PAN1070 Timer Sample Application Note

PAN-CLT-VER-B0, Rev 0.1

PanchipMicroelectronics

www.panchip.com

修订历史

版本	修订日期	描述
V0.1	2023-10-10	初始版本创建

目录

第1章 例程演示内容

1.1 测试内容

1. 寄存器默认值（Register default value）
2. 基本计数模式（Basic Counting Modes）
   a) 单次计数模式（One-Shot Mode）
   b) 周期计数模式（Periodic Mode）
   c) 切换输出模式（Toggle-Output Mode）
   d) 连续计数模式（Continuous-Counting Mode）
3. 事件计数模式（Event Counting Operation Mode）
4. 外部输入捕获模式（Input Capture Modes）
   a) 自由计数捕获模式（Free-Counting Capture Mode）
   b) 外部复位计数捕获模式（External Reset Counter Mode）
   c) 触发计数捕获模式（Trigger-Counting Capture Mode）
   d) 内部 RCL 时钟捕获功能（Internal RCL Capture Function）
5. 睡眠模式唤醒系统（Wakeup Function）

1.2 环境准备

1.2.1 软件环境

1.2.1.1 演示代码

测试工程文件：

<PAN1070-DK>\03_MCU\mcu_samples\TIMER\keil\TIMER.uvprojx

测试源文件目录：

<PAN1070-DK>\03_MCU\mcu_samples\TIMER\src

1.2.1.2 软件工具

1、SecureCRT（用于显示 PC 与 Test Board 的交互过程，打印 log 等）

2、KingstVIS（逻辑分析仪 LA1010 配套软件）

1.2.2 硬件环境

1、PAN1070 COB 1 块

a) UART0（测试交互接口，TX：P16，RX：P17，波特率：921600）

b) TIMER0/1/2（待测模块，软件可配，默认 TIMER0，Toggle-Output Pin: P16，External-Input Pin: P12）

c) PWM1 Channel 6/7（辅助测试 PWM，用来产生特定频率和占空比的方波，输出给待测 Timer，默认使用 Channel 6，PWM-Output Pin: P30）

d) SWD（用来调试和烧录程序，SWDCLK: P46，SWDIO: P47）

2、逻辑分析仪（波形抓取工具）

3、JLink（SWD 调试与烧录工具）

第2章 例程演示流程

2.1 环境配置

2.1.1 程序编译烧录

打开演示工程，确保可以编译通过。编译之前，可以在 timer_common.h 中配置待测 Timer 为 TIMER0、TIMER1 或 TIMER2。本测试工程默认将 TIMER0 作为待测 Timer（Target Timer），将 TIMER1 作为辅助测试 Timer（Auxiliary Timer），另外将 PWM1 的 Channel 6 用作辅助测试的 PWM Source。

2.1.2 硬件接线

接线方面，需要：

1. 将 UART0 与 PC 串口调试助手通过 USB 转串口芯片连接；

2. 将 TIMER0（待测模块）的 Input Pin（P10）与 PWM1 Channel 6（辅助测试模块）的 Output Pin（P30）相连。

2. 将 TIMER0（待测模块）的 Output Pin（P16）与逻辑分析仪相连。

2.2 Timer 工作流程

参考 User Manual 文档。

2.3 演示例程初始化

硬件连线完成并烧录测试程序后，EVB 上电，观察串口是否正常打印演示例程主菜单。

```
[14:55:23.345]收←◆CPU @ 48000000Hz

+-------------------------------------------------------------------------+
|                    PN107 Timer Sample Code.                             |
+-------------------------------------------------------------------------+
|   Press key to start specific testcase:                                 |
|                                                                         |
|   Input '0'    Testcase 0: Register Default Value Check.                |
|   Input '1'    Testcase 1: Timer Counting Operation Modes.              |
|   Input '2'    Testcase 2: Event Counting Mode.                         |
|   Input '3'    Testcase 3: Input Capture Modes.                         |
|   Input '4'    Testcase 4: Wake up from low power mode.                 |
|   Input '5'    Testcase 5: Timer counter bit check.                     |
+-------------------------------------------------------------------------+
```

2.4 演示步骤

2.4.1 Timer 所有寄存器默认状态

在主菜单下，输入‘0’命令 打印所有寄存器默认值：

测试目的：

检查所有 Timer 相关寄存器复位 Default 值状态。

测试预期：

寄存器默认值应和 Datasheet 上 Timer 模块默认值一致。

测试现象：

```
0
Target Timer0 Register Default Values:
---------------------------
CTL             = 0x00000005
CMP             = 0x00000000
INTSTS          = 0x00000000
CNT             = 0x00000000
CAP             = 0x00000000
EXTCTL          = 0x00000000
EINTSTS         = 0x00000000
TIMER Test OK, Success case: 0
```

测试分析：

参考芯片手册对比寄存器信息，发现是完全一致的，符合预期。

2.4.2 基本计数模式

在主菜单下，输入‘1’命令 进入 Subcase 菜单：

```
+-----------------------------------------------------------------+
|    Press key to test specific function:                         |
|                                                                 |
|    Input 'A'    One-Shot Mode.                                  |
|    Input 'B'    Periodic Mode.                                  |
|    Input 'C'    Toggle-Output Mode.                             |
|    Input 'D'    Continuous-Counting Mode.                       |
|    Press ESC key to back to the top level case list.            |
+-----------------------------------------------------------------+
```

2.4.2.1 单次计数模式

测试目的：

验证单次计数模式（One-Shot Mode）工作是否正常。

测试预期：

Timer 达到设定的 Timeout 后，能够触发（且只触发一次）中断并打印对应 Log。

测试现象：

输入‘A’命令，首先打印待测 Timer 的配置参数，然后开始计数。短暂等待（1000ms）后，打印计数中断触发的 Log：“CNT1”。

```
a
Timer0, Compare Value:16000000
Expected Cnt Freq:16000000, Real Cnt Freq:16000000
Expected Timeout:1000ms, Real Timeout:1000ms
Start Timer...
CNT1

+---------------------------------------------------------------------+
|    Press key to test specific function:                             |
|                                                                     |
|    Input 'A'    One-Shot Mode.                                      |
|    Input 'B'    Periodic Mode.                                      |
|    Input 'C'    Toggle-Output Mode.                                 |
|    Input 'D'    Continuous-Counting Mode.                           |
|    Press ESC key to back to the top level case list.                |
+---------------------------------------------------------------------+
```

测试分析：

由 Log 可知，待测 Timer 的 CMP Value 设置为 16M，预期计数频率为 16MHz（预期 Timeout 为 16M/16MHz = 1s），实际计数频率为 16MHz，与预期一致。

可见 Timer 应该在开始计数的 1s 后触发计数中断并打印相关 Log。实际观察到的现象也是与预期一致的。

注意：由于预期计数频率不一定能被 APB Clock 整除，所以实际计数频率可能会与预期不一致；此 testcase 实际与预期一致，是因为 APB 频率为 32MHz，正好是预期频率 16MHz 的整数倍。

2.4.2.2 周期计数模式

测试目的：

验证周期计数模式（Periodic Mode）工作是否正常。

测试预期：

Timer 达到设定的 Timeout 后，能够触发中断，打印对应 Log，然后重新计数，达到 Timeout 后继续触发中断，如此往复。

测试现象：

输入‘B’命令，首先打印待测 Timer 的配置参数，然后开始计数。并每隔一小段时间（1000ms）会打印计数中断触发的 Log：“CNTx”（x=1,2,3,4,5），5 次之后停止打印。

```
b
Timer0, Compare Value:16000000
Expected Cnt Freq:16000000, Real Cnt Freq:16000000
Expected Timeout:1000ms, Real Timeout:1000ms
Start Timer...
CNT1
CNT2
CNT3
CNT4
CNT5

+-----------------------------------------------------------------+
|    Press key to test specific function:                         |
|                                                                 |
|    Input 'A'    One-Shot Mode.                                  |
|    Input 'B'    Periodic Mode.                                  |
|    Input 'C'    Toggle-Output Mode.                             |
|    Input 'D'    Continuous-Counting Mode.                       |
|    Press ESC key to back to the top level case list.            |
+-----------------------------------------------------------------+
```

测试分析：

Timeout 设定与单次计数模式的 Testcase 是完全一致的，因此 Timer 应该在开始计数后，每隔 1000ms 触发一次计数中断并打印相关 Log，如此循环。不过测试程序在触发 5 次中断后，会强制将中断关闭并复位 Timer，因此 Log 只会打印 5 次中断 Log。实际观察到的现象也是与预期一致的。

2.4.2.3 切换输出模式

测试目的：

验证切换输出模式（Toggle-Output Mode）工作是否正常。

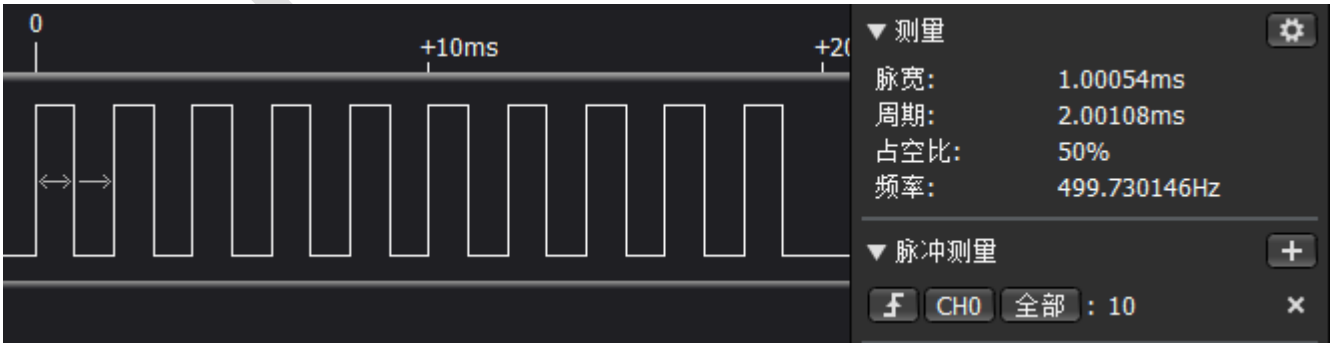
测试预期：

Timer 达到设定的 Timeout 后，变换 Output Pin 的电平，并触发中断，打印对应 Log，然后重新计数，达到 Timeout 后继续触发中断，如此往复。

测试现象：

连接好逻辑分析仪与待测 Timer 的 Output Pin，并输入‘C’命令，首先打印待测 Timer 的配置参数，然后开始计数。每隔一小段时间（1000us）会打印计数中断触发的 Log：“CNTx”（x=1~20），20 次之后停止打印。

抓取 Output Pin 的波形如下：



测试 Log 如下：

```
c
Timer0, Compare Value:16000
Expected Cnt Freq:16000000, Real Cnt Freq:16000000
Expected Timeout:1000us, Real Timeout:1000us
Output Waveform Frequency, Expected:500Hz, Real:500Hz
Start Timer...
CNT1
CNT2
CNT3
CNT4
CNT5
CNT6
CNT7
CNT8
CNT9
CNT10
CNT11
CNT12
CNT13
CNT14
CNT15
CNT16
CNT17
CNT18
CNT19
CNT20
```

测试分析：

由 Log 可知，待测 Timer 的 CMP Value 设置为 16K，预期计数频率为 16MHz（预期 Timeout 为 16K/16MHz = 1ms），实际计数频率为 16MHz。

因此 Timer 应该在开始计数后，每隔 1ms（16K/16MHz）触发一次计数中断并打印相关 Log，如此循环。另外由于测试程序在触发 20 次中断后，会强制将中断关闭并复位 Timer，因此 Log 只会打印 20 次中断 Log。

再看波形图，从图中也可看出，Output Pin 输出脉宽 1000us，占空比 50%的方波，频率为 500Hz，与预期一致。

2.4.2.4 连续计数模式

测试目的：

验证连续计数模式（Continuous-Counting Mode）工作是否正常。

测试预期：

Timer 达到设定的 Timeout 后，能够触发中断，打印对应 Log，然后继续计数，这时候重新配置 CMP Value 为更大的值，则稍后 Counter 值与新设定的 CMP Value 值相等时，再次触发中断，以此类推。

测试现象：

输入‘D’命令，Log 首先打印 Counter Freq 为 300KHz，然后 Start Timer 开始计数，大概 1s 左右打印第一条中断 log CNT1 和此时对应的 CMP Value 和 CNT Value；再过 1s 左右打印第二条中断 log CNT2 和此时对应的 CMP Value 和 CNT Value；再过 2s 左右打印第三条中断 log CNT3

和此时对应的 CMP Value 和 CNT Value。

```
d
Timer0, Cnt Freq:300469Hz
Start Timer...
CNT1
Timer Compare Value:300000, Counter Value:300024
CNT2
Timer Compare Value:600000, Counter Value:600016
CNT3
Timer Compare Value:1200000, Counter Value:1200017
Timer0 stopped.

+---------------------------------------------------------------------+
|    Press key to test specific function:                             |
|                                                                     |
|    Input 'A'    One-Shot Mode.                                      |
|    Input 'B'    Periodic Mode.                                      |
|    Input 'C'    Toggle-Output Mode.                                 |
|    Input 'D'    Continuous-Counting Mode.                           |
|    Press ESC key to back to the top level case list.                |
+---------------------------------------------------------------------+
```

测试分析：

Timer 计数时钟频率设置为 300KHz，第一次 CMP Value 设定为 300000，因此 Timer 应该在开始计数 1s 后左右，第一次触发计数中断并打印相关 Log，然后立刻读取此时的 CNT Value 值，得到 300024，基本与预期的 300000 一致，小的误差是预期的，因为程序从中断 Handler 中返回再获取 CNT Value 也耗费了一些时间。

由于设置成连续计数模式，因此 Counter Value 在 300000 之后继续计数，这时候把 CMP Value 增至 600000，因此 Time 将会在修改 CMP Value 的 1s 后，第二次触发计数中断并打印相关 Log，这时候读取 CNT Value，得到 600016，也是符合预期的。

同理，继续将 CMP Value 增至 1200000，从而使 Timer 在本次修改 CMP Value 的 2s 后，第三次触发计数中断并打印相关 Log，这时候读取 CNT Value，得到 1200017，同样符合预期。

2.4.3 事件计数模式

在主菜单下，输入 ‘2’ 命令 进入 Subcase 菜单：

```
+---------------------------------------------------------------------+
|    Press key to test specific function:                             |
|                                                                     |
|    Input 'A'    Event Counting Mode.                                |
|    Press ESC key to back to the top level case list.                |
+---------------------------------------------------------------------+
```

测试目的：

验证事件计数模式（Event Counting Operation Mode）工作是否正常。

测试预期：

Timer 在不同频率、占空比、持续时间的输入方波下，均能准确对输入事件（电平变化）进行计数。

测试现象：

输入‘A’命令，Log 先打印 CMP Value，接着使用 PWM 依次产生 3 段不同的方波，并在每段方波结束的时候打印预期计数值与实际计数值。

```
a
Timer start counting, Compare Value = 6100...

PWM start generating the 1st waveform (800Hz 50% 100ms)...
The 1st waveform counting done, expect count:80, actual count:80

PWM start generating the 2nd waveform (2000Hz 80% 3000ms)...
The 2nd waveform counting done, expect count:6080, actual count:6081

PWM start generating the 3rd waveform (20Hz 15% 2500ms)...
CNT1
The 3rd waveform counting done, expect count:6130, actual count:31


+-----------------------------------------------------------------+
|    Press key to test specific function:                         |
|                                                                 |
|    Input 'A'    Event Counting Mode.                            |
|    Press ESC key to back to the top level case list.            |
+-----------------------------------------------------------------+
```

测试分析：

Timer 设定：计数模式为周期计数，Compare Value 为 6100，监测的外部 Event 为波形下降沿。

使用 PWM 依次产生 3 段不同的方波。

第一段方波，频率 800Hz，占空比 50%，持续 100ms，由此可计算此段方波的下降沿个数为 800*0.1=80 个，从 Log 可以看到 CNT Value 的确是 80，符合预期。

第二段方波，频率 2KHz，占空比 80%，持续 3s，由此可计算此段方波下降沿个数为 2000*3=6000 个，因此第二段方波将会是 CNT Value 在第一段方波的基础上增加 6000，即 6080，从 Log 可以看到 CNT Value 的确变为 6081，符合预期。

第三段方波，频率为 20Hz，占空比 15%，持续 2.5s，由此计算此段方波下降沿个数为 20*2.5=50 个，因此第三段方波将会是 CNT Value 在第二段放标的基础上再增加 50，即 6130。但是从 Log 来看，CNT Value 的值是 30，这是正常的，原因是本测试 case 下，Timer 的计数模式被配置成周期计数，CMP Value 设定成 6100，因此超过 6100 后 CNT Value 将会从 0 开始重新计数。所以真正的 CNT Value 应该是 6130-6100=30，符合预期。

2.4.4 外部输入捕获模式

在主菜单下，输入‘3’命令 进入 Subcase 菜单：

```
+-----------------------------------------------------------------+
|    Press key to test specific function:                         |
|                                                                 |
|    Input 'A'    Free-Counting Capture Mode.                     |
|    Input 'B'    External Reset Counter Mode.                    |
|    Input 'C'    Trigger-Counting Capture Mode.                  |
|    Press ESC key to back to the top level case list.            |
+-----------------------------------------------------------------+
```

2.4.4.1 自由计数捕获模式

测试目的：

验证自由计数捕获模式（Free-Counting Capture Mode）工作是否正常。

测试预期：

能及时捕获外部事件（电平变化）并将 Counter 值载入 Capture Data Register。

测试现象：

输入‘A’命令，首先打印 Timer 设定基本信息，然后 PWM 产生一段方波，capture 中断被连续触发多次，同时打印当前 CNT Value 和 CAP Value 值。

```
a
Timer0 start, Cnt Freq:1000000Hz, Compare Value:400000
Start external waveform signal (freq:20Hz, duty cycle:80%, duration:500ms)...
CAP01 CntVal:  5842, CapVal:  5835
CAP02 CntVal: 45836, CapVal: 45835
CAP03 CntVal: 55836, CapVal: 55835
CAP04 CntVal: 95835, CapVal: 95834
CAP05 CntVal:105836, CapVal:105835
CAP06 CntVal:145835, CapVal:145834
CAP07 CntVal:155836, CapVal:155835
CAP08 CntVal:195835, CapVal:195834
CAP09 CntVal:205836, CapVal:205835
CAP10 CntVal:245835, CapVal:245834
CAP11 CntVal:255836, CapVal:255835
CAP12 CntVal:295835, CapVal:295834
CAP13 CntVal:305836, CapVal:305835
CAP14 CntVal:345835, CapVal:345834
CAP15 CntVal:355836, CapVal:355835
CAP16 CntVal:395835, CapVal:395834
CAP17 CntVal:      0, CapVal:      0
CAP18 CntVal:      0, CapVal:      0
CAP19 CntVal:      0, CapVal:      0
CAP20 CntVal:      0, CapVal:      0
```

测试分析：

Timer 设定为计数频率 1MHz，CMP Value 为 400K，计数模式为单次计数，捕获方式为双边沿（Both Edge）捕获。

外部 PWM 信号为频率 20Hz，占空比 80%，持续 500ms，因此有 20*0.5=10 个上升沿和 10 个下降沿，由占空比 80%可知相邻的高电平和低电平持续时间比值为 4:1。

由中断 Log 可以看出，PWM 持续期间共触发了 20 次中断，CntVal 与 CapVal 基本一致，微小误差是预期的，这是因为打印 Log 的时候，Counter 值已经与 Capture 的时候不同了。再纵向比较 CapVal 的值，相邻的 CapVal 相减：

CapVal02-CapVal01 = 45835-5835 = 40000

CapVal03-CapVal02 = 55835-45835 = 10000

CapVal04-CapVal03 = 95834-55835 = 39999

CapVal05-CapVal04 = 105835-95834 = 10001

CapVal06-CapVal05 = 145835-105835 = 40000

...

由此可见，高电平与低电平持续时间比值也是 4:1，且可以推算出外部信号频率为：1MHz /（39999+10001）= 20Hz，与 PWM 设定的信号一致。

另外，从 Log 可以看到，第 16 个 INT 之后，CntVal 和 CapVal 均为 0，这个原因是 Timer 计数模式是单次计数，且 CMP Value 设为 400K，因此当 CNT 超过 400K 后，Counter 会被 Disable，且 CNT register 也被清 0。

2.4.4.2 外部复位计数捕获模式

测试目的：

验证外部复位计数捕获模式（External Reset Counter Mode）工作是否正常。

测试预期：

能及时捕获外部事件（电平变化）并将 Counter 值载入 Capture Data Register 后清空 CNT Value Register。

测试现象：

输入‘B’命令，首先打印 Timer 设定基本信息，然后 PWM 产生一段方波，capture 中断被连续触发多次，同时打印当前 CNT Value 和 CAP Value 值。

```
b
Timer0 start, Cnt Freq:1000000Hz, Compare Value:255
Start external waveform signal (freq:20Hz, duty cycle:25%, duration:500ms)...
CAP01 CntVal:     7, CapVal: 19593
CAP02 CntVal:     0, CapVal: 12499
CAP03 CntVal:     0, CapVal: 37500
CAP04 CntVal:     0, CapVal: 12499
CAP05 CntVal:     0, CapVal: 37500
CAP06 CntVal:     0, CapVal: 12499
CAP07 CntVal:     0, CapVal: 37500
CAP08 CntVal:     0, CapVal: 12499
CAP09 CntVal:     0, CapVal: 37500
CAP10 CntVal:     0, CapVal: 12499
CAP11 CntVal:     0, CapVal: 37500
CAP12 CntVal:     0, CapVal: 12499
CAP13 CntVal:     0, CapVal: 37500
CAP14 CntVal:     0, CapVal: 12499
CAP15 CntVal:     0, CapVal: 37500
CAP16 CntVal:     0, CapVal: 12499
CAP17 CntVal:     0, CapVal: 37500
CAP18 CntVal:     0, CapVal: 12499
```

测试分析：

Timer 设定为计数频率 1MHz，CMP Value 为 255，计数模式为连续计数，捕获方式为双边沿（Both Edge）捕获。

外部 PWM 信号为频率 20Hz，占空比 25%，持续 500ms，因此有 20*0.5=10 个上升沿和 10

个下降沿，由占空比 25%可知相邻的高电平和低电平持续时间比值为 3:1。

由中断 Log 可以看出，PWM 持续期间共触发了 20 次中断，CntVal 除第一个为 7 外，其他全部为 0，可见中断触发的时候，CNT register 已经被硬件清 0。再纵向比较 CapVal 的值，可以看到相邻的 CapVal 比值为 3:1，亦即高电平与低电平持续时间比值是 3:1，由此同样可以推算出外部信号频率为：

1MHz /（12499+37500）= 20Hz，与 PWM 设定的信号一致。

另外，虽然设定的 CMP Value 为 255，但由于计数模式配置为连续计数，因此 CapVal 仍然可以超过 255，不受 CMP Value 影响。

2.4.4.3 触发计数捕获模式

测试目的：

验证触发计数捕获模式（Trigger-Counting Capture Mode）工作是否正常。

测试预期：

能及时捕获外部第一个事件（电平变化）并开始计数，并能及时捕获外部第二个事件，将 Counter 值载入 Capture Data Register。

测试现象：

输入‘C’命令，首先打印 Timer 设定基本信息，然后 PWM 产生一段方波，capture 中断被连续触发多次，同时打印当前 CNT Value 和 CAP Value 值。

```
c
Timer0 start, Cnt Freq:1000000Hz, Compare Value:16777215
Start external waveform signal (freq:20Hz, duty cycle:60%, duration:500ms)...
CAP01 CntVal:     0, CapVal: 29998
CAP02 CntVal:     0, CapVal:     0
CAP03 CntVal:     0, CapVal:     0
CAP04 CntVal:     0, CapVal:     0
CAP05 CntVal:     0, CapVal:     0
CAP06 CntVal:     0, CapVal:     0
CAP07 CntVal:     0, CapVal:     0
CAP08 CntVal:     0, CapVal:     0
CAP09 CntVal:     0, CapVal:     0
CAP10 CntVal:     0, CapVal:     0

+-----------------------------------------------------------------+
|    Press key to test specific function:                         |
|                                                                 |
|    Input 'A'    Free-Counting Capture Mode.                     |
|    Input 'B'    External Reset Counter Mode.                    |
|    Input 'C'    Trigger-Counting Capture Mode.                  |
|    Input 'D'    Internal RCL Capture Function.                  |
|    Press ESC key to back to the top level case list.            |
+-----------------------------------------------------------------+
```

测试分析：

Timer 设定为计数频率 1MHz，CMP Value 为 16777215（0xFFFFFF），计数模式为单次计数，捕获方式为“上升沿计数下降沿捕获”（Capture Rising Then Falling Edge）。

外部 PWM 信号为频率 20Hz，占空比 60%，持续 500ms，因此有 20*0.5=10 个上升沿和 10

个下降沿，由占空比 60%可知，一个周期内高电平时间为 1s/20 * 60% = 30ms。

由中断 Log 可以看出，PWM 持续期间共触发了 10 次 Capture 中断，CntVal 均为 0，CapVal 只有在第一条 Log 内有值 29998，其余均为 0；于是根据第一个 capval 的值即可计算出输入波形的上升沿持续时间为 29998/1MHz = 30ms，符合预期。

之所以触发 10 次中断，原因是 PWM 波共有 10 个下降沿，而此模式下只有下降沿才会触发中断，因此 10 次中断是预期的；CntVal 一直为 0，CapVal 只在第一次中断的时候有值，原因是第一次 Capture 到有效值后，硬件会将 Counter 关掉。

另外，之所以 Capture INT 会触发 10 次，是因为 Capture 到有效值后，只有 Counter 被关掉（CNTEN 置 0），但是 CAP INT 仍然有效，所以 Capture 中断可以一直被触发。

2.4.5 睡眠模式唤醒系统

在主菜单下，输入‘4’命令 进入 Subcase 菜单：

```
+-----------------------------------------------------------------+
|    Press key to test specific function:                         |
|                                                                 |
|    Input 'A'    Wakeup from power down mode.                    |
|    Press ESC key to back to the top level case list.            |
+-----------------------------------------------------------------+
```

测试目的：

验证 Timer 睡眠唤醒功能。

测试预期：

睡眠模式下能用 Timer 唤醒系统。

测试现象：

输入‘A’命令，Timer 的 CLK Source 设置为 RCL，然后设定 timeout 为 5s，并开启 wakeup EN。5s 后中断到来，Wakeup flag 成功被 set。

```
[14:55:27.462]发→◇A□
[14:55:27.467]收←◆
Timer1, Timeout:5s
Start Timer...
Enter sleep
[14:55:32.629]收←◆ mode...
CNT1
WAKEUP
sleep irq

+-----------------------------------------------------------------+
|    Press key to test specific function:                         |
|                                                                 |
|    Input 'A'    Wakeup from power down mode.                    |
|    Press ESC key to back to the top level case list.            |
+-----------------------------------------------------------------+
```

测试分析：

打印 WAKEUP 字样，测试成功。

2.4.6 定时器位宽验证

在主菜单下，输入‘5’命令 进入 Subcase 菜单：

```
Press key to choose timer:

Input 'A'     Timer0 selected.
Input 'B'     Timer1 selected.
Input 'C'     Timer2 selected.
Press ESC key to back to the top level case list.
```

2.4.6.1 Timer0 selected

测试目的：

验证定时器 timer0 的位宽，通过观察在计数值大于 24bits 是否会溢出，从而确定其是否为 32 位。

测试预期：

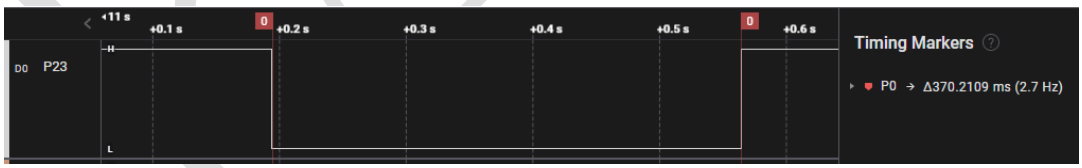
由于 timer0 是 32 位的，设置一个值为 17,760,256 并不应导致计数溢出，Timeout 完整。

测试现象：

对于 timer0，输入‘A’命令。

由 Log 可以看出，设置的比较值为 17,760,256。期望和实际的计数频率均为 48,000,000。期望和实际的超时时间均为 370ms。观察波形 Timeout 完整，表明 timer0 计数没有溢出。

```
Timer0, Compare Value:17760256
Expected Cnt Freq:48000000, Real Cnt Freq:48000000
Expected Timeout:370ms, Real Timeout:370ms
Start Timer...
CNT1
TIMER Test OK, Success case: 5
```



测试分析：

Timer0 计数至 17,760,256 时没有溢出，说明它是 32 位定时器。

2.4.6.2 Timer1 selected

测试目的：

验证定时器 timer1 的位宽，通过观察在计数值大于 24bits 是否会溢出，从而确定其是否为

24 位。

测试预期：

由于 timer1 是 24 位的，当计数值达到 17,760,256 时应发生溢出，Timeout 减少。

测试现象：

对于 timer1，输入‘B’命令。

由 Log 可以看出，设置的比较值为 17,760,256。期望和实际的计数频率均为 48,000,000。期望和实际的超时时间均减少至 20ms。观察波形 Timeout 减少至 20ms，表明 timer1 计数已溢出。

```
Timer1, Compare Value:17760256
Expected Cnt Freq:48000000, Real Cnt Freq:48000000
Expected Timeout:20ms, Real Timeout:20ms
Start Timer...
CNT1
TIMER Test OK, Success case: 5
```

测试分析：

Timer1 在计数至 17,760,256 时出现溢出，超时时间减少至 20ms，说明它是 24 位定时器。

2.4.6.3 Timer2 selected

测试目的：

验证定时器 timer2 的位宽，通过观察在计数值大于 24bits 是否会溢出，从而确定其是否为 24 位。

测试预期：

由于 timer2 是 24 位的，当计数值达到 17,760,256 时应发生溢出，Timeout 减少。

测试现象：

对于 timer2，输入‘B’命令。

由 Log 可以看出，设置的比较值为 17,760,256。期望和实际的计数频率均为 48,000,000。期望和实际的超时时间均减少至 20ms。观察波形 Timeout 减少至 20ms，表明 timer2 计数已溢出。

```
Timer2, Compare Value:17760256
Expected Cnt Freq:48000000, Real Cnt Freq:48000000
Expected Timeout:20ms, Real Timeout:20ms
Start Timer...
CNT1
TIMER Test OK, Success case: 5
```

测试分析：

Timer2 在计数至 17,760,256 时出现溢出，超时时间减少至 20ms，说明它是 24 位定时器。

第3章 使用注意事项

1、 CMP DATA 寄存器应设置为 0 和 1 以外的值，否则可能会进入未知状态

2、 Toggle output mode 下，输出电平改变的前提是 TIF flag 是被清除的状态（0），所以此模式必须开中断，然后在中断 handler 中将 TIF flag 清掉，以保证 output 电平变化正常

3、 Event Counter Mode EN（CTL bit24）与 Capture EN（EXTCTL bit3）不可同时置 1，否则会得不到预期结果

4、 TIMER_Stop()只是暂停计数，CNT Val 不会被清掉。如果希望清除 CNT Val，应使用 TIMER_Reset()或 TIMER_Close()。