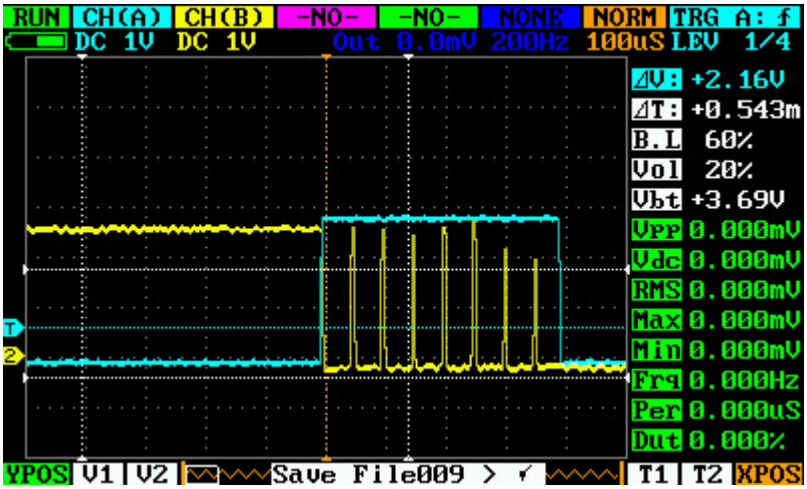
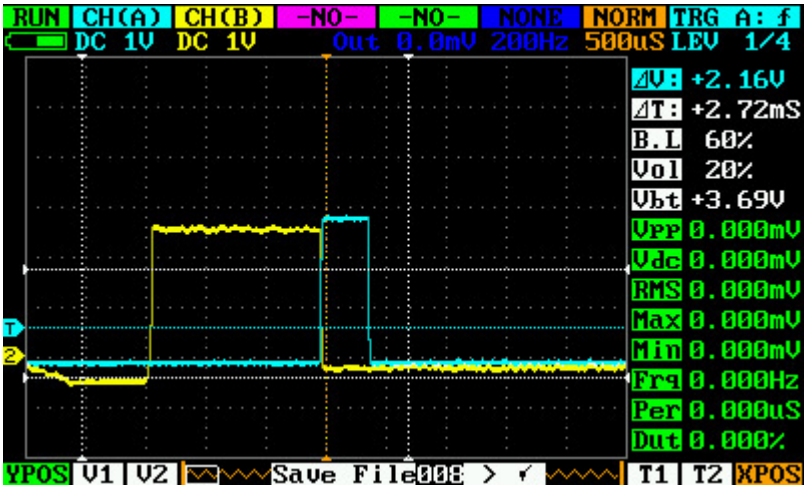
I'm hammering my table with my head the whole evening. I can't get the simpliest thing to work. Ok, here's the problem: I need TIM1 to gate TIM3. I've got TIM1 configured as a single-shot master with either TIM_TRGOSource_Enable or TIM_TRGOSource_OC1Ref signal sent to slaves. Then I have TIM3 set as a slave PWM generator. I want TIM3 to immidiately start firing pulses. Pictures will tell the better story:



The blue waveform is TIM1, yellow is TIM3. Notice that TIM3 is now acting with TIM_OCPolarity_High and in PWM1 mode. _ **Absolutely the same picture**_ is when TIM_OCPolarity_Low and PWM2 mode. And this is crazy, because notice that yellow is HIGH _before_ the gate comes to the scene. The same picture with different scale:



TIM3_CH1 goes high immediately after calling TIM_OC1Init(TIM3, &TIM_OCInitStructure); It is the combination of polarity and PWM mode that defines if the output will be HIGH or LOW. Timer have not even started counting, and the output goes high.

I believe this is a very bad bug here.Any comments/advices?

```
001. #include "stm32f10x_tim.h"
002. #include "stm32f10x_rcc.h"
003. #include "stm32f10x_gpio.h"
004.
005. TIM_TimeBaseInitTypeDef  TIM_TimeBaseStructure;
006. TIM_OCInitTypeDef  TIM_OCInitStructure;
007. TIM_BDTRInitTypeDef TIM_BDTRInitStructure;
008.
009. void RCC_Configuration(void);
010. void GPIO_Configuration(void);
011.
012. void tim3ch3onePulse()
013. {
```

```
014.     /*!< At this stage the microcontroller clock setting is already configured,
015.            this is done through SystemInit() function which is called from startup
016.            file (startup_stm32f10x_xx.s) before to branch to application main.
017.            To reconfigure the default setting of SystemInit() function, refer to
018.            system_stm32f10x.c file
019.      */
020.
021.     /* System Clocks Configuration */
022.     RCC_Configuration();
023.
024.     /* GPIO Configuration */
025.     GPIO_Configuration();
026.
027.     /* TIM3 Peripheral Configuration ----------------------------------------*/
028.
029.     /* TIM3 works in Gated mode. */
030.
031.     TIM_SelectSlaveMode(TIM3, TIM_SlaveMode_Gated);
032.     TIM_SelectInputTrigger(TIM3, TIM_TS_ITR0);
033.
034.     /* TIM3 Slave Configuration: PWM1 Mode */
035.     TIM_TimeBaseStructure.TIM_Prescaler = 23;
036.     TIM_TimeBaseStructure.TIM_Period = 50;
037.     TIM_TimeBaseStructure.TIM_ClockDivision = 0;
038.     TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
039.
040.     TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);
041.
042.
043.     TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
044.     TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
045.     TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
046.     TIM_OCInitStructure.TIM_Pulse = 5;
047.
048.     TIM_OC1Init(TIM3, &TIM_OCInitStructure);
049.
050.     /* TIM1 Peripheral Configuration ----------------------------------------*/
051.     /* Time Base configuration */
052.     TIM_TimeBaseStructure.TIM_Prescaler = 23;
053.     TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
054.     TIM_TimeBaseStructure.TIM_Period = 401;
055.     TIM_TimeBaseStructure.TIM_ClockDivision = 0;
056.     TIM_TimeBaseStructure.TIM_RepetitionCounter = 0;
057.
058.     TIM_TimeBaseInit(TIM1, &TIM_TimeBaseStructure);
059.
060.     /* Channel 1 Configuration in PWM mode */
061.     TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
062.     TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
063.     TIM_OCInitStructure.TIM_OutputNState = TIM_OutputNState_Disable;
064.     TIM_OCInitStructure.TIM_Pulse = 300;
065.     TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
066.     TIM_OCInitStructure.TIM_OCNPolarity = TIM_OCNPolarity_Low;
067.     TIM_OCInitStructure.TIM_OCIdleState = TIM_OCIdleState_Reset;
068.     TIM_OCInitStructure.TIM_OCNIdleState = TIM_OCIdleState_Reset;
069.
070.     TIM_OC1Init(TIM1, &TIM_OCInitStructure);
071.
072.     /* Master Mode selection */
073.     TIM_SelectOutputTrigger(TIM1, TIM_TRGOSource_Enable);
074.
075.     /* Select the Master Slave Mode */
076.     TIM_SelectMasterSlaveMode(TIM1, TIM_MasterSlaveMode_Enable);
077.
078.     TIM_CtrlPWMOutputs(TIM1, ENABLE);
079.
```

```
080.     TIM_SelectOnePulseMode(TIM1, TIM_OPMode_Single);
081.
082.     /* TIM enable counter */
083.
084.
085.     /* Main Output Enable */
086.     /* TIM1 counter enable */
087.     TIM_Cmd(TIM1, ENABLE);
088.     TIM_Cmd(TIM3, ENABLE);
089.
090.
091.
092.     while (1)
093.     {}
094. }
095.
096. void GPIO_Configuration(void)
097. {
098.     GPIO_InitTypeDef GPIO_InitStructure;
099.
100.
101.     /* GPIOA Configuration: TIM1 Channel1 and TIM3 Channel1 as alternate function push-pull
*/
102.     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6 | GPIO_Pin_8;
103.     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
104.     GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
105.     GPIO_Init(GPIOA, &GPIO_InitStructure);
106.     GPIO_ResetBits(GPIOA, GPIO_Pin_6);
107. }
108.
109. void RCC_Configuration(void)
110. {
111.     /* TIM1, GPIOA and GPIOB clock enable */
112.     RCC_APB2PeriphClockCmd(RCC_APB2Periph_TIM1 | RCC_APB2Periph_GPIOA |
113.             RCC_APB2Periph_GPIOB | RCC_APB2Periph_AFIO, ENABLE);
114.
115.     /* TIM3 and TIM4 clock enable */
116.     RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);
117. }
```