10. 基于Cortex-A9的pwm详解

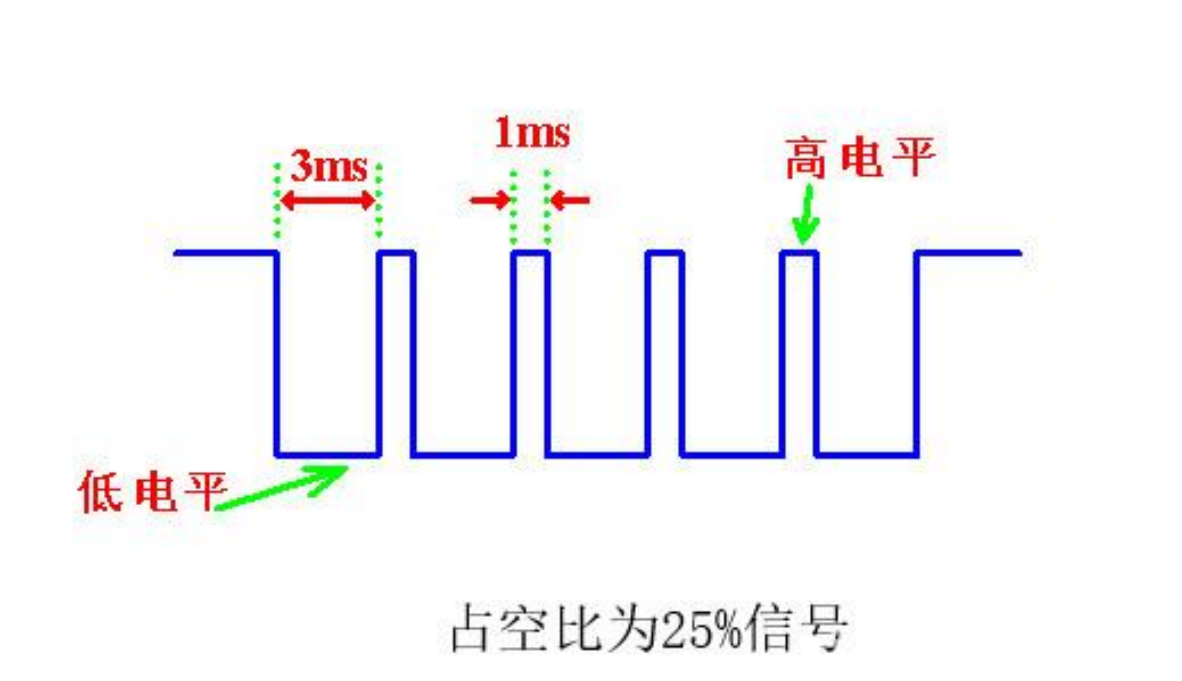
原创 土豆居士 一口Linux 2020-12-24 11:50

收录于合集

#所有原创 206 #从0学arm 27

一、什么是PWM

PWM，英文名Pulse Width Modulation，是脉冲宽度调制缩写，它是通过对一系列脉冲的宽度进行调制，等效出所需要的波形（包含形状以及幅值），对模拟信号电平进行数字编码，也就是说通过调节占空比的变化来调节信号、能量等的变化，占空比就是指在一个周期内，信号处于高电平的时间占据整个信号周期的百分比，例如方波的占空比就是50%.



二、PWM信号输出输出和作用

1. 如果要实现PWM信号输出如何输出呢?

1）可以直接通过芯片内部模块输出PWM信号，前提是这个I/O口要有集成的pwm控制器，只需要通过对应的寄存器即可，这种自带有PWM输出的功能模块在程序设计更简便，同时数据更精确。

2）但是如果IC内部没有PWM功能模块，或者要求不是很高的话可以利用I/O口设置一些参数来输出PWM信号，因为PWM 信号其实就是一高一低的一系列电平组合在一起。具体方法是给I/O加一个定时器，对于你要求输出的PWM信号频率与你的定时器一致，用定时器中断来计数，但是这种方法一般不采用，除非对于精度、频率等要求不是很高可以这样实现。

2. PWM信号应用

PWM信号把模拟信号转化为数字电路所需要的编码，现在基本是采用数字电路，因此在很多场合都采用PWM信号。

我们经常见到的就是**交流调光电路**，也可以说是无级调速，高电平占多一点，也就是占空比大一点亮度就亮一点，占空比小一点亮度就没有那么亮，前提是PWM的频率要大于我们人眼识别频率，要不然会出现闪烁现象。

除了在调光电路应用，还有在**直流斩波电路、蜂鸣器驱动、电机驱动、逆变电路、加湿机雾化量**等都会有应用。

三、蜂鸣器

蜂鸣器广泛用于计算机、打印机、复印机、报警器、电子玩具、汽车电子设备、电话机、定时器等电子产品中作发声器件。

蜂鸣器分为压电式及电磁式的二大类：

1. 压电式蜂鸣器主要由多谐振荡器、压电蜂鸣片、阻抗匹配器及共鸣箱、外壳等组成。它是以压电陶瓷的压电效应，来带动金属片的振动而发声；

2. 电磁式的蜂鸣器，由振荡器、电磁线圈、磁铁、振动膜片及外壳等组成。接通电源后，振荡器产生的音频信号电流通过电磁线圈，使电磁线圈产生磁场。振动膜片在电磁线圈和磁铁的相互作用下，周期性地振动发声。通电时将金属振动膜吸下，不通电时依振动膜的弹力弹回。

有源蜂鸣器， 只要给它加上恒定的电压， 就能发声；无源蜂鸣器， 必须给它加上一定频率的方波或正弦波才能发声

有源蜂鸣器内部带震荡源，所以一通电就会叫。而无源内部不带震荡源，所以如果用直流信号无法令其鸣叫。

有源蜂鸣器往往比无源的贵，就是因为里面多个震荡电路。

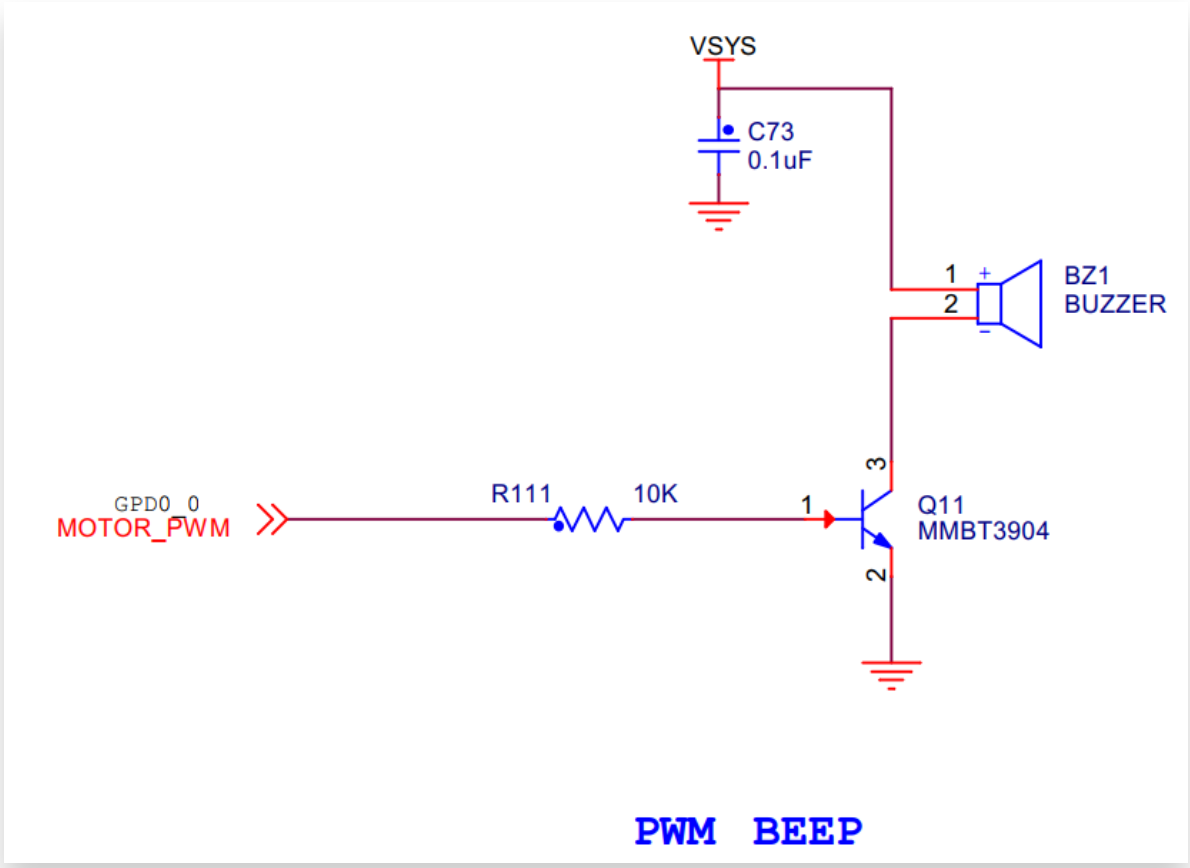
【优点】无源蜂鸣器的优点是：

1. 便宜
2. 声音频率可控，可以做出“多来米发索拉西”的效果。
3. 在一些特例中，可以和LED复用一个控制口 有源蜂鸣器的优点是：
4. 程序控制方便 。

应用：电风扇、收音机的声音按钮、任何模拟值都可以使用PWM进行编码

四、fs4412电路图

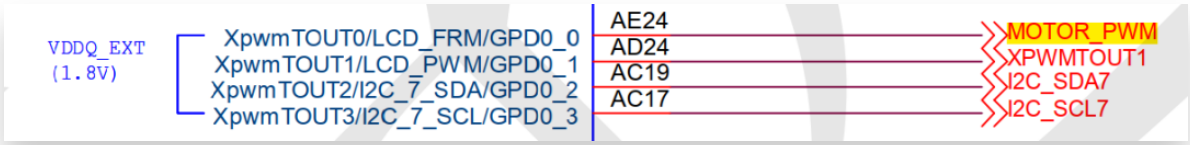
本例采用fs4412开发板，pwm外接了一个蜂鸣器BUZZER，电路图如下：



从上面电路图可知：

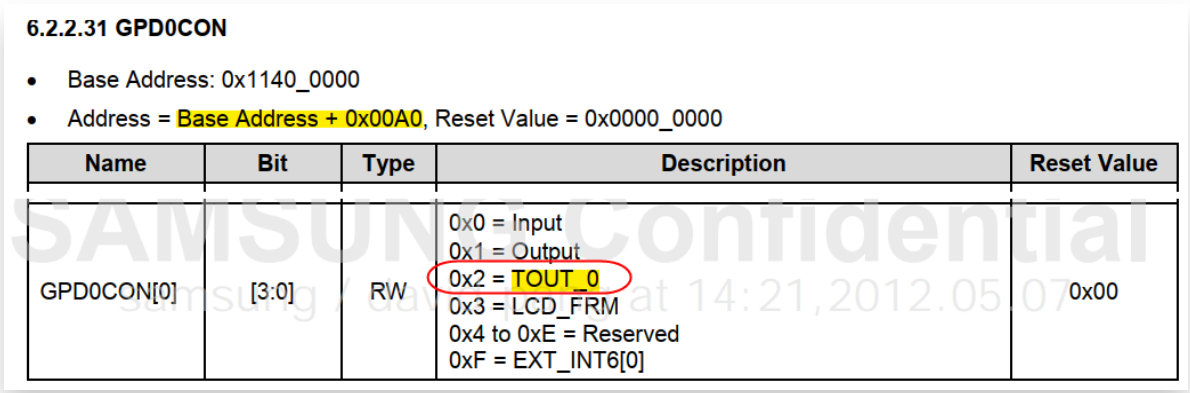
1. 该BUZZER是无源蜂鸣器，如果要想发出声音，需要正负极产生电流变化，我们通过生成方波，从而实现图中三极管1->2周期性导通和关闭来让BUZZER俩边电压产生变化，从而实现电流变化；
2. 三极管的基极连接的是SOC的GPD0_0引脚；
3. 产生方波我们借助的是PWM，标号为MOTOR_PWM。

继续查找MOTOR_PWM：



由上图可知，MOTOR_PWM连接的是PWM的XpwmTOUT0，和LCD一起复用引脚GPD0_0，

去datasheet继续查看GPD0_0说明，



6.2.2.31 GPD0CON

- Base Address: 0x1140_0000
- Address = Base Address + 0x00A0, Reset Value = 0x0000_0000

Name	Bit	Type	Description	Reset Value
GPD0CON[0]	[3:0]	RW	0x0 = Input 0x1 = Output 0x2 = TOUT_0 0x3 = LCD_FRM 0x4 to 0xE = Reserved 0xF = EXT_INT6[0]	0x00

由上图可知，GPD0_0配置由寄存器的GPD0CON[3:0]位控制，要想作为PWM输出，要设置为TOUT_0即0x2。

同时也可以看到，该引脚还可以设置为外部中断信号[EXT_INT6]功能即0xF。

五、Exynos 4412 PWM

概述

Exynos 4412 SCP有五个32位脉冲宽度调制（PWM）定时器。这些定时器产生内部中断对于ARM子系统。此外，定时器0、1、2和3包括驱动外部I/O的PWM功能信号。定时器0中的PWM有一个可选的死区发生器功能，以支持大量的设备。定时器4是一个没有输出引脚的内部定时器。

定时器使用APB-PCLK作为源时钟。定时器0和1共享可编程8位预分频器为PCLK提供第一级分频。定时器2、3和4共享不同的8位预分频器。每个计时器都有它自己的专用时钟分频器，提供第二级时钟划分频（预分频器除以2、4、8或16）。

每个定时器都有它的32位递减计数器；定时器时钟驱动这个计数器。定时器计数缓冲寄存器（TCNTBn）加载递减计数器的初始值。如果递减计数器达到零，它将生成计时器中断请求，通知CPU定时器操作完成。如果定时器下降计数器达到零，相应TCNTBn的值自动重新加载到下一个循环开始。但是，如果定时器停止，例如，在定时器运行模式下，通过清除TCONn的定时器使能位，TCNTBn的值将不会重新加载到计数器中。

PWM功能使用TCMPBn寄存器的值。定时器控制逻辑改变输出电平下计数器值与定时器控制逻辑中比较寄存器的值相匹配。因此，比较寄存器决定PWM输出的开启时间或关闭时间。

每个定时器都是双缓冲结构，带有TCNTBn和TCMPBn寄存器，允许定时器参数在周期中更新。新值在当前计时器周期完成之前不会生效。

Exynos ＰWM定时器的特性

１） ５个３２位定时器；

２） ２个８位PCLK分频器提供一级预分， ５个２级分频器用来预分外部时钟； ３）可编程选择PWM独立通道。

４） ４个独立的可编程的控制及支持校验的ＰWM通道。

５）静态配置：PWM停止；

６）动态配置：PWM启动；

７）支持自动重装模式及触发脉冲模式；

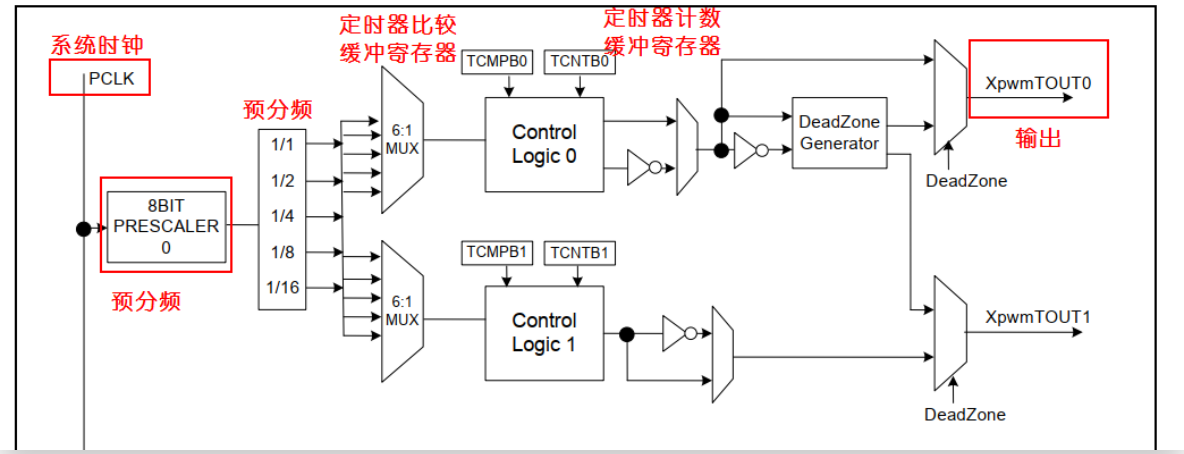
８）一个外部启动引脚。

９）两个PWM输出可带Dead-Zone 发生器。

１０）中断发生器。

PWM内部模块图

Figure 24-2 illustrates the clock generation scheme for individual PWM channels.

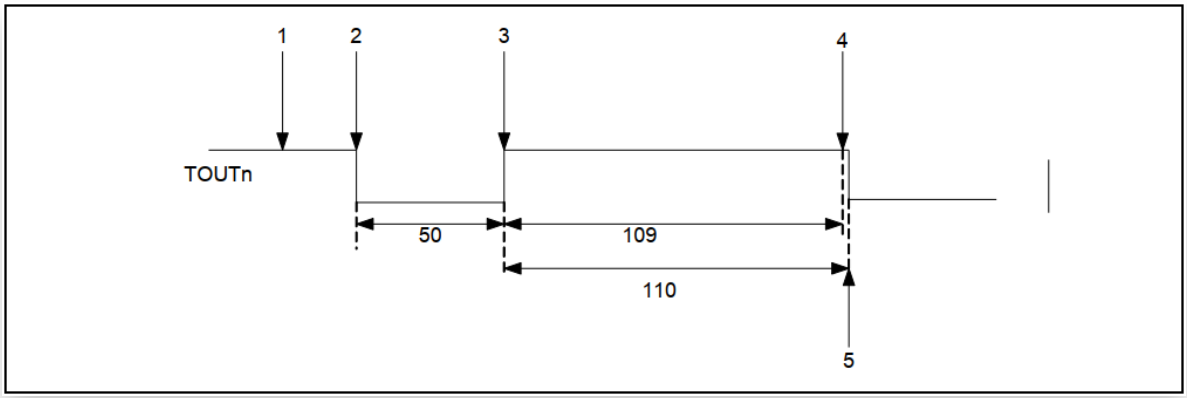


工作的步骤：

1. 当时钟PCLK被使能后，定时器计数缓冲寄存器（TCNTBn）把计数器初始值下载到递减计数器中。
2. 定时器比较缓冲寄存器（TCMPBn）把其初始值下载到比较寄存器中，并将该值与递减计数器的值进行比较。当递减计数器和比较寄存器值相同时，输出电平翻转。
3. 递减计数器减至0后，输出电平再次翻转，完成一个输出周期。这种基于TCNTBn和TCMPBn的双缓冲特性使定时器在频率和占空比变化时能产生稳定的输出。
4. 每个定时器都有一个专用的由定时器时钟驱动的16位递减计数器。当递减计数器的计数值达到0时，就会产生定时器中断请求来通知C P U定时器操作完成。当定时器递减计数器达到 0 的时候，如果设置了Auto-Reload 功能，相应的TCNTBn的值会自动重载到递减计数器中以继续下次操作。
5. 然而，如果定时器停止了，比如在定时器运行时清除TCON中定时器使能位，TCNTBn的值不会被重载到递减计数器中。
6. 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匹配的时候，定时器控制逻辑将改变输出电平。因此，比较寄存器决定了PWM 输出的开关时间。

举例

下面我们举个实例来看下，



1. 初始化寄存器 TCNTBn = 159 (50 + 109)，TCMPBn =109.
2. 开启定时器: 通过设置TCON的开启位. 寄存器TCNTBn 的值159将自动加载到递减寄存器down-counter, 同时输出引脚TOUTn 设置为低电平.
3. 当down-counter 的值递减打破和寄存器TCMPBn 的值109相同时, 输出引脚将从低拉到高.
4. 当down-counter递减到0时, 产生一个中断请求.
5. 如果我们设置成autoreload模式，那么down-counter会自动加载TCNTBn的值到down-counter，开启新的一个周期。

我们可以通过设置TCNTBn、TCMPBn来控制占空比，而每个pwm周期后都可以重新设置新的值到TCNTBn、TCMPBn，我们通过精确的计算来设置TCNTBn、TCMPBn的值并通过设置dead zone我们可以设计出各种复杂的矩形波。

如下图所示：

本例我们只需要产生规则的举行方波即可，所以我们只需要设置占空比为50%即可。

六、寄存器

由第四章可知，我们使用PWM控制器的timer 0，对应的寄存器组如下图所示：

24.5.1 Register Map Summary

- Base Address: 0x139D_0000 (PWM)
- Base Address: 0x1216_0000 (PWM_ISP)

Register	Offset	Description	Reset Value
TCFG0	0x0000	Specifies the timer configuration register 0 that configures the two 8-bit prescaler and dead-zone or dead zone length	0x0000_0101
TCFG1	0x0004	Specifies the timer configuration register 1 that controls five MUX select bit	0x0000_0000
TCON	0x0008	Specifies the timer control register	0x0000_0000
TCNTB0	0x000C	Specifies the timer 0 count buffer register	0x0000_0000
TCMPB0	0x0010	Specifies the timer 0 compare buffer register	0x0000_0000
TCNTO0	0x0014	Specifies the timer 0 count observation register	0x0000_0000

1、TFCG0

定时器配置寄存器0（TFCG0），主要用于预分频设置。

24.5.1.1 TCFG0

- Base Address: 0x139D_0000 (PWM)
- Base Address: 0x1216_0000 (PWM_ISP)
- Address = Base Address + 0x0000, Reset Value = 0x0000_0101

Name	Bit	Type	Description	Reset Value
RSVD	[31:24]	–	Reserved Bits	0x00
Dead zone length	[23:16]	RW	Dead zone length	0x00
Prescaler 1	[15:8]	RW	Prescaler 1 value for Timer 2, 3, and 4	0x01
Prescaler 0	[7:0]	RW	Prescaler 0 value for timer 0 and 1	0x01

我们是timer 0，所以只需要设置该寄存器的bite【7:0】即可，最终的输出频率和value的公式如下：

Timer Input Clock Frequency = PCLK/({prescaler value + 1})/{divider value}
{prescaler value} = 1 to 255
{divider value} = 1, 2, 4, 8, 16
Dead zone length = 0 to 254

参考24.3.1节：

24.3.1 Prescaler and Divider

An 8-bit prescaler and 3-bit divider generates these output frequencies:

Table 24-1 describes the minimum and maximum resolution based on prescaler and clock divider values.

Table 24-1 Minimum and Maximum Resolution Based on Prescaler and Clock Divider Values

4-bit Divider Settings	Minimum Resolution (Prescaler Value = 1)	Maximum Resolution (Prescaler Value = 255)	Maximum Interval (TCNTBn = 4294967295)
1/1 (PCLK = 66 MHz)	0.030us (33.0 MHz)	3.879us (257.8 kHz)	16659.27s
1/2 (PCLK = 66 MHz)	0.061us (16.5 MHz)	7.758us (128.9 kHz)	33318.53s
1/4 (PCLK = 66 MHz)	0.121us (8.25 MHz)	15.515us (64.5 kHz)	66637.07s
1/8 (PCLK = 66 MHz)	0.242us (4.13 MHz)	31.03us (32.2 kHz)	133274.14s
1/16 PCLK = 66 MHz)	0.485us (2.06 MHz)	62.061us (16.1 kHz)	266548.27s

其中方波的频率必须在音频范围内，也就是20Hz到20KHZ之间， 但是20Hz到20KHZ的频率送给蜂鸣器后， 只有某一点的频率是最响的， 这个频率称为蜂鸣器的谐振频率， 离它越远，蜂鸣器发出的声音越轻。

所以Prescaler 0 value值应该设置为255，divider value 应该是1/16,值由TCFG1设置。

```
PWM.TCFG0 = PWM.TCFG0 & (~(0xff))|0xf9;
```

2、TCFG1

定时器配置寄存器1（TCFG1） 主要用于PWM定时器的divider value设置。

24.5.1.2 TCFG1

- Base Address: 0x139D_0000 (PWM)
- Base Address: 0x1216_0000 (PWM_ISP)
- Address = Base Address + 0x0004, Reset Value = 0x0000_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:20]	–	Reserved	0x000
Divider MUX0	[3:0]	RW	Selects Mux input for PWM timer 0 0000 = 1/1 0001 = 1/2 0010 = 1/4 0011 = 1/8 0100 = 1/16	0x0

由上一节分析，秩序设置TCFG1 bite【3:0】为0100即0x2即可。

```
PWM.TCFG1 = PWM.TCFG1 & (~(0xf)) | 0x2;
```

3、TCON

timer控制寄存器TCON

24.5.1.3 TCON

- Base Address: 0x139D_0000 (PWM)
- Base Address: 0x1216_0000 (PWM_ISP)
- Address = Base Address + 0x0008, Reset Value = 0x0000_0000

Name	Bit	Type	Description	Reset Value
Reserved	[7:5]	–	Reserved Bits	0x0
Dead zone enable/disable	[4]	RW	Enables/Disables Dead zone generator	0x0
Timer 0 auto reload on/off	[3]	RW	0 = One-shot 1 = Interval mode (auto-reload)	0x0
Timer 0 output	[2]	RW	0 = Inverter Off	0x0
inverter on/off			1 = TOUT_0 inverter-on	
Timer 0 manual update	[1]	RW	0 = No operation 1 = Updates TCNTB0 andTCMPB0	0x0
Timer 0 start/stop	[0]	RW	0 = Stops Timer 0 1 = Starts Timer 0	0x0

1. bite[3]：设置定时器是只执行一个周期(One-shot)还是周期执行（auto-reload）
2. bite[1]: 置为1，则更新TCNTB0 、TCMPB0 的值
3. bit[0]：开启或者停止定时器

针对不同操作，我们可以设置不同的值：

1. 装载

```
PWM.TCON = PWM.TCON & (~(0xff)) | (1 << 0) | (1 << 1) ;
```

2. 开启定时器，蜂鸣器响

```
PWM.TCON = PWM.TCON & (~(0xff)) | (1 << 0) | (1 << 3) ;
```

3. 关闭定时器，蜂鸣器灭

```
PWM.TCON = PWM.TCON & (~(1 << 0)) ;
```

4、TCNTB0

定时器计数缓冲寄存器（TCNTB0） 根据测算，设置为100

24.5.1.4 TCNTB0

- Base Address: 0x139D_0000 (PWM)
- Base Address: 0x1216_0000 (PWM_ISP)
- Address = Base Address + 0x000C, Reset Value = 0x0000_0000

Name	Bit	Type	Description	Reset Value
Timer 0 count buffer	[31:0]	RW	Timer 0 Count Buffer register	0x0000_0000

TCNTB0

```
PWM.TCNTB0 = 100;
```

5、TCMPB0

定时器比较缓冲寄存器（TCMPB0 ） 设置为50,占空比为50%

24.5.1.5 TCMPB0

- Base Address: 0x139D_0000 (PWM)
- Base Address: 0x1216_0000 (PWM_ISP)
- Address = Base Address + 0x0010, Reset Value = 0x0000_0000

Name	Bit	Type	Description	Reset Value
Timer 0 compare buffer	[31:0]	RW	Timer 0 Compare Buffer register	0x0000_0000

```
 PWM.TCMPB0 = 50;
```

七、代码

完整代码后台回复**【armprintf】**。

```
#include "exynos_4412.h"
void  delay_ms(unsigned int num)
{   int i,j;
    for(i=num; i>0;i--)
 for(j=1000;j>0;j--);
}
void pwm_init(void)
{
 GPD0.CON = GPD0.CON & (~(0xf))| 0x2;
 PWM.TCFG0 = PWM.TCFG0 & (~(0xff))|0xf9;
 PWM.TCFG1 = PWM.TCFG1 & (~(0xf)) | 0x2;
 PWM.TCMPB0 = 50;
 PWM.TCNTB0 = 100;
 PWM.TCON = PWM.TCON & (~(0xff)) | (1 << 0) | (1 << 1) ;
}
void beep_on(void)
{
 PWM.TCON = PWM.TCON & (~(0xff)) | (1 << 0) | (1 << 3) ;
}
void beep_off(void)
{
 PWM.TCON = PWM.TCON & (~(1 << 0)) ;
}
#define SYS_SET_FREQUENCE 25000
void beep_set_frequence( unsigned int fre )
{ //若蜂鸣器的发声频率为0则返回
 if( 0==fre )
  return ;
 PWM.TCMPB0 =  SYS_SET_FREQUENCE/(fre+fre);//根据设定频率重新设定计数器比较的值
 PWM.TCNTB0 =  SYS_SET_FREQUENCE/fre; //根据频率重新调整计数值
}
int main (void)
{ pwm_init();
```

```
 while(1)
 {  beep_on();   //发出一个音
    delay_ms(100);
    beep_off();          //关闭蜂鸣器，  每个音播放完成后有间隔感
    delay_ms(100);
 }
    return 0;
}
```

我们也可以通过修改频率和音响的时间来播放音乐。

要获取源码可以加一口君好友！

其他网友提问汇总

1. 两个线程，两个互斥锁，怎么形成一个死循环？
2. 一个端口号可以同时被两个进程绑定吗？
3. 一个多线程的简单例子让你看清线程调度的随机性
4. 粉丝提问|c语言：如何定义一个和库函数名一样的函数，并在函数中调用该库函数
5. [网友问答5]i2c的设备树和驱动是如何匹配以及何时调用probe的？
6. [粉丝问答6]子进程进程的父进程关系
7. 【粉丝问答7】局域网内终端是如何访问外网？答案在最后

推荐阅读

【1】嵌入式工程师到底要不要学习ARM汇编指令？**必读**

【2】嵌入式工程师到底要不要学习ARM汇编指令？

【3】【从0学ARM】你不了解的ARM处理异常之道

【4】4. 从0开始学ARM-ARM汇编指令其实很简单

【5】为什么使用结构体效率比较高？ **必读**

【6】9. 基于Cortex-A9 LED汇编、C语言驱动编写**必读**

【7】一文包你学会网络数据抓包**必读**

进群，请加一口君个人微信，带你嵌入式入门进阶。

收录于合集 #从0学arm 27

上一篇
9. 基于Cortex-A9 LED汇编、C语言驱动编写

下一篇
11. 基于ARM Cortex-A9中断详解

喜欢此内容的人还喜欢

pinia
睡不着所以学编程

面试连环问--操作系统
阿Q正砖

Konva实现图片自适应裁剪
A逐梦博客