专业：自动化
姓名：骆乃瑞
学号：3110000033
日期：2014-04-04
地点：教 2-125

浙江大学实验报告

课程名称：DSP 原理与应用　指导老师：杨家强　成绩：

实验名称：实验九 EV 定时器及 PWM 输出实验　实验类型：　同组学生姓名：

一、 实验目的和要求

1．了解 TMS320F2812 事件管理器模块的脉宽调制电路 PWM 的特性参数;

2．掌握 PWM 电路的控制方法;

3．掌握用程序控制产生不同占空比的 PWM 波形。

二、 实验内容和原理

1．脉宽调制电路 PWM 的特性

每个事件管理器模块可同时产生多达 8 路的 PWM 波形输出。由 3 个带可编程死区控制的比较单元产生独立的 3 对(即 6 路输出),以及由通用定时器比较产生的 2 路独立的 PWM 输出。

PWM 的特性如下:

- 16 位寄存器;
- 有从 0 到 16μs 的可编程死区发生器控制 PWM 输出对;
- 最小的死区宽度为 1 个 CPU 时钟周期;
- 对 PWM 频率的变动可根据需要改变 PWM 的载波频率;
- 在每个 PWM 周期内和以后可根据需要改变 PWM 脉冲的宽度;
- 外部可屏蔽的功率驱动保护中断;
- 脉冲形式发生器电路,用于可编程对称、非对称以及 4 个空间矢量 PWM 波形产生;

- 自动重装载的比较和周期存器使 CPU 的负荷最小。

2．PWM 电路的设置

在电机控制和运动控制的应用中,PWM电路被设计为减少产生PWM 波形的CPU 开销和减少用户的工作量。与比较单元相关的 PWM 电路其 PWM 波形的产生由以下寄存器控制:对于 EVA 模块,T1CON、COMCONA、ACTRA 和 DBTCONA;对于 EVB 模块,T3CON、COMCONB、ACTRB 和 DBTCONB。产生 PWM 的寄存器设置:

- 设置和装载 ACTRx 寄存器;
- 如果使能死区,则设置和装载 DBTCONx 寄存器;
- 设置和装载 T1PR 或 T3PR 寄存器,即规定 PWM 波形的周期;
- 初始化 CMPRx 寄存器;
- 设置和装载 COMCONx 寄存器;
- 设置和装载 T1CON 或 T3CON 寄存器来启动比较操作;
- 更新 CMPRx 寄存器的值,使输出的 PWM 波形的占空比发生变化。

三、 代码

对于思考题中的 4 路同频不同相非对称 PWM 输出，采用 4 个 PWM 模块自动置位清零，没有使用其中断.

由于实验时间不足，其完善实现在我自己的 C2000 Launchpad 上完成，由于新一代 Piccolo 处理器在结构上使用 ePWM(增强型)模块，因此代码实现有一些不同，PWM 调制由比较器送至 Action Qualifier 改变输出电平，以下是核心代码，完整的代码在 github 或者 http://dwz.cn/fXU2i 可看到:

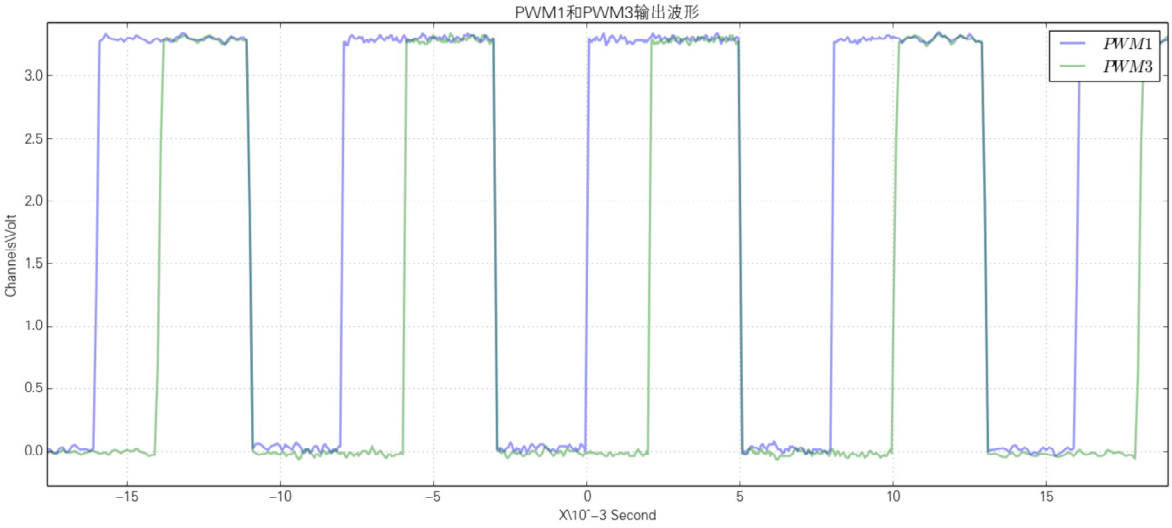
```
1  PWM_Handle myPwm1, myPwm2, myPwm3, myPwm4;
2
3 typedef struct {
4  int PERIOD, ST_CNT, PHASE;
5  int CMPA, CMPB;
6  PWM_Number_e NUM;
7 } pwm_info;
8
```

```
 9  const pwm_info pwmProfiles[]={
10   { 8000, 0000, 0000, 0000, 5000, PWM_Number_1 },
11   { 8000, 0000, 0000, 5000, 0000, PWM_Number_2 },
12   { 8000, 0000, 0000, 5000, 0000, PWM_Number_3 },
13   { 8000, 0000, 0000, 5000, 0000, PWM_Number_4 }
14  };
15
16  void InitEPWM(PWM_Handle pwm, pwm_info pi){
17
18   CLK_enablePwmClock(myClk, pi.NUM);
19   PWM_setHighSpeedClkDiv(pwm, PWM_HspClkDiv_by_10); // Clock ratio to SYSCLKOUT
20  #ifdef _DEBUG
21   PWM_setClkDiv(pwm, PWM_ClkDiv_by_128);
22  #else
23   PWM_setClkDiv(pwm, PWM_ClkDiv_by_1);
24  #endif
25   // Setup TBCLK
26   PWM_setCounterMode(pwm, PWM_CounterMode_Up); // Count up
27   PWM_setPeriod(pwm, pi.PERIOD); // Set timer period
28   PWM_setPhase(pwm, pi.PHASE);
29   PWM_setCount(pwm, pi.ST_CNT); // Set counter
30
31   // Set Compare values
32   PWM_setCmpA(pwm, pi.CMPA); // Set compare A value
33   PWM_setCmpB(pwm, pi.CMPB); // Set Compare B value
34
35   // Set actions
36   PWM_setActionQual_Zero_PwmA(pwm, PWM_ActionQual_Set); // Set PWM1A on Zero
37   PWM_setActionQual_CntUp_CmpA_PwmA(pwm, PWM_ActionQual_Clear); // Clear PWM1A on
event A, up count
38   PWM_setActionQual_Zero_PwmB(pwm, PWM_ActionQual_Set); // Set PWM1B on Zero
39   PWM_setActionQual_CntUp_CmpB_PwmB(pwm, PWM_ActionQual_Clear); // Clear PWM1B on
event B, up count
40
41   // Setup shadow register load on ZERO
42   PWM_setShadowMode_CmpA(pwm, PWM_ShadowMode_Shadow);
43   PWM_setShadowMode_CmpB(pwm, PWM_ShadowMode_Shadow);
44   PWM_setLoadMode_CmpA(pwm, PWM_LoadMode_Zero);
45   PWM_setLoadMode_CmpB(pwm, PWM_LoadMode_Zero);
46
47   // Interrupt where we will change the Compare Values
48   PWM_setIntMode(pwm, PWM_IntMode_CounterEqualZero); // Select INT on Zero event
49   PWM_enableInt(pwm); // Enable INT
50
51  }
52  void main(void)
53  {
54   // Initialize all the handles
55      myClk = CLK_init((void *)CLK_BASE_ADDR, sizeof(CLK_Obj));
56      myGpio = GPIO_init((void *)GPIO_BASE_ADDR, sizeof(GPIO_Obj));
57      myCpu = CPU_init((void *)NULL, sizeof(CPU_Obj));
58
59      initSysCtrl();
60      initGPIO();
61
62      // Initialize ePWMs
63      CLK_disableTbClockSync(myClk);
64
```

```
65  myPwm1 = PWM_init((void *)PWM_ePWM1_BASE_ADDR, sizeof(PWM_Obj));
66  myPwm2 = PWM_init((void *)PWM_ePWM2_BASE_ADDR, sizeof(PWM_Obj));
67  myPwm3 = PWM_init((void *)PWM_ePWM3_BASE_ADDR, sizeof(PWM_Obj));
68  myPwm4 = PWM_init((void *)PWM_ePWM4_BASE_ADDR, sizeof(PWM_Obj));
69
70      InitEPWM(myPwm1, pwmProfiles[0]);
71      InitEPWM(myPwm2, pwmProfiles[1]);
72      InitEPWM(myPwm3, pwmProfiles[2]);
73      InitEPWM(myPwm4, pwmProfiles[3]);
74      CLK_enableTbClockSync(myClk);
75
76      CPU_enableDebugInt(myCpu);
77
78      // Create phase shift
79      PWM_setCount(myPwm2, ((PWM_Obj *)myPwm1)->TBCTR - 2000);
80      PWM_setCount(myPwm3, ((PWM_Obj *)myPwm1)->TBCTR - 4000);
81      PWM_setCount(myPwm4, ((PWM_Obj *)myPwm1)->TBCTR - 6000);
82
83      for(;;) ;
84  }
```

四、 实验结果与分析(必填)

1．用示波器记录在不同占空比下的 PWM 输出波形。



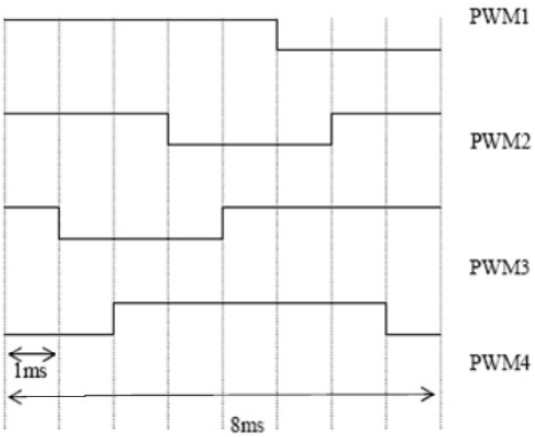
理论占空比为 62.5%与 36%，测量结果占空比为 62.35%与 35.61%

2．说明其载波频率、占空比与程序中的什么设置相关?

载波频率由 TxPR 决定周期时钟数，并由高速外围时钟 HSPCLK 决定时钟频率.

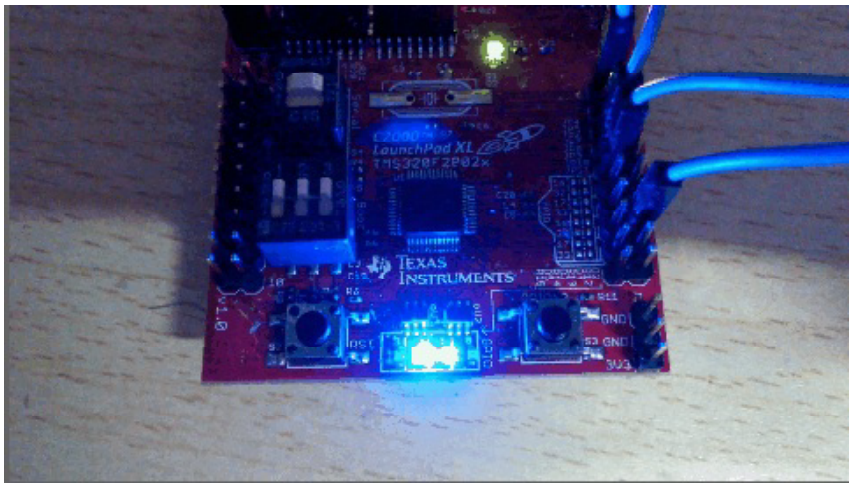
占空比由各 PWM 比较器 TxCPMR, CPMRx 决定.

3．4 路形如:

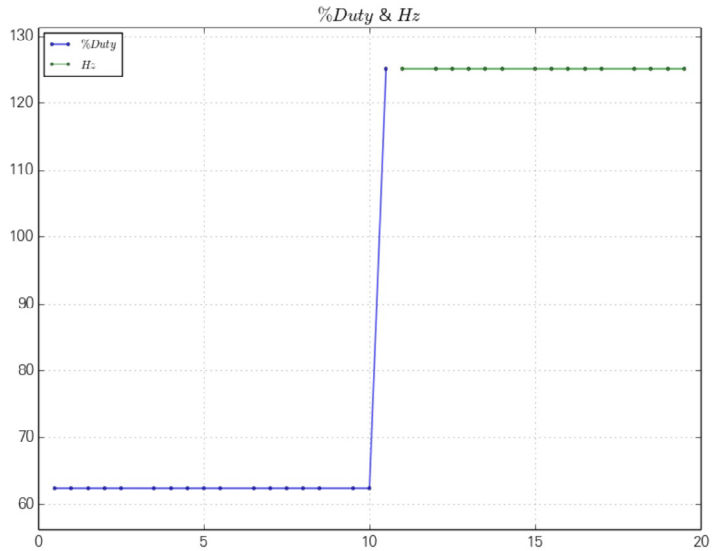


的 PWM 波的输出结果:

在调试模式下载波频率为 0.977Hz，可以观察到输出到 LED 的效果(动态 http://dwz.cn/fXUeU):



在发布模式下用万用表测得 PWM 输出为 125.2Hz，占空比 62.5%:



五、 讨论、心得

对比不同子系列的C2000 DSP芯片的PWM模块可以看到ePWM模块比EV更加强大与好用，各PWM模块之间可以相互作用，时基，比较器，动作器，死区、斩波、陷阱区生成器，中断与事件触发器的组合使得硬件上生成更多样使用的输出，外围间相互作用降低CPU的开销，开发上更加方便：