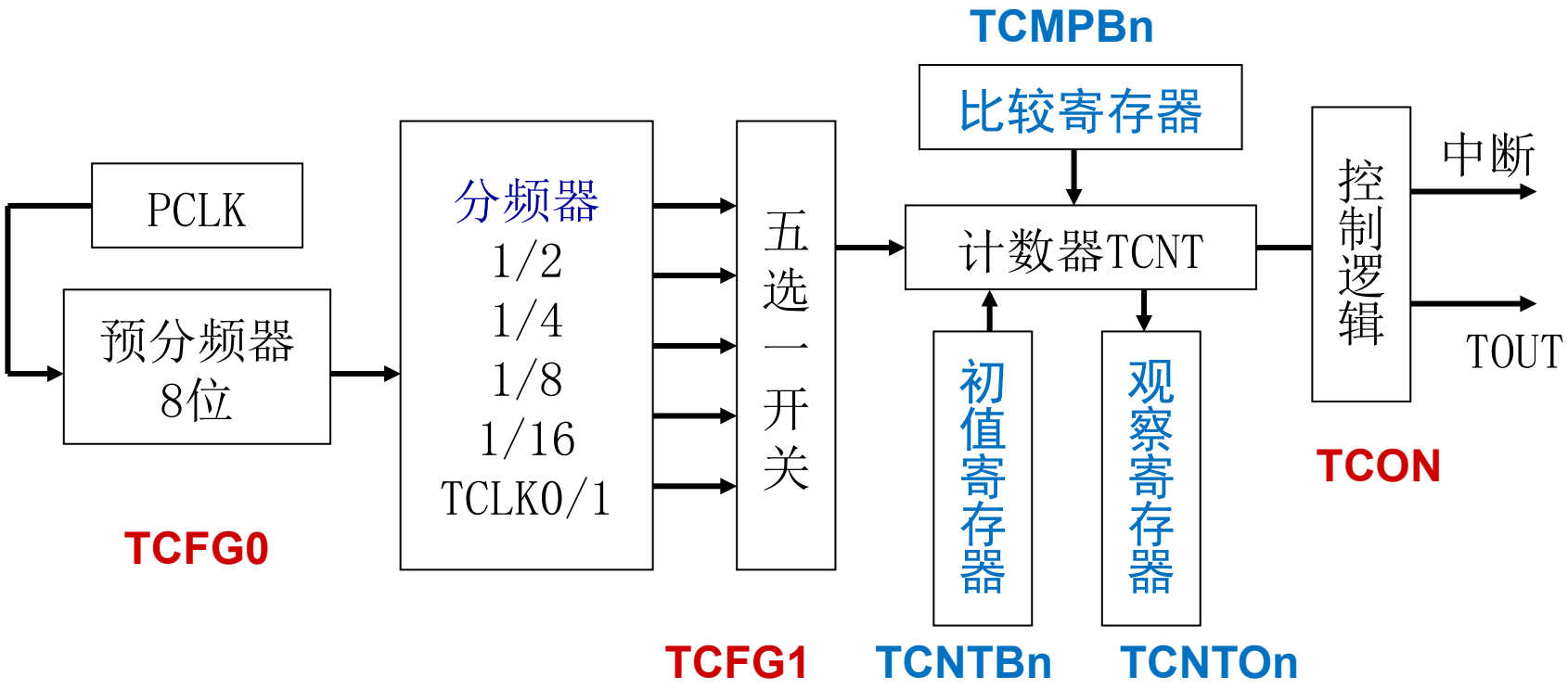
- Timer0、Timer1、Timer2、Timer3具有PWM功能，其中Timer0有死区控制（dead-zone),可用于驱动大功率设备
- Timer4仅作定时器使用（没有输出管脚）
- Timer0和Timer1共用一个8位的预分频器，Timer2、Timer3、Timer4共用另一个8位预分频器


PCLK
8-Bit Prescaler
1/2
1/4
1/8
1/16
TCLK0
Clock Divider
8-Bit Prescaler
1/2
1/4
1/8
1/16
TCLK1
Clock Divider
TCFG0
TCFG1
TCON
TCMPB0
TCNTB0
TCMPB1
TCNTB1
TCMPB2
TCNTB2
TCMPB3
TCNTB3
TCNTB4
5:1 MUX
5:1 MUX
5:1 MUX
5:1 MUX
5:1 MUX
Control Logic0
Control Logic1
Control Logic2
Control Logic3
Control Logic4
Dead Zone Generator
Dead Zone
Dead Zone
TOUT0
TOUT1
TOUT2
TOUT3
No Pin

S3C2410定时器的主要特性

- 5个16位定时器；
- 2个8位预分频器和2个4位分频器；
- 可编程PWM输出占空比输出；
- 具有初值自动重装连续输出模式和单脉冲输出模式；
- 具有死区(dead zone)发生器。

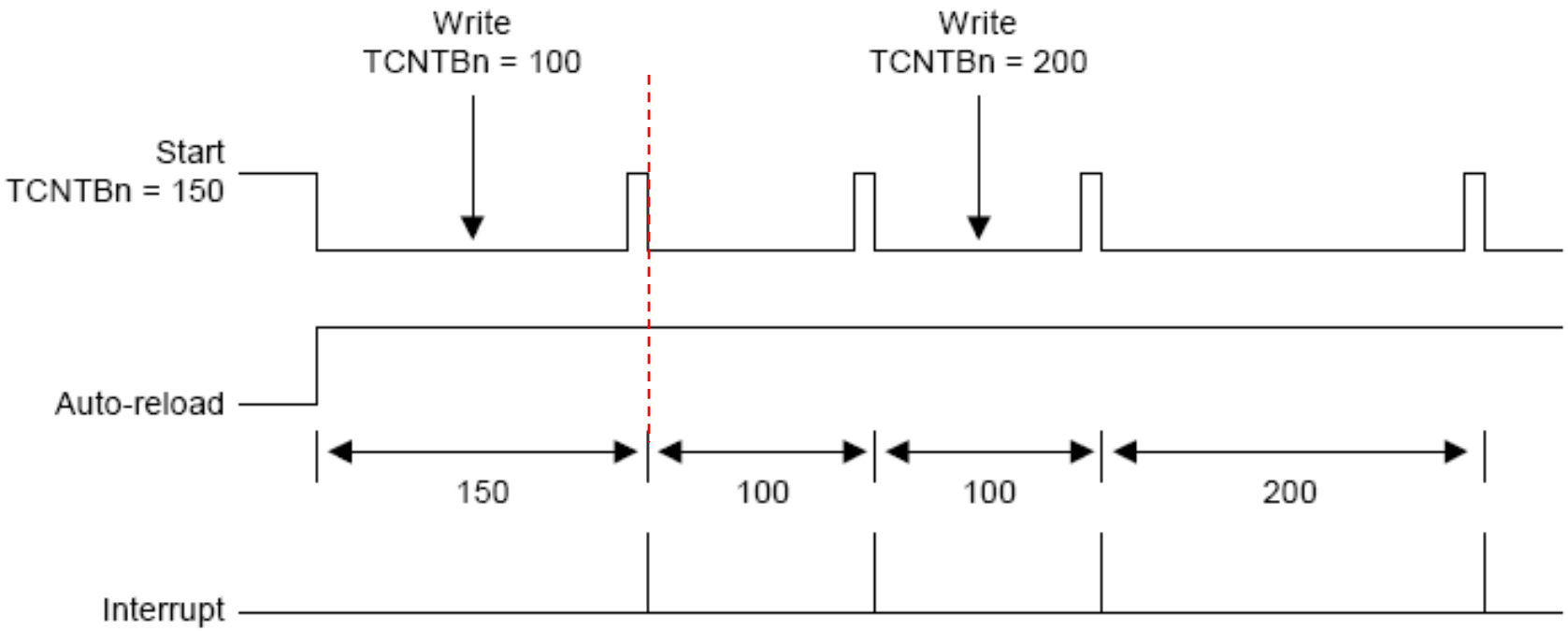
1. 定时器工作原理


TCMPBn
比较寄存器
PCLK
预分频器
8位
分频器
1/2
1/4
1/8
1/16
TCLK0/1
五选一开关
计数器TCNT
控制逻辑
中断
TOUT
TCON
TCFG0
TCFG1
初值寄存器
TCNTBn
观察寄存器
TCNTOn

（1）定时器工作过程

定时器工作过程：装入初值、启动计数（减法），计数结束(=0)产生中断请求，并且可以重装初值连续计数。

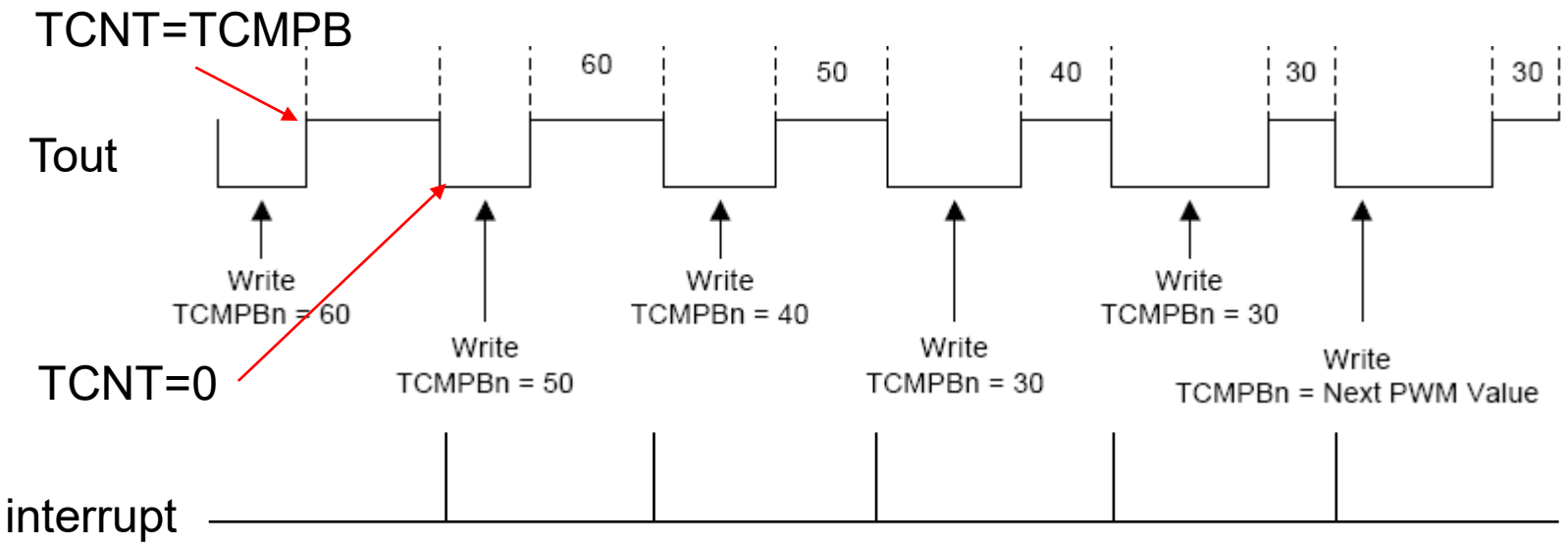
定时器双缓冲功能：不影响当前定时器操作而加载下一轮定时器初值。



（2）PWM输出

当计数器TCNT中的值减到与TCMPB的值相等时，TOUT的输出值取反。TOUT的输出可以设置为反相输出。改变TCMPB的值，便改变了输出方波的占空比。因此：

- PWM脉冲周期由TCNTBn确定
- PWM脉冲宽度由TCMPBn确定



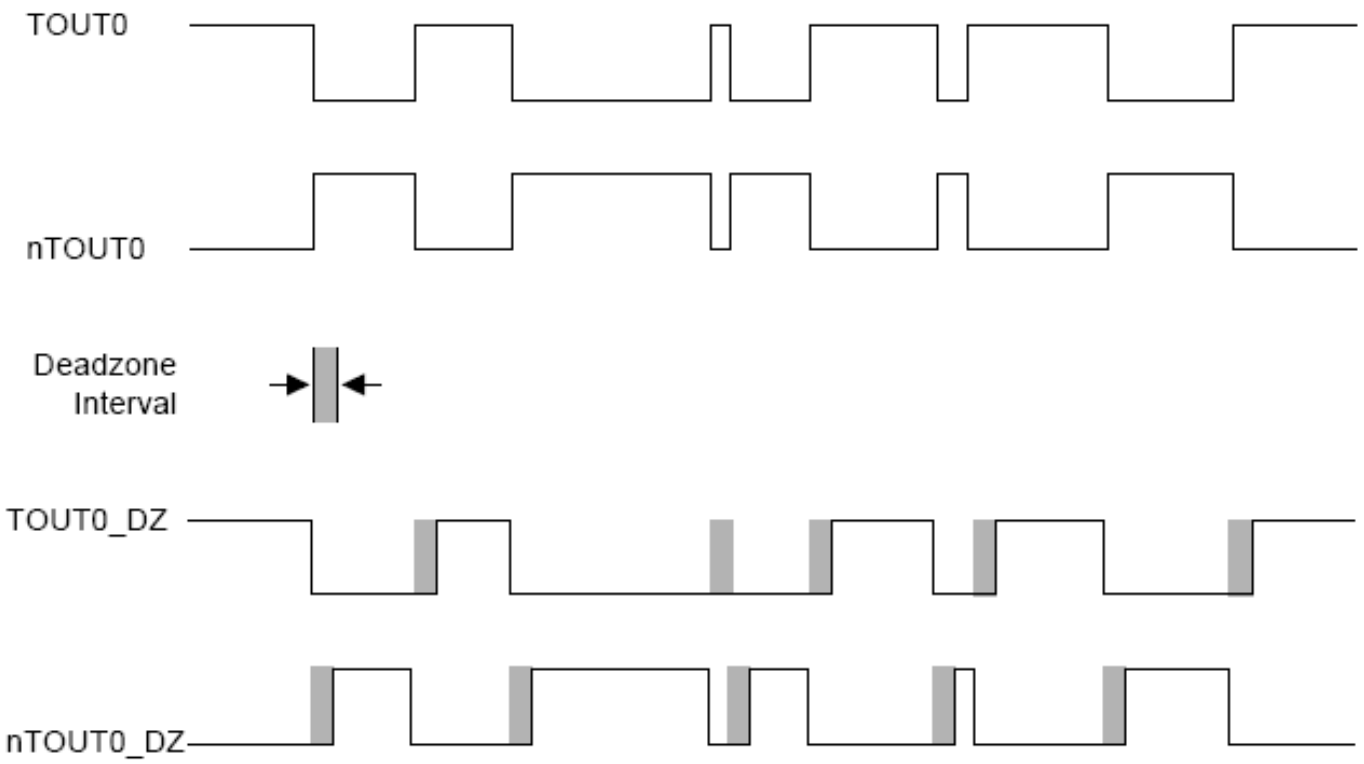
（3）死区发生器

死区：是一小段时间间隔，在这个时间间隔内，禁止两个开关同时处于开启状态，以避免开关管烧毁。

死区发生器是在大功率设备PWM控制中常采用的一种技术，该功能用于在一个开关设备的断开和另一个开关设备的闭合之间插入一个时间间隔，这个时间间隔使得两个开关设备不可能同时被打开，即使是很短的一段时间也是如此。

S3C2410的timer0具有死区发生器功能，可用于控制大功率设备。

死区发生器开启前后输出波形对比



（5）DMA请求模式

S3C2410中定时器的DMA功能：系统中的5个定时器都有DMA请求功能，但是在同一时刻只能设置一个使用DMA功能，通过设置其DMA模式位来实现。

DMA请求过程：定时器可以在任意时间产生DMA请求，并且保持DMA请求信号(nDMA_REQ)为低直到定时器收到ACK信号。当定时器收到ACK信号时，它使请求信号变得无效。

DMA请求与中断的关系：如果一个定时器被配置为DMA模式，该定时器不会产生中断请求了；其他的定时器会正常的产生中断。

（6）计数时钟和输出计算

1）定时器输入时钟频率fTclk (即计数时钟频率)：

fTclk=[$f$pclk/(Prescaler+1)]x分频值

式中：Prescaler，预分频值，0--255；分频值为1/2、1/4、1/8、1/16。

2）PWM输出时钟频率：

PWM输出时钟频率= fTclk / TCNTBn

3）PWM输出信号占空比(即高电平持续时间所占信号周期的比例)：

PWM输出信号占空比 = TCMPBn/TCNTBn

（7）定时器最大、最小输出周期

设PCLK的频率为50MHz，经过预分频和分频器后，送给定时器的可能计数时钟频率由下表给出。

表　定时器最大、最小输出周期

	预分频器=0		预分频器=255	
分频值	最小输出周期 TCNTBn=1	最大输出周期 TCNTBn=65535	最小输出周期 TCNTBn=1	最大输出周期 TCNTBn=65535
1/2	25.00MHz(0.04μs)	381Hz	97656Hz	0.6710s
1/4	12.50MHz(0.08μs)	191Hz	48828Hz	1.3421s
1/8	6.250MHz(0.16μs)	95Hz	24414Hz	2.6843s
1/16	3.125MHz(0.32μs)	48Hz	12207Hz	5.3686s

2. 定时器专用寄存器

S3C2410定时器共有6种17个控制寄存器：

Register	Address	R/W	Description	Reset Value
TCFG0	0x51000000	R/W	配置寄存器 0	0x00000000
TCFG1	0x51000004	R/W	配置寄存器 1	0x00000000
TCON	0x51000008	R/W	控制寄存器	0x00000000
TCNTBn	0x510000xx	R/W	计数初值寄存器(5个)	0x0000
TCMPBn	0x510000xx	R/W	比较寄存器(4个)	0x0000
TCNTOn	0x510000xx	R	观察寄存器(5个)	0x0000

TCNTBn---Timern计数初值寄存器(计数缓冲寄存器)，16位
TCMPBn---Timern比较寄存器(比较缓冲寄存器)，16位
TCNTOn---Timern计数读出寄存器(观察缓冲寄存器)，16位

（1）TCFG0---预分频器配置寄存器

31	……	24	23	……	16	15	……	8	7	……	0
保留（为0）			Dead zone length			Prescaler1			Prescaler0		

Dead zone length---死区宽度设置位

其值N为： 0~255，以timer0的定时时间为单位

死区宽度为：(N+1)×timer0的定时时间

Prescaler1---timer2、3、4的预分频值

其值N为： 0~255

输出频率为：PCLK ÷(N+1)

Prescaler0--- timer0、1的预分频值

其值N为： 0~255

输出频率为：PCLK ÷(N+1)

（2）TCFG1---DMA模式与分频选择寄存器

31 … 24	23 … 20	19…16	15…12	11…8	7 … 4	3 … 0
保留（为0）	DMA mode	MUX4	MUX3	MUX2	MUX1	MUX0

DMA mode---DMA通道选择设置位

0000：不使用DMA方式，所有通道都用中断方式

0001：选择timer0　　0010：选择timer1

0011：选择timer2　　0100：选择timer3

0101：选择timer4　　011X：保留

MUX4~MUX0---timer4~timer0分频值选择(Clock Divider)

0000：1/2　0001：1/4　0010：1/8　0011：1/16

01XX：选择外部TCLK0、1(对timer0、1是选TCLK0-GPB4，对timer2、3、4是选TCLK1-GPG11)

（3）TCON---定时器控制寄存器

31…23	22	21	20	19	18	17	16	15	14	13
保留	TL4	TUP4	TR4	TL3	TO3	TUP3	TR3	TL2	TO2	TUP2

12	11	10	9	8	7…5	4	3	2	1	0
TR2	TL1	TO1	TUP1	TR1	保留	DZE	TL0	TO0	TUP0	TR0

TL4~TL0---计数初值自动重装控制位

0：单次计数

1：计数器值减到0时，自动重新装入初值连续计数。

TUP4~TUP0---计数初值手动装载控制位。

0：不操作　　1：立即将TCNTBn中的计数初值装载到计数寄存器TCNTn中。

说明：如果没有执行手动装载初值，则计数器启动时无初值。

(3) TCON---定时器控制寄存器(续)

31…23	22	21	20	19	18	17	16	15	14	13
保留	TL4	TUP4	TR4	TL3	TO3	TUP3	TR3	TL2	TO2	TUP2

12	11	10	9	8	7…5	4	3	2	1	0
TR2	TL1	TO1	TUP1	TR1	保留	DZE	TL0	TO0	TUP0	TR0

TR4~TR0---TIMER4~TIMER0运行控制位

0：停止　　1：启动对应的TIMER

TO3~TO0---TIMER4~TIMER0输出控制位

0：正相输出　　1：反相输出

DZE---TIMER0死区操作控制位

0：禁止死区操作　　1：使能死区操作

3.定时器的使用

1)定时器初始化方法

（1）写TCFG0，设置计数时钟的预分频值和Timer0死区宽度；

（2）写TCFG1，选择各个定时器的分频值和DMA、中断服务；

（3）对TCNTBn和TCMPBn分别写入计数初值和比较初值（它们决定了PWM定时器输出信号的占空比）；

（4）写TCON，设置计数初值自动重装、手动装载初值、设置反相输出；

（5）再写TCON，清除手动装载初值位、设置正相输出、启动计数。

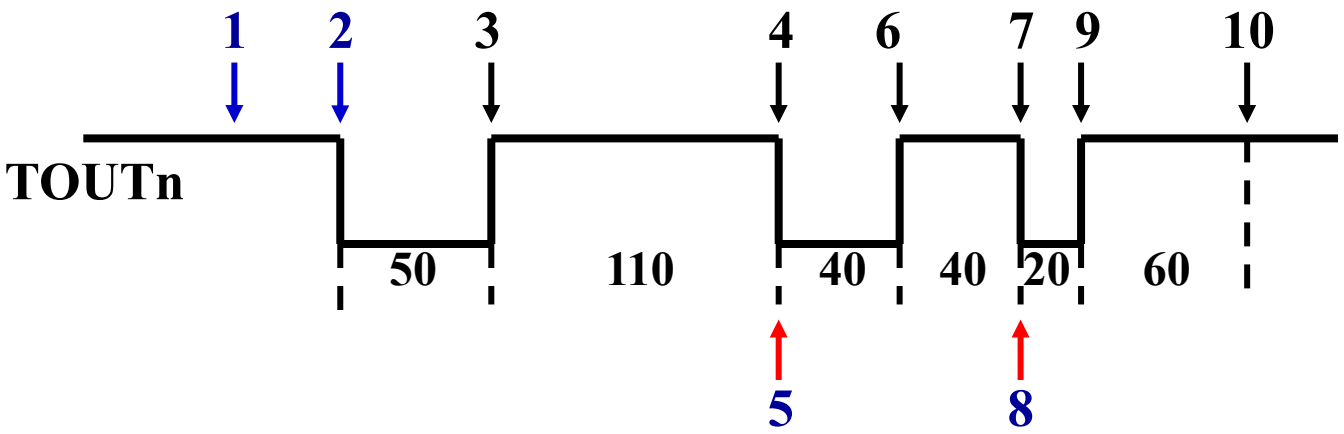
定时器的使用

2)定时器停止运行方法

写TCON，禁止计数初值自动重装。（一般不使用运行控制位停止运行）

3)定时器作操举例

① 初始化：TCNTBn=160(50+110)，TCMPBn=110；手动装入初值后，又重设TCNTBn=80， TCMPBn=40；
② 启动定时器，做减法计数；
③ 与第一个比较值相同，输出取反；
④ TCNTn等于0，第一次计数结束，产生中断，自动重装初值80、40；
⑤ 在第一次中断处理程序中重设TCMPBn=60；
⑥ 当TCNTn和TCMPn的值相等时，TOUTn输出电平由低变高；
⑦ 当TCNTn=0时，第二次计数结束，并触发第二次中断，把TCNTBn的值自动装入TCNTn；
⑧ 在第二次中断处理程序禁止自动重装初值，准备结束计数；
⑨ 当TCNTn和TCMPn的值相等时，TOUTn输出电平由低变高；
⑩ 第三次计数结束，不再计数。



定时器应用举例

```
#define BIT_TIMER0     (0x1<<10)
int  variable0,variable1,variable2,variable3,variable4;
void __irq Timer0Done(void)
{
   rSRCPND = BIT_TIMER0;       //Clear pending bit
   rINTPND = BIT_TIMER0;       //Clear serve bit
   variable0++;
}
void __irq Timer1Done(void)
{  rSRCPND = BIT_TIMER1;       //Clear pending bit
   rINTPND = BIT_TIMER1;
   variable1++;
}
……
```

```
#define pISR_TIMER0    (*(unsigned *)(_ISR_STARTADDRESS+0x48))
#define pISR_TIMER1    (*(unsigned *)(_ISR_STARTADDRESS+0x4c))
……
//在Test_TimerInt()中等待定时器中断产生
void Test_TimerInt(void)
{
   variable0 = 0; variable1 = 0; variable2 = 0;
   variable3 = 0; variable4 = 0;

   //将各中断向量写入中断向量表中
   pISR_TIMER0 = (int)Timer0Done;
   pISR_TIMER1 = (int)Timer1Done;
   pISR_TIMER2 = (int)Timer2Done;
   pISR_TIMER3 = (int)Timer3Done;
   pISR_TIMER4 = (int)Timer4Done;
   //开中断
   rINTMSK = ~(BIT_TIMER4 | BIT_TIMER3 |
                                          BIT_TIMER2 | BIT_TIMER1 | BIT_TIMER0);

   Uart_Printf("\n[ Timer 0,1,2,3,4 Interrupt Test ]\n\n");
```

```
rTCFG0 = rTCFG0 & ~(0xffffff) | 0x000f0f;        //死区宽度=0,
                                    //T1预分频=0x0f, T0预分频=0x0f
rTCFG1  =rTCFG1 & ~(0xffffff) | 0x001233;        //全部使用中断
                                                  //T4---T0分频值为：1/2,1/4, 1/8, 1/16, 1/16

                                    //设置T0---T4计数初值
rTCNTB0 = 0xffff;        // (1/(50MHz/16/16)) * 0xffff (65535) = 0.334s rTCNTB1 = 0xffff;
// (1/(50MHz/16/16)) * 0xffff (65535) = 0.334s rTCNTB2 = 0xffff;        // (1/(50MHz/16/8 ))
* 0xffff (65535) = 0.163s rTCNTB3 = 0xffff;        // (1/(50MHz/16/4 )) * 0xffff (65535) =
0.078s rTCNTB4 = 0xffff;        // (1/(50MHz/16/2 )) * 0xffff (65535) = 0.039s

rTCON   = rTCON & ~(0xffffff) | 0x6aaa0a;    //Auto reload,
                                    //Inverter off, Manual update, Dead zone disable, Stop

rTCON   = rTCON & ~(0xffffff) | 0x599901;   //Auto reload
// (T0=One-shot), Inverter off, No operation, Dead zone disable, Start
while(variable0 == 0);
```

```
Delay(1);                    //To compensate timer error(<1 tick period)

if(variable4==8 && variable3==4 && variable2==2 &&
                                                variable1==1 && variable0==1)
        Uart_Printf("Timer 0,1,2,3,4 Interrupt Test --> OK\n");
else
        Uart_Printf("Timer 0,1,2,3,4 Interrupt Test --> Fail.....\n");
```

```
    Uart_Printf("Press any key to exit Timer interrupt test\n");
    while(!Uart_GetKey());

    rTCON  = 0x0;            //One-shot, Inverter off, No operation, Dead
                                                    zone disable, Stop
    Uart_Printf("Timers interrupt number is as below:\n");

    Uart_Printf("Timer0 - %d ,Timer1 - %d ,Timer2 - %d ,
            Timer3 - %d ,Timer4 - %d \n", variable0,variable1,
                                        variable2,variable3,variable4);

    rINTMSK | = (BIT_TIMER4 | BIT_TIMER3 | BIT_TIMER2 | BIT_TIMER1 |
    BIT_TIMER0);
}
```

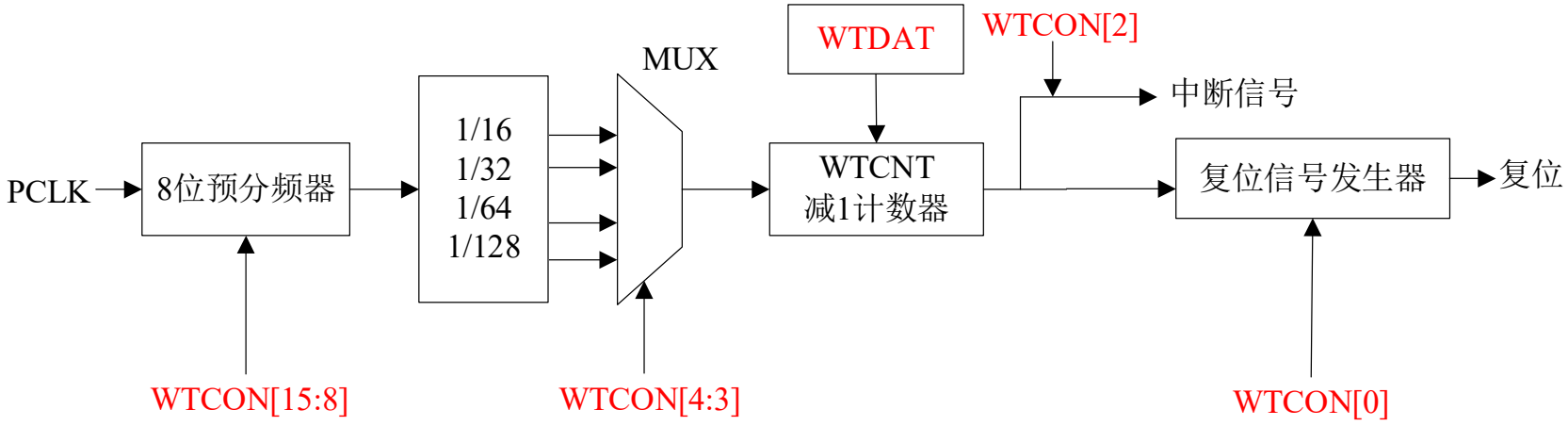
```
#define BIT_TIMER0    (0x1<<10)
#define BIT_TIMER1    (0x1<<11)
#define BIT_TIMER2    (0x1<<12)
#define BIT_TIMER3    (0x1<<13)
#define BIT_TIMER4    (0x1<<14)
```

6.4.2 看门狗定时器

1. 概念

当嵌入式系统运行时可能受到外部干扰或系统错误，程序有时会出现“跑飞”，导致整个系统瘫痪。

在系统稳定性要求较高的场合，为了防止这一现象的发生，专门设计了“看门狗”(Watchdog)电路，当系统“跑飞”而进入死循环时，产生复位信号，恢复系统的运行。



2. S3C2410 看门狗的功能

S3C2410 的看门狗定时器有两个功能：

（1）定时器功能：可以作为常规定时器使用，它是一个十六位的定时器，并且可以产生中断，中断名为INT_WDT，中断号(中断偏移量)是0x09。

（2）复位功能：作为看门狗定时器使用，当时钟计数减为0(超时)时，它将产生一个128个时钟周期的复位信号nReset（用于ARM920T核的复位）。

3. S3C2410 看门狗定时时间

预分频器(Prescaler)为8位，值为：0-255

分频器可选择值为：16、32、64、128

输入到计数器的时钟周期T_wtd(计数时钟周期)为：

T_wtd=1/[PCLK/(Prescaler+1)/Division_factor]

看门狗的定时周期为：

$$T = WTDAT \times T\_wtd$$

注意：一旦看门狗的定时器启动工作，其数据寄存器（WTDAT）中的值将不会自动读到时间寄存器 (WTCNT)中去。因此，必须在看门狗定时器启动之前，将一个初始值写入到看门狗的时间计数器(WTCNT)中间去，即先对时间计数器(WTCNT)赋初值，再启动看门狗工作。

4. 看门狗专用寄存器

看门狗控制寄存器共3个，分别是控制寄存器WTCON、数据寄存器WTDAT、计数寄存器WTCNT。

寄存器	地址	描　述	初值
WTCON	0x53000000	看门狗控制寄存器	0x8021
WTDAT	0x53000004	看门狗数据寄存器	0x8000
WTCNT	0x53000008	看门狗计数寄存器	0x8000

1）控制寄存器（WTCON）

字段名	位	意　　义	初值
Prescaler Value	15:8	预分频值。0--255。	0x80
Reserved	7:6	保留（为0）	00
Watchdog Timer	5	看门狗控制位。 0：禁止；1：允许	1
Clock Select	4:3	分频值选择。 00：16； 01：32；10：64；11：128	00
Interrupt Generation	2	看门狗中断控制。 0：禁止； 1：允许。	0
Reserved	1	保留（为0）	0
Reset Enable	0	看门狗复位功能控制。 0：禁止； 1：允许。	1

2）看门狗数据寄存器（WTDAT）

- 该数据寄存器为看门狗计数器重装计数值。初始值为0x8000；
- 初始化看门狗操作中，WTDATA不会自动加载到WTCNT定时计数器中，需要手动对WTCNT赋初值；
- 在计数溢出后，WTDAT的值将被装载到WTCNT 寄存器中。

3）看门狗计数寄存器（WTCNT）

字段名	位	意　　义	初值
Count Value	15:0	看门狗的当前计数值	0x8000

说明：在计数过程中只能读，不能写（即写不起作用）。

【例】设S3C2410的PCLK为50MHz，编写一程序，利用S3C2410看门狗中断产生频率为1kHz的方波，并且从GPB0引脚输出。

解：(1)计算看门狗数据寄存器值(WTDAT)

1）取分频值为16，分频后的频率为：

50M/16 = 3125000Hz

2）取预分频值为25，分频后的频率为：

3125000/25 = 125000Hz

3）取计数值为60，则计数器的频率为：

125000/60 = 2083.3Hz

4）方波频率为：

2083.3/2 = 1042Hz

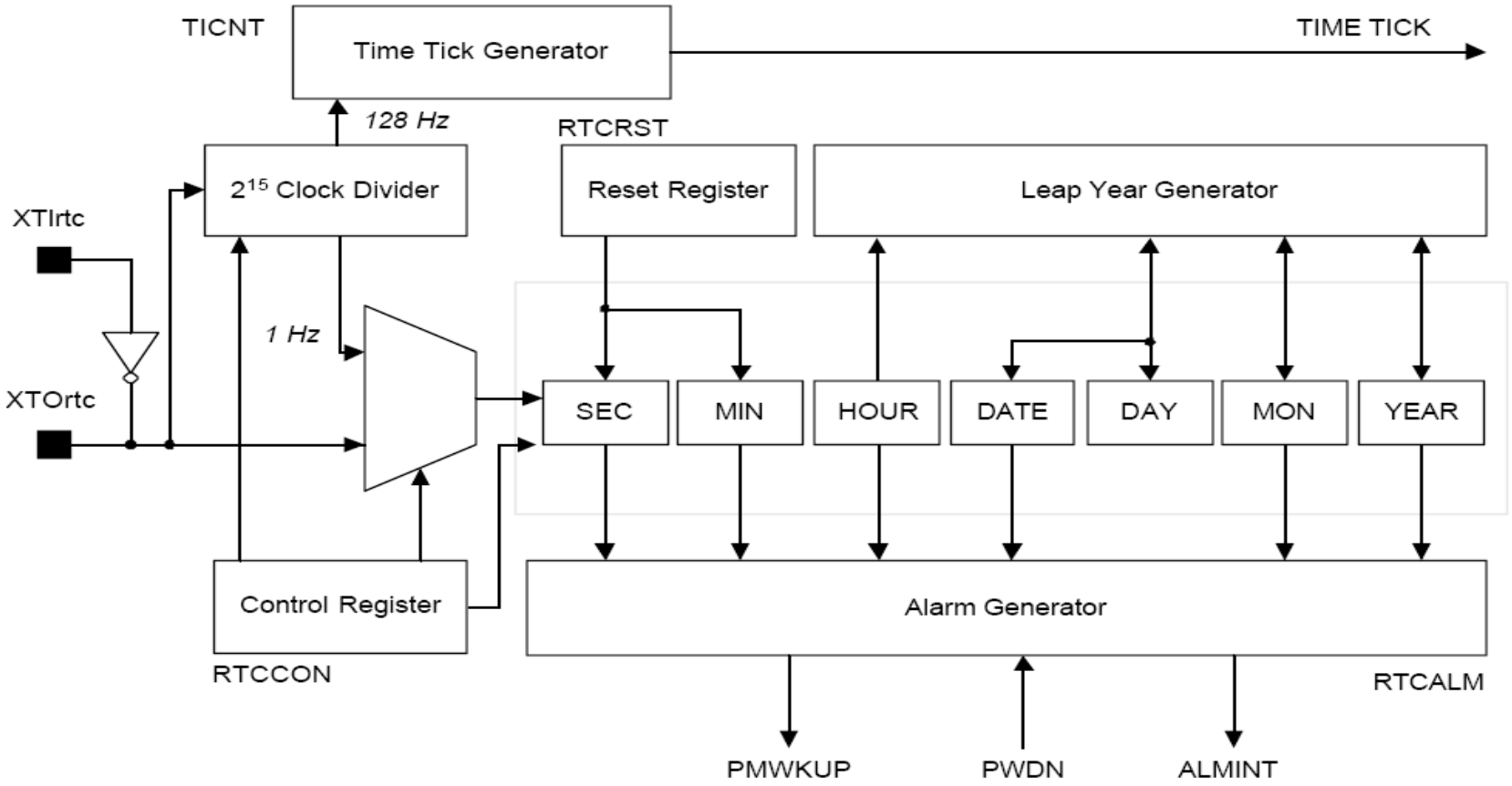
不可能实现精确的1000Hz方波。

```
BIT_WDT    EQU      (0x1<<9)
#include <string.h>
#include "2410addr.h"
#include "2410lib.h"
#include "timer.h"
void __irq Wdt_Int(void);
void Test_WDT_IntReq(void)
{
        Uart_Printf("WatchDog Timer Interrupt Request Test！\n");
        rINTMSK  &= ~(BIT_WDT);   //开看门狗中断
        pISR_WDT = (unsigned)Wdt_Int;          //设置中断向量
        rGPBCON = (rGPBCON|0x03)&0xfffffffd;          //把GPB0设为输出
        rWTCON   = 0x1804;          //写控制寄存器
        rWTDAT   = 60;       //写数据寄存器
        rWTCNT   = 60;       //写计数器
        rWTCON   = rWTCON | (1<<5);  //看门狗定时器使能(启动WTD)
        Uart_Printf("Press any Key to Exit!\n");
        Uart_Getch();        //等待按键
        rWTCON & = ~(1<<5);          //关闭看门狗定时器
        rINTMSK | =  (BIT_WDT);      //屏蔽看门狗中断
}
```

```
void _irq Wdt_Int(void)
{
    rGPBDATA ^ = 0x01;        //对GPB0取反
    rSRCPND | = BIT_WDT;   //清除看门狗中断请求标志
    rINTPND | = BIT_WDT;   //清除看门狗中断服务标志
}
```

6.4.2 RTC 部件

RTC（Real Time Clock: 实时时钟）功能：提供年、月、日、时、分、秒、星期等实时时间信息，且在系统关机状态下也能够正常工作（由后备电池供电）。



S3C2410的RTC的特点：

- 年、月、日、时、分、秒、星期 时钟数据采用BCD 编码
- 闰年的年月日进行自动处理
- 告警功能，当系统处于关机状态时，能产生告警中断
- 具有独立的电源输入（RTCVDD）
- 提供毫秒级时钟中断，可用作实时操作系统的内核时钟
- 进位复位功能

注：当RTC时间片计数器的值变为0时，引起中断(INT_TICK)，可用来产生实时操作系统内核所需的时间片。

中断信号周期=(n+1)/128 (s)，n=1～127

RTC专用寄存器

RTC有17个专用寄存器，按字节读写。

Register	Address	R/W	Description	Reset Value
RTCCON	0x57000040/3	R/W	RTC控制寄存器	0x0
TICNT	0x57000044/7	R/W	RTC时间片计数器	0x00
RTCALM	0x57000050/3	R/W	RTC报警控制寄存器	0x00
RTCRST	0x5700006C/F	R/W	RTC循环复位寄存器	0x0
ALMSEC	0x57000054/7	R/W	报警秒数寄存器	0x00
ALMMIN	0x57000058/B	R/W	报警分钟数寄存器	0x00
ALMHOUR	0x5700005C/F	R/W	报警小时数寄存器	0x00
ALMDAY	0x57000060/3	R/W	报警天(日)数寄存器	0x01
ALMMON	0x57000064/7	R/W	报警月数寄存器	0x01
ALMYEAR	0x57000068/B	R/W	报警年数寄存器	0x00

Register	Address	R/W	Description	Reset Value
BCDSEC	0x57000070/3	R/W	秒当前值寄存器	0xXX
BCDMIN	0x57000074/7	R/W	分钟当前值寄存器	0xXX
BCDHOUR	0x57000078/B	R/W	小时当前值寄存器	0xXX
BCDDAY	0x5700007C/F	R/W	日当前值寄存器	0xXX
BCDDATE	0x57000080/3	R/W	星期当前值寄存器	0xXX
BCDMON	0x57000084/7	R/W	月当前值寄存器	0xXX
BCDYEAR	0x57000088/B	R/W	年当前值寄存器	0xXX

1. RTC控制寄存器（RTCCON）

Register	Address	R/W	Description	Reset Value
RTCCON	0x57000040(L) 0x57000043(B)	R/W (字节)	RTC控制寄存器	0x0

字段名	位	意　　义	初值
CLKRST	3	RTC时钟计数复位。 0：不复位；1 = BCD计数复位。	0
CNTSEL	2	BCD计数选择。0: 合并BCD计数； 1 = 保留(单独的BCD计数器)	0
CLKSEL	1	BCD时钟选择。 0: XTAL/32768 (2^{15}) 1：用XTAL原值（但只用于测试）	0
RTCEN	0	RTC控制使能。 0: 禁止；1: 使能，BCD时间计数可以读取	0

2. RTC时间片计数器（TICNT）

Register	Address	R/W	Description	Reset Value
TICNT	0x57000044(L) 0x57000047(B)	R/W (字节)	RTC节拍时间计数器	0x00

字段名	位	意　义	初值
TICK INT ENABLE	7	时间片中断使能。 0：禁止，1：使能 。	0
TICK TIME COUNT	6:0	时间片计数器的值（1~127）。	000 0000

说明：该计数器为减1计数，用户不能在计数过程中读取该计数器的值。

3. RTC报警控制寄存器（RTCCON）

Register	Address	R/W	Description	Reset Value
RTCALM	0x57000050(L) 0x57000053(B)	R/W (字节)	RTC报警 控制寄存器	0x0

字段名	位	意　　义	初值
Reserved	7	保留（为0）	0
ALMEN	6	报警总使能位。0：禁止；1：使能	0
YEAREN	5	年报警使能位。0：禁止；1：使能	0
MONEN	4	月报警使能位。0：禁止；1：使能	0
DATEEN	3	日报警使能位。0：禁止；1：使能	0
HOUREN	2	时报警使能位。0：禁止；1：使能	0
MINEN	1	分报警使能位。0：禁止；1：使能	0
SECEN	0	秒报警使能位。0：禁止；1：使能	0

4. RTC报警秒数寄存器（ALMSEC）

Register	Address	R/W	Description	Reset Value
ALMSEC	0x57000054(L) 0x57000057(B)	R/W (字节)	报警秒数 寄存器	0x00

字段名	位	意　　义	初值
Reserved	7	保留（为0）	0
ALMSECH	6:4	报警时间秒十位，BCD值。0 ~ 5	000
ALMSECL	3:0	报警时间秒个位，BCD值。0 ~ 9	0000

5. 报警时间分钟数寄存器（ALMMIN）

Register	Address	R/W	Description	Reset Value
ALMMIN	0x57000058(L) 0x5700005B(B)	R/W (字节)	报警分钟数 寄存器	0x00

字段名	位	意　　义	初值
Reserved	7	保留（为0）	0
ALMMINH	6:4	报警时间分钟十位， BCD值。0 ~ 5	000
ALMMINL	3:0	报警时间分钟个位， BCD值。0 ~ 9	0000

6. 报警时间小时数寄存器（ALMHOUR）

Register	Address	R/W	Description	Reset Value
ALMHOUR	0x5700005C(L) 0x5700005F(B)	R/W (字节)	报警小时 寄存器	0x00

字段名	位	意　　义	初值
Reserved	7:6	保留（为0）	00
ALMHOURH	5:4	报警时间小时十位， BCD值。0 ~ 2	00
ALMHOURL	3:0	报警时间小时个位， BCD值。0 ~ 9	0000

7. RTC报警天数寄存器（ALMDATE）

Register	Address	R/W	Description	Reset Value
ALMDATE	0x57000060(L) 0x57000063(B)	R/W (字节)	报警日期 天数寄存器	0x01

字段名	位	意　　义	初值
Reserved	7:6	保留（为0）	00
ALMDATEH	5:4	报警日期天数十位， BCD值。0 ~ 3	00
ALMDATEL	3:0	报警日期天数个位， BCD值。0 ~ 9	0001

8. 报警时间月数寄存器（ALMMON）

Register	Address	R/W	Description	Reset Value
ALMMON	0x57000064(L) 0x57000067(B)	R/W (字节)	报警日期 月数寄存器	0x01

字段名	位	意　　义	初值
Reserved	7:5	保留（为0）	000
ALMMONH	4	报警日期月数十位， BCD值。0 ~ 1	0
ALMMONL	3:0	报警日期月数个位， BCD值。0 ~ 9	0001

9. 报警时间年数寄存器（ALMYEAR）

Register	Address	R/W	Description	Reset Value
ALMYEAR	0x57000068(L) 0x5700006B(B)	R/W (字节)	报警年数 寄存器	0x00

字段名	位	意　　义	初值
ALMYEARH	7:4	报警日期年数十位， BCD值。0 ~ 9	0000
ALMYEARL	3:0	报警日期年数个位， BCD值。0 ~ 9	0000

说明： 年数的千位和百位应该是20。

10. 秒循环复位寄存器（RTCRST）

Register	Address	R/W	Description	Reset Value
RTCRST	0x5700006C(L) 0x5700006F(B)	R/W (字节)	秒循环复位寄存器	0x00

字段名	位	意　　义	初值
Reserved	7:4	保留（为0）	000
SRSTEN	3	秒循环复位控制位。 0：禁止；1：允许	0
SECCR	2:0	秒循环进位周期。 011：30秒； 100：40秒；101：50秒。	00

说明：对于秒循环进位周期设为其它值(0,1,2,6,7)，设定只复位，不会向分钟进位。

11. 当前时间秒数寄存器（BCDSEC）

Register	Address	R/W	Description	Reset Value
BCDSEC	0x57000070(L) 0x57000073(B)	R/W (字节)	当前时间 秒数寄存器	-

字段名	位	意　　义	初值
Reserved	7	保留（为0）	-
NOWSECH	6:4	当前时间秒十位， BCD值。0 ~ 5	-
NOWSECL	3:0	当前时间秒个位， BCD值。0 ~ 9	-

12. 当前时间分钟寄存器（BCDMIN）

Register	Address	R/W	Description	Reset Value
BCDMIN	0x57000074(L) 0x57000077(B)	R/W (字节)	当前时间 分钟寄存器	-

字段名	位	意　　义	初值
Reserved	7	保留（为0）	-
NOWMINH	6:4	当前时间分钟十位， BCD值。0 ~ 5	-
NOWMINL	3:0	当前时间分钟个位， BCD值。0 ~ 9	-

13. 当前时间小时数寄存器（BCDHOUR）

Register	Address	R/W	Description	Reset Value
BCDHOUR	0x57000078(L) 0x5700007B(B)	R/W (字节)	当前时间 小时寄存器	-

字段名	位	意　　义	初值
Reserved	7:6	保留（为0）	-
NOWHOURH	5:4	当前时间小时十位， BCD值。0 ~ 2	-
NOWHOURL	3:0	当前时间小时个位， BCD值。0 ~ 9	-

14. 当前日期天数寄存器（BCDDATE）

Register	Address	R/W	Description	Reset Value
BCDDATE	0x5700007C(L) 0x5700007F(B)	R/W (字节)	当前日期 天数寄存器	-

字段名	位	意　　义	初值
Reserved	7:6	保留（为0）	-
NOWDATEH	5:4	当前日期天数十位， BCD值。0 ~ 3	-
NOWDATEL	3:0	当前日期天数个位， BCD值。0 ~ 9	-

15. 当前星期寄存器（BCDDAY）

Register	Address	R/W	Description	Reset Value
BCDDAY	0x57000080(L) 0x57000083(B)	R/W (字节)	当前星期 寄存器	-

字段名	位	意　　义	初值
Reserved	7:3	保留（为0）	-
NOWDAY	2:0	当前星期值。1 ~ 7	-

1：星期日；　2：星期一；　3：星期二；
4：星期三；　5：星期四；　6：星期五；
7：星期六

16. 当前日期月数寄存器（BCDMON）

Register	Address	R/W	Description	Reset Value
BCDMON	0x57000084(L) 0x57000087(B)	R/W (字节)	报警日期 月数寄存器	0x01

字段名	位	意　　义	初值
Reserved	7:5	保留（为0）	000
NOWMONH	4	当前日期月数十位， BCD值。0 ~ 1	0
NOWMONL	3:0	当前日期月数个位， BCD值。0 ~ 9	0001

17. 当前日期年数寄存器（BCDYEAR）

Register	Address	R/W	Description	Reset Value
BCDHOUR	0x57000088(L) 0x5700008B(B)	R/W (字节)	当前日期 年数寄存器	0x00

字段名	位	意　　义	初值
NOWYEARH	7:4	当前日期年数十位， BCD值。0 ~ 9	0000
NOWYEARL	3:0	当前日期年数个位， BCD值。0 ~ 9	0000

说明：年数的千位和百位应该是20

S3C2410 RTC使用方法

1. 读/写寄存器

（1）设置允许读写：对寄存器RTCCON[0]位(RTCEN)写1 。

（2）显示时间、日期方法：需要不断地从BCDSEC、BCDMIN、BCDHOUR、BCDDAY、BCDDATE、BCDMON和BCDYEAR 寄存器读取数据，然后显示出来。

（3）可能会引起显示错误。

例如，假设用户在2006 年12月31日23点59分59秒读取寄存器BCDYEAR 到BCDSEC，在用户读取BCDSEC 寄存器时，如果结果是0，那么很有可能年、月、日、时、分已经变成了2006年1月1日0时0分了，数据组合在一起可能是错的。读取的数据可能是：2006年12月1日0时0分 或 2006年1月1日0时0分。

解决的方法：当读取到的BCDSEC 等于0 时，用户应该再读取一次BCDYEAR到BCDSEC 的值。

2. 报警功能

RTC的报警寄存器(RTCALM)决定了报警的使能、禁止以及报警时间设定的条件。

在RTC报警使能情况下：

(1) 在正常工作模式下，报警中断(ALMINT)是激活状态的。

(2)在掉电模式下(PWDN信号有效)，电源管理唤醒信号(PMWKUP)与报警中断（ALMINT)都是激活状态。

3.时间片中断（节拍中断：INT_TICK)

RTC时间片用于中断请求。

TICNT 寄存器：有中断使能位、时间片计数位。

当时间片计数器中的值到达0 时，就会触发时间片中断INT_TICK。时间片中断的间隔时间计算如下：

Period=(n+1)/128 秒 ， n：时间片计数值(1~127)

说明：RTC时间片中断可以作为RTOS（实时操作系统）内核的时间节拍。

【例】编写一程序，对S3C2410的RTC进行设置，使用时间片中断（节拍中断），每1秒中断一次，中断后显示出当前的日期和时间。初始日期、时间设置为正确值。

答：1. 设置控制寄存器： RTCCON=0x0001 = 0x01

含义：RTC不复位、正常BCD计数、BCD时钟选择为1/32768、允许读出RTC值;

2. 设置时间片计数器：TICNT=0x11111111 = 0xFF

含义：允许时间片中断、时间片计数器的值为127，每1秒钟中断一次。

```
BIT_TICK        EQU     (0x1<<8)
BIT_ALLMSK      EQU     (0xffffffff)
BIT_RTC         EQU     (0x1<<30)

#include "2410addr.h"
#include "2410lib.h"

void Test_Rtc_Tick(void);
void _irq Rtc_Tick(void);
void Test_Rtc_Tick(void)
{
   Uart_Printf("RTC Tick interrupt
                    test for S3C2410！ \n");
   pISR_TICK = (unsigned)Rtc_Tick;  //设置中断服务程序地址
   rINTMSK &  = ~(BIT_TICK);          // (0x1<<8), 开时间片中断
```

```
//设置日期与时间
  rBCDYEAR = 0x07;
  rBCDMON  = 0x05;
  rBCDDAY  = 0x03;            //SUN:1 MON:2 TUE:3 WED:4 THU:5 FRI:6 SAT:7
  rBCDDATE = 0x22;
  rBCDHOUR = 0x08;
  rBCDMIN  = 0x38;
  rBCDSEC  = 0x25;
  rTICNT   = (1<<7) + 127; //设置时间片值，Period = (n + 1)/128 =1sec
  rRTCCON  = 0x01;     //启动计时、允许读写
  Uart_Printf("Press any key to exit.\n");
  Uart_Getch();   //等待按键
  rINTMSK  | = BIT_TICK; //(1<<8)    //关闭时间片中断
}
```

```
void __irq Rtc_Tick(void)
{
        char     year,month,date,hour,min,sec;        //读取年月日
        year    = rBCDYEAR;
        month   = rBCDMON & 0x1F;
        date    = rBCDDATE & 0x3F;
        hour    = rBCDHOUR & 0x3F;  //读取时分秒
        min     = rBCDMIN & 0x7F;
        sec     = rBCDSEC & 0x7F;       //显示日期、时间
        Uart_Printf("20%2x年%2x月%2x日,%2x：%2x：%2x\n",
year, month ,date hour,min,sec);
        rSRCPND | = BIT_TICK;           //清除中断请求标志
        rINTPND | = BIT_TICK;           //清除中断请求标志
}
```

课外作业

第2题. 描述IRQ异常的处理过程。

第3题. 在S3C2410芯片中采用中断方式控制I/O端口或部件操作时，其中断处理编程应该涉及哪些方面？举例说明。

第5题. S3C2410芯片的看门狗电路有哪些工作方式？若希望系统程序的周期不大于50us，fpclk=100MHz，写出相应的看门狗初始化程序。

第8题. 若需要利用S3C2410芯片Timer部件中的Timer2通道产生一个周期约为1000ms的脉冲信号，系统的频率fpclk=66MHz，写出初始化程序。

End of Chapter 6