哈爾濱工業大學（深圳）
HARBIN INSTITUTE OF TECHNOLOGY

实验报告

课程名称：DSP 的原理与应用

实验名称：实验 4 PWM 输出实验

专业-班级：电气 1 班 学号：220330124 姓名：舒晟超

实验日期：2024 年 5 月 26 日

试验台号：

报告总分数：

教师评语：

助教签字：

教师签字：

日　　期：

一、实验目的

1. 了解 TMS320F28335 DSP 片内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. 学会用程序控制产生不同占空比的 PWM 波形。

二、实验过程

1. 导入工程并导入硬件仿真文件。

2. 打开程序并推测实验现象；

3. 下载并运行程序，观察实验现象。

三、实验结果

1. 导入工程并导入硬件仿真文件。

2. 阅读代码，编译并下载程序。

```
// For this example, only initialize the ePWM
void InitEPwm1Example()
{
    // 初始化 TB 模块
    EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP; // 设置计数模式为增计数
    EPwm1Regs.TBPRD = EPWM1_TIMER_TBPRD;       // 设置周期寄存器
    EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;    // 失能相位控制
    EPwm1Regs.TBPHS.half.TBPHS = 0x0000;       // 相位寄存器值清零
    EPwm1Regs.TBCTR = 0x0000;                  // 计数器清零
    EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV2;   // 设置 TBCLK 频率
    EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV2;      // 设置分频
    // 初始化 CC 模块
    EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW; // 使能 CMPA 影子寄存器
    EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW; // 使能 CMPB 影子寄存器
    EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO; // 影子寄存器在计数到 0 时装载
    EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO; // 影子寄存器在计数到 0 时装载
    EPwm1Regs.CMPA.half.CMPA = EPWM1_MIN_CMPA; // Set compare A value
    EPwm1Regs.CMPB = EPWM1_MIN_CMPB;           // Set Compare B value
    // 初始化 AQ 模块
    EPwm1Regs.AQCTLA.bit.ZRO = AQ_SET;   // TB 计数器为 0 时置为高电平
    EPwm1Regs.AQCTLA.bit.CAU = AQ_CLEAR; // TB 计数器到达 CMPA 值时置为低电平
    EPwm1Regs.AQCTLB.bit.ZRO = AQ_SET;   // TB 计数器为 0 时置为高电平
    EPwm1Regs.AQCTLB.bit.CBU = AQ_CLEAR; // TB 计数器到达 CMPB 值时置为低电平
    // 初始化 ET 模块
    EPwm1Regs.ETSEL.bit.INTSEL = ET_CTR_ZERO; // 时基计数器等于 0 时触发中断
    EPwm1Regs.ETSEL.bit.INTEN = 1;            // 中断线使能
    EPwm1Regs.ETPS.bit.INTPRD = ET_3RD;       // 每三个事件触发中断
    // 设置自定义 ePWM 寄存器
    // Start by increasing CMPA & CMPB
    epwm1_info.EPwm_CMPA_Direction = EPWM_CMP_UP;
    epwm1_info.EPwm_CMPB_Direction = EPWM_CMP_UP;
    epwm1_info.EPwmTimerIntCount = 0; // Zero the interrupt counter
```

```
    // Set the pointer to the ePWM module
    epwm1_info.EPwmRegHandle = &EPwm1Regs;
    // Setup min/max CMPA/CMPB values
    epwm1_info.EPwmMaxCMPA = EPWM1_MAX_CMPA;
    epwm1_info.EPwmMinCMPA = EPWM1_MIN_CMPA;
    epwm1_info.EPwmMaxCMPB = EPWM1_MAX_CMPB;
    epwm1_info.EPwmMinCMPB = EPWM1_MIN_CMPB;
}
```

（1）注意到 PWM1 和 PWM2 在程序中的配置如下：

HSPCLKDIV 为 2 分频；CLKDIV 为 2 分频；

通过公式：

$$f_{TBCLK} = \frac{SYSCLKOUT}{HSPCLKDIV \times CLKDIV}$$

$$T_{pwm} = (TBPRD + 1) \times T_{TBCLK}$$

由于为增计数模式，计算得到 PWM 频率为 18740Hz。实际测得 PWM 频率符合上述条件。

（2）由于中断程序中循环改变了 CMPA 和 CMPB 比较值，可以观察到 PWM 的占空比是增加（到 100%）-> 减少（到 0%）-> 增加循环变化的。

（3）对于 PWM1A 和 PWM1B，注意到两路 PWM 配置相同，故波形相同，如下图：

PWM2A 和 PWM1A 是反相的，如下图：

PWM2B 和 PWM2A 占空比相加得到 100%，如下图。

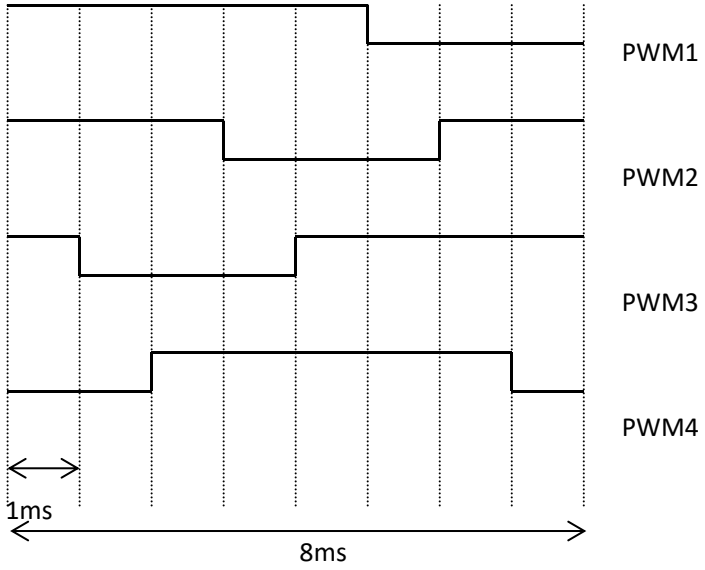
（3）PWM3 在程序中的配置如下：

HSPCLKDIV 为 1 分频；CLKDIV 为 1 分频；PWM 频率计算得到 74962Hz。

注意到 PWM3B 的动作是计数到 0 时反转，所以为 50%占空比的方波，频率为 37481Hz。

四、问题与思考

1、试设计四路（PWM1,PWM2,PWM3,PWM4）输出，载波频率为 8ms，波形关系如下（一个周期），要求写出设计思路、写出四路 PWM 的周期和占空比计算过程、画出四路 TB 计数器计数波形、画出四路 CC 模块的比较值波形、画出四路 AQ 模块的动作标识符号、提供程序代码、提供的示波器实验波形图标出要求的频率和相位关系。（4 分）



（1）设计思路，周期和占空比计算：

1. 考虑到如果使用增(减)计数模式得到的 TBPRD 值均大于 65535（考虑 HSPCLKDIV 和 CLKDIV 均为 4 分频时）

$$f_{TBCLK} = \frac{SYSCLKOUT}{HSPCLKDIV \times CLKDIV}$$

$$T_{pwm} = (TBPRD + 1) \times T_{TBCLK}$$

取 HSPCLKDIV 为 4 分频，CLKDIV 为 4 分频，由 Tpwm = 8ms，得到 TBPRD 值为 74999。大于 int 类型的限定值 65535。
使用增减计数模式：

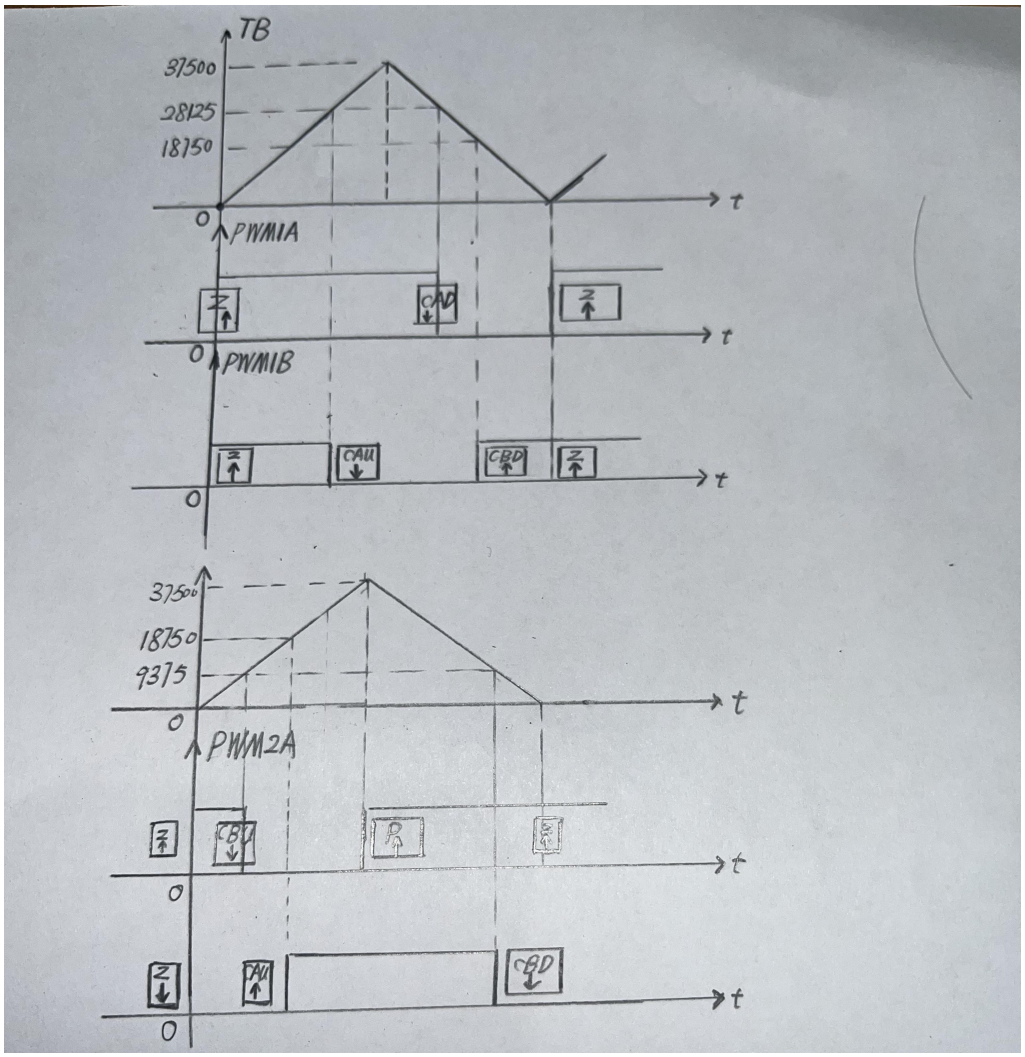
```
1.  EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN;
2.  EPwm2Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN;
```

由以下公式：

$$f_{TBCLK} = \frac{SYSCLKOUT}{HSPCLKDIV \times CLKDIV}$$

$$T_{pwm} = 2 \times TBPRD \times T_{TBCLK}$$

取 HSPCLKDIV 为 4 分频，CLKDIV 为 4 分频，由 Tpwm = 8ms，得到 TBPRD 值为 37500。
接下来设计四路波形，注意到为增减计数模式，每一个比较值有上升到达和下降到达两种形式，波形设置如下：



PWM1： 占空比为 1-28125/75000 = 62.5%;

PWM2：占空比为(28125+18750)/75000 = 62.5%;
PWM3：占空比为(9375+37500)/75000 = 62.5%;
PWM3：占空比为 1- (18750+9375)/75000 = 62.5%;

实验代码如下：

```
// TI File $Revision: /main/8 $
// Checkin $Date: August 10, 2007   09:04:53 $
//###########################################################################
//
// FILE:    Example_2833xEPwm3UpAQ.c
//
// TITLE:   Action Qualifier Module Upcount mode.
//
// ASSUMPTIONS:
//
//     This program requires the DSP2833x header files.
//
//     Monitor the ePWM1 - ePWM3 pins on a oscilloscope as
//     described below.
//
//        EPWM1A is on GPIO0
//        EPWM1B is on GPIO1
//
//        EPWM2A is on GPIO2
//        EPWM2B is on GPIO3
//
//        EPWM3A is on GPIO4
//        EPWM3B is on GPIO5
//
//     As supplied, this project is configured for "boot to SARAM"
//     operation.  The 2833x Boot Mode table is shown below.
//     For information on configuring the boot mode of an eZdsp,
//     please refer to the documentation included with the eZdsp,
//
//       $Boot_Table:
//
//         GPIO87   GPIO86     GPIO85   GPIO84
//          XA15     XA14       XA13     XA12
//           PU       PU         PU       PU
//        ========================================
//            1        1          1        1    Jump to Flash
//            1        1          1        0    SCI-A boot
//            1        1          0        1    SPI-A boot
//            1        1          0        0    I2C-A boot
//            1        0          1        1    eCAN-A boot
//            1        0          1        0    McBSP-A boot
//            1        0          0        1    Jump to XINTF x16
//            1        0          0        0    Jump to XINTF x32
//            0        1          1        1    Jump to OTP
```

```
45. //            0       1         1       0    Parallel GPIO I/O boot
46. //            0       1         0       1    Parallel XINTF boot
47. //            0       1         0       0    Jump to SARAM     <- "boot to SARA
    M"
48. //            0       0         1       1    Branch to check boot mode
49. //            0       0         1       0    Boot to flash, bypass ADC cal
50. //            0       0         0       1    Boot to SARAM, bypass ADC cal
51. //            0       0         0       0    Boot to SCI-A, bypass ADC cal
52. //                                           Boot_Table_End$
53. //
54. // DESCRIPTION:
55. //
56. //    This example configures ePWM1, ePWM2, ePWM3 to produce an
57. //    waveform with independant modulation on EPWMxA and
58. //    EPWMxB.
59. //
60. //    The compare values CMPA and CMPB are modified within the ePWM's ISR
61. //
62. //    The TB counter is in upmode for this example.
63. //
64. //    View the EPWM1A/B, EPWM2A/B and EPWM3A/B waveforms
65. //    via an oscilloscope
66. //
67. //
68. //###########################################################################
69. // $TI Release: DSP2833x Header Files V1.01 $
70. // $Release Date: September 26, 2007 $
71. //###########################################################################
72.
73.
74. #include "DSP2833x_Device.h"     // DSP2833x Headerfile Include File
75. #include "DSP2833x_Examples.h"   // DSP2833x Examples Include File
76.
77. // Prototype statements for functions found within this file.
78. void InitEPwm1Example(void);
79. void InitEPwm2Example(void);
80. interrupt void epwm1_isr(void);
81. interrupt void epwm2_isr(void);
82.
83. // Configure the period for each timer
84. #define EPWM1_TIMER_TBPRD  37500  // Period register
85.
86. #define EPWM2_TIMER_TBPRD  37500  // Period register
87.
88.
89. void main(void)
90. {
```

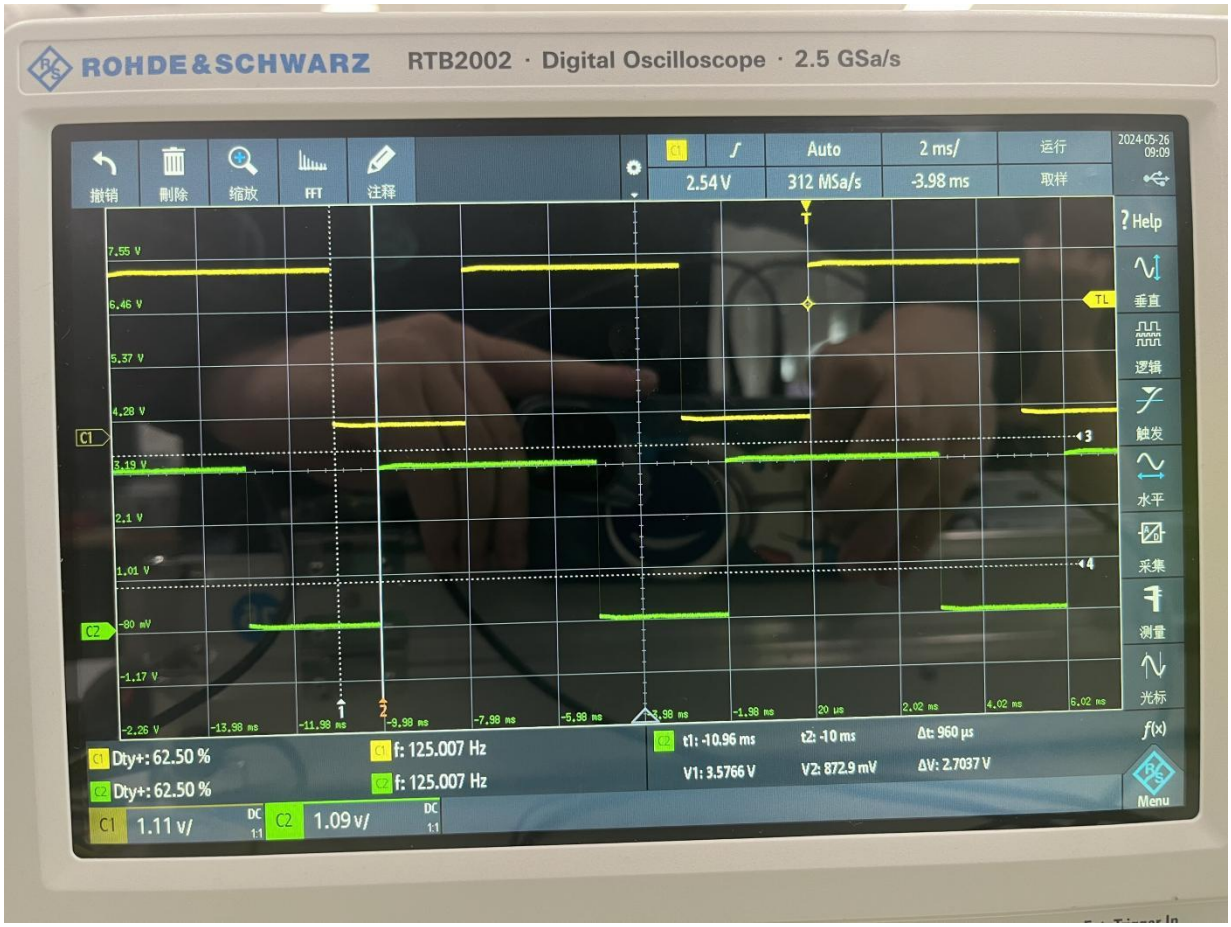
```
// Step 1. Initialize System Control:
// PLL, WatchDog, enable Peripheral Clocks
// This example function is found in the DSP2833x_SysCtrl.c file.
   InitSysCtrl();

// Step 2. Initalize GPIO:
// This example function is found in the DSP2833x_Gpio.c file and
// illustrates how to set the GPIO to it's default state.
// InitGpio();  // Skipped for this example

// For this case just init GPIO pins for ePWM1, ePWM2, ePWM3
// These functions are in the DSP2833x_EPwm.c file
   InitEPwm1Gpio();
   InitEPwm2Gpio();


// Step 3. Clear all interrupts and initialize PIE vector table:
// Disable CPU interrupts
   DINT;

// Initialize the PIE control registers to their default state.
// The default state is all PIE interrupts disabled and flags
// are cleared.
// This function is found in the DSP2833x_PieCtrl.c file.
   InitPieCtrl();

// Disable CPU interrupts and clear all CPU interrupt flags:
   IER = 0x0000;
   IFR = 0x0000;

// Initialize the PIE vector table with pointers to the shell Interrupt
// Service Routines (ISR).
// This will populate the entire table, even if the interrupt
// is not used in this example.  This is useful for debug purposes.
// The shell ISR routines are found in DSP2833x_DefaultIsr.c.
// This function is found in DSP2833x_PieVect.c.
   InitPieVectTable();

// Interrupts that are used in this example are re-mapped to
// ISR functions found within this file.
   EALLOW;  // This is needed to write to EALLOW protected registers
   PieVectTable.EPWM1_INT = &epwm1_isr;
   PieVectTable.EPWM2_INT = &epwm2_isr;
   EDIS;    // This is needed to disable write to EALLOW protected registers

// Step 4. Initialize all the Device Peripherals:
// This function is found in DSP2833x_InitPeripherals.c
// InitPeripherals();  // Not required for this example
```

```
139.
140. // For this example, only initialize the ePWM
141.
142.    EALLOW;
143.    SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 0;
144.    EDIS;
145.
146.    InitEPwm1Example();
147.    InitEPwm2Example();
148.
149.    EALLOW;
150.    SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 1;
151.    EDIS;
152.
153.
154. // Step 5. User specific code, enable interrupts:
155.
156. // Enable CPU INT3 which is connected to EPWM1-3 INT:
157.    IER |= M_INT3;
158.
159. // Enable EPWM INTn in the PIE: Group 3 interrupt 1-3
160.    PieCtrlRegs.PIEIER3.bit.INTx1 = 1;
161.    PieCtrlRegs.PIEIER3.bit.INTx2 = 1;
162.
163. // Enable global Interrupts and higher priority real-time debug events:
164.    EINT;   // Enable Global interrupt INTM
165.    ERTM;   // Enable Global realtime interrupt DBGM
166.
167. // Step 6. IDLE loop. Just sit and loop forever (optional):
168.    for(;;)
169.    {
170.        asm("          NOP");
171.    }
172.
173. }
174.
175. interrupt void epwm1_isr(void)
176. {
177.    // Clear INT flag for this timer
178.    EPwm1Regs.ETCLR.bit.INT = 1;
179.
180.    // Acknowledge this interrupt to receive more interrupts from group 3
181.    PieCtrlRegs.PIEACK.all = PIEACK_GROUP3;
182. }
183.
184.
185. interrupt void epwm2_isr(void)
```

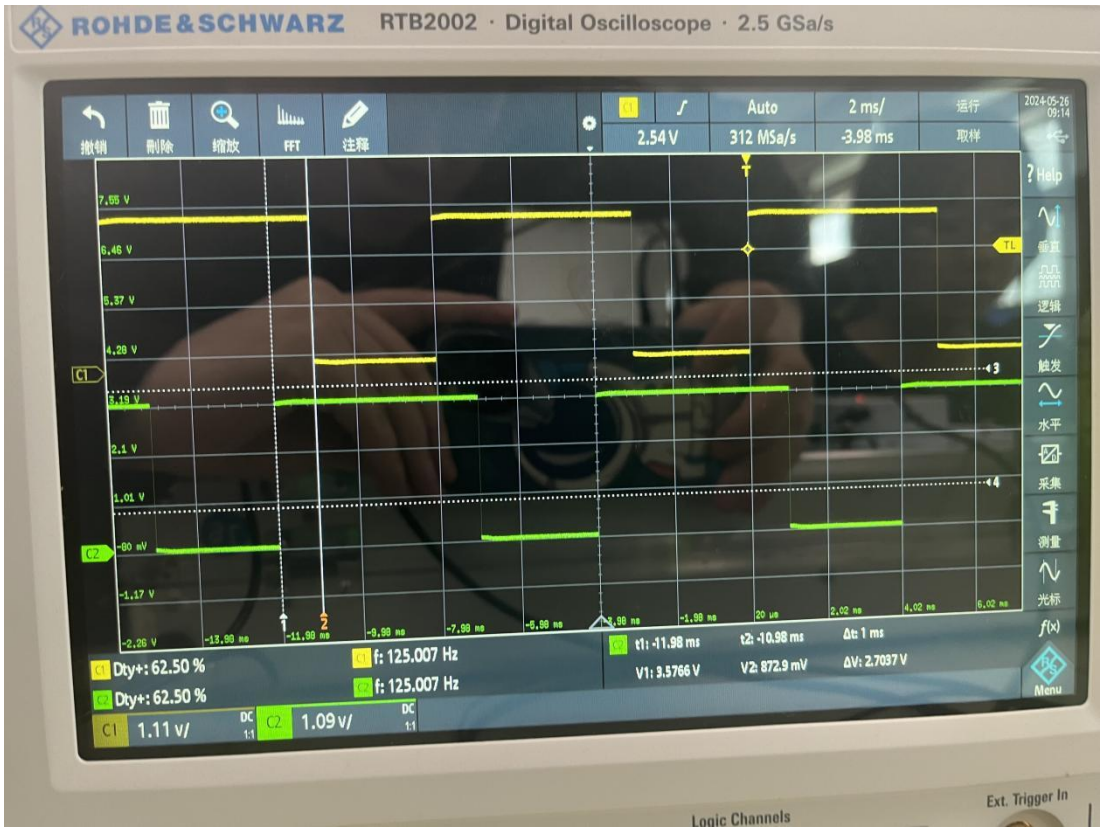
```
{
    // Clear INT flag for this timer
    EPwm2Regs.ETCLR.bit.INT = 1;

    // Acknowledge this interrupt to receive more interrupts from group 3
    PieCtrlRegs.PIEACK.all = PIEACK_GROUP3;
}

void InitEPwm1Example()
{

    // Setup TBCLK
    EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN; // Count up
    EPwm1Regs.TBPRD = EPWM1_TIMER_TBPRD;       // Set timer period
    EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;    // Disable phase loading
    EPwm1Regs.TBPHS.half.TBPHS = 0x0000;       // Phase is 0
    EPwm1Regs.TBCTR = 0x0000;                  // Clear counter
    EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV4;   // Clock ratio to SYSCLKOUT
    EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV4;

    // Setup shadow register load on ZERO
    EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
    EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
    EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;
    EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;

    // Set Compare values
    EPwm1Regs.CMPA.half.CMPA = 28125;    // Set compare A value
    EPwm1Regs.CMPB = 18750;              // Set Compare B value

    // Set actions
    EPwm1Regs.AQCTLA.bit.ZRO = AQ_SET;            // Set PWM1A on Zero
    EPwm1Regs.AQCTLA.bit.CAD = AQ_CLEAR;

    EPwm1Regs.AQCTLB.bit.ZRO = AQ_SET;            // Set PWM1B on Zero
    EPwm1Regs.AQCTLB.bit.CAU = AQ_CLEAR;
    EPwm1Regs.AQCTLB.bit.CBD = AQ_SET;

    EPwm1Regs.ETSEL.bit.INTSEL = ET_CTR_ZERO;     // Select INT on Zero event
    EPwm1Regs.ETSEL.bit.INTEN = 1;                // Enable INT
    EPwm1Regs.ETPS.bit.INTPRD = ET_3RD;           // Generate INT on 3rd event
}


void InitEPwm2Example()
```

```
{
    // Setup TBCLK
    EPwm2Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN; // Count up
    EPwm2Regs.TBPRD = EPWM2_TIMER_TBPRD;       // Set timer period
    EPwm2Regs.TBCTL.bit.PHSEN = TB_DISABLE;    // Disable phase loading
    EPwm2Regs.TBPHS.half.TBPHS = 0x0000;       // Phase is 0
    EPwm2Regs.TBCTR = 0x0000;                  // Clear counter
    EPwm2Regs.TBCTL.bit.HSPCLKDIV = TB_DIV4;   // Clock ratio to SYSCLKOUT
    EPwm2Regs.TBCTL.bit.CLKDIV = TB_DIV4;

    // Setup shadow register load on ZERO
    EPwm2Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
    EPwm2Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
    EPwm2Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;
    EPwm2Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;

    // Set Compare values
    EPwm2Regs.CMPA.half.CMPA = 18750;       // Set compare A value
    EPwm2Regs.CMPB = 9375;                  // Set Compare B value

    // Set actions
    EPwm2Regs.AQCTLA.bit.ZRO = AQ_SET;
    EPwm2Regs.AQCTLA.bit.CBU = AQ_CLEAR;
    EPwm2Regs.AQCTLA.bit.PRD = AQ_SET;


    EPwm2Regs.AQCTLB.bit.ZRO = AQ_CLEAR;
    EPwm2Regs.AQCTLB.bit.CAU = AQ_SET;
    EPwm2Regs.AQCTLB.bit.CBD = AQ_CLEAR;


    // Interrupt where we will change the Compare Values
    EPwm2Regs.ETSEL.bit.INTSEL = ET_CTR_ZERO;        // Select INT on Zero event
    EPwm2Regs.ETSEL.bit.INTEN = 1;                   // Enable INT
    EPwm2Regs.ETPS.bit.INTPRD = ET_3RD;              // Generate INT on 3rd event
}

//===========================================================================
// No more.
//===========================================================================
```

示波器波形如下，上侧皆为 PWM1



PWM1 滞后 PWM2 相位 90°；



PWM1 滞后 PWM3 相位 180°；

PWM1 超前 PWM4 相位 90°。