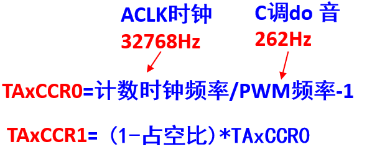
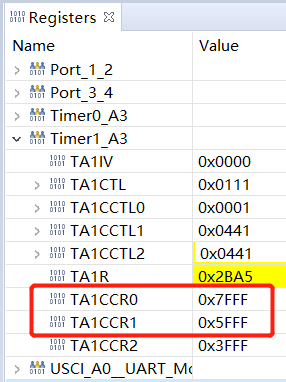
**实验6 定时器PWM输出及应用**

**一．实验目的**

1. 掌握利用定时器输出PWM波形方法；

****2. (提高)掌握低功耗模式控制方法。

1. **实验任务**
2. **利用定时器A比较输出功能，输出PWM波形**
3. 阅读TA\_PWM.c程序，了解定时器PWM波形实现原理；
4. 将P2.1和P2.4两引脚分别连到两发光二级管上，运行TA\_PWM.c程序，通过观察二级管闪烁的频率，以及亮、灭时间的相对长度，了解PWM输出波形的频率、占空比控制方法，其中占空比是指高电平占一个周期中的比例。可在DEBUG下，如图6-1，在View/Registers窗口下，修改TA1CCR0、TACCR1、TACCR2的置，观察现象，理解程序实现原理；（特点:同一个定时器的两个比较器）。
5. 在TA\_PWM.c的基础上，编程在P2.1 、P2.4上输出频率为0.5Hz、占空比为90%、10%的方波；

（特点：同一个定时器的两个比较器，改频率，改占空比）。

图6-1 修改定时器设置

1. 编程在P1.6引脚上输出频率为1Hz、占空比为50%的PWM波形;

在P2.5引脚上输出频率为0.5Hz、占空比为50%的PWM波形。

(特点：不同定时器，不同比较器)。

**注意：**PWM波形的输出本身与中断无关，只与定时器的工作设置有关。

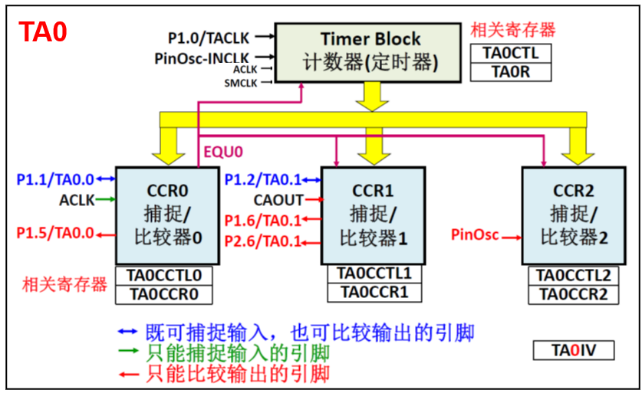
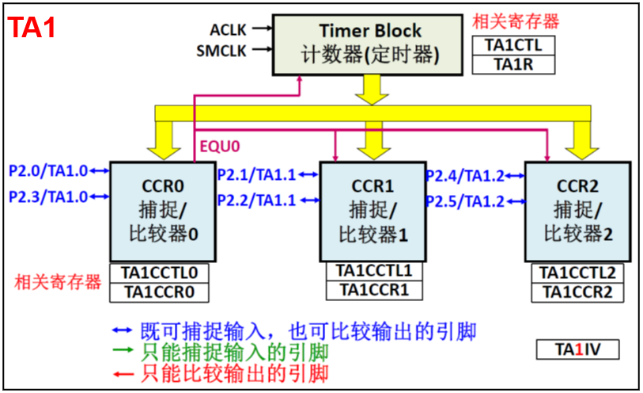


图6-2 msp430g2553定时器A引脚分配

TA\_PWM.c

#include "msp430.h"

int main ( void )

{ WDTCTL = WDTPW + WDTHOLD; //关闭看门狗

P2SEL |=BIT1+BIT4; //置P2.1和 P2.4为定时器TA1的PWM输出引脚

P2SEL2 &=~(BIT1+BIT4); //P2.1为比较器1的PWM输出引脚

P2DIR |=BIT1+BIT4; //P2.4为比较器2的PWM输出引脚

TA1CTL |=TASSEL0; //选择TA1计数时钟为ACLK，使用上电复位设置，即32768Hz

TA1CCR0=32767; //设置PWM周期，周期= (TA1CCR0+1)\*T =(TA1CCR0+1)/计数时钟频率

//即PWM频率=1/PWM周期=计数时钟频率/(TA1CCR0+1)

TA1CCTL1|=OUTMOD1; //设置TA1比较器1的PWM输出为模式2： 计数到CCR1值翻转，到CCR0值置0

TA1CCR1=24575; //设置TA1比较器1设定值CCR1，TA1CCR1=TA1CCR0\*(1-PWM波形占空比25%)

TA1CCTL2|=OUTMOD1; //设置比较器2的PWM输出为模式2： 计数到CCR2值翻转，到CCR0值置0

TA1CCR2=16383; //设置TA1比较器2设定值CCR2，TA1CCR2=TA1CCR0\*(1-PWM波形占空比50%)

TA1CTL |=TACLR+MC0; //设置增计数方式，使计数器从0开始计数，计数到TA1CCR0后又从0计数。

while(1){ }; // 主循环，CPU可做其他事情

}

1. **蜂鸣器的发声控制**

在任务1的基础上，将定时器输出的PWM波连接到蜂鸣器的控制引脚上，编程控制蜂鸣器发出低、中、高三个音域1、2、3、4、5、6、7等不同音符的音调。可分别用无源和有源蜂鸣器完成，比较二者的不同之处。

**蜂鸣器的发声原理：**

蜂鸣器分有源和无源蜂鸣器两种。这里的源，指的是振荡源。

有源蜂鸣器，内部有振荡源，在它的两个引脚接上电压就会发出固定频率的声音；

无源蜂鸣器，也称电磁式蜂鸣器，内部没有振荡源，主要由永磁体、线圈、振荡片构成。需要由外部提供一定频率的方波信号作为振荡信号(源)，作用于内部线圈，使内部振荡片产生振动，发出声音。

两种蜂鸣器的发声可由输入的方波信号来改变，其中音调的高低由方波的频率决定，音长由方波的个数，即输出方波的时间长短决定。调节各音的音调和音长，可以使蜂鸣器发出不同旋律的音乐。

表6-1是钢琴中央C开始的一个8度的各音符频率，高8度和低8度各音的频率分别是表中对应唱名音的2倍（高音区）和1/2（低音区），更多音符对应的相应频率可参看图6-3。

有源蜂鸣器的发声除控制方波的频率外，还需控制方波的占空比为90%以上，减小其固有频率带来的杂声，音质效果才较好。

表6-1 C大调各音符参考频率

|  |  |  |  |
| --- | --- | --- | --- |
| 唱名 | 简谱 | C调(Hz) | C调(Hz)(取整) |
| do | 1 | 261.6 | 262 |
| re | 2 | 293.7 | 294 |
| mi | 3 | 329.6 | 330 |
| fa | 4 | 349.2 | 349 |
| sol | 5 | 392.0 | 392 |
| la | 6 | 440.0 | 440 |
| si | 7 | 493.9 | 494 |
| do(高) | i | 523.2 | 523 |

1. **(提高)调节小车运动速度**

将定时器TA输出的两路PWM波分别连接小车的控制端PWMA、PWMB，完成：

1. 控制小车以低速、中速、快速3个档位速度前行或后退；
2. 控制小车在前行的过程中左转、右转；
3. 控制小车在后退的过程中左转、右转；
4. **(提高)低功耗模式学习**

用跳线将P2.3与发光二极管L4短接，将P2.4用长杜邦线与蜂鸣器buzz短接，P1.1与按键K2短接，利用程序L4\_LPM.c，完成下面操作，了解低功耗模式的进入和退出。

1. 运行程序，观察现象，记录进入低功耗前、进入低功耗后、响应中断后、退出中断后的发光二极管和蜂鸣器状态，并做分析。
2. 如果中断程序中有LPM4\_EXIT 语句，运行的结果会有什么不同？请分析。

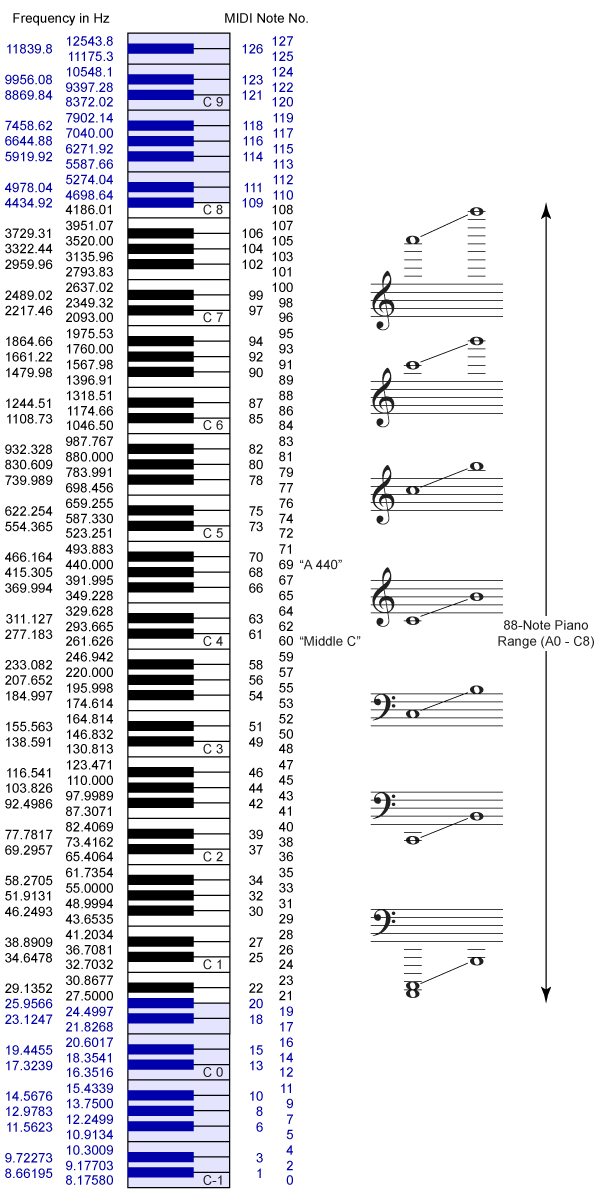


图6-3 钢琴各音频率表

任务4：L6\_LPM.c（提供电子版）：

#include "msp430.h"

void delay( unsigned int i) //延时函数

{ unsigned int j; //定义局部变量

for (j=0;j<i;j++);

}

void Blink( ) //LED闪烁

{ unsigned int i;

for (i=0;i<3;i++)

{ P2OUT &= ~BIT3;

delay(0xe000);

P2OUT |=BIT3;

delay(0xe000);

};

}

void Buzz( ) //蜂鸣响

{ unsigned int i;

for (i=0;i<2;i++)

{ P2OUT &= ~BIT4;

delay(0xf800);

P2OUT |=BIT4;

delay(0xf800);

};

}

int main ( void )

{

WDTCTL = WDTPW + WDTHOLD; //关闭看门狗

//设置端口P2.3输出，控制LED,P2.4输出，控制蜂鸣器

P2SEL &=~(BIT3+BIT4);

P2SEL2&=~(BIT3+BIT4);

P2OUT |= BIT3+BIT4;

P2DIR |= BIT3+BIT4;

//设置端口P1.1允许中断

P1SEL &=~BIT1;

P1SEL2 &=~BIT1;

P1REN |=BIT1;

P1OUT |=BIT1;

P1DIR &=~BIT1;

P1IES |=BIT1;

P1IFG &=~BIT1;

P1IE |=BIT1;

\_EINT();

//P1.0输出时钟ACLK, P1.4输出时钟SMCLK

P1SEL |=BIT0+BIT4;

P1SEL2 &=~(BIT0+BIT4);

P1DIR |=BIT0+BIT4;

Blink();

Buzz();

for (;;) //主循环

{ LPM4;

Blink();

}

}

#pragma vector=PORT1\_VECTOR //中断向量设置

\_\_interrupt void port\_ISR( ) //中断函数

{ Buzz();

P1IFG&=~(BIT1); //清中断标志

// LPM4\_EXIT;

}