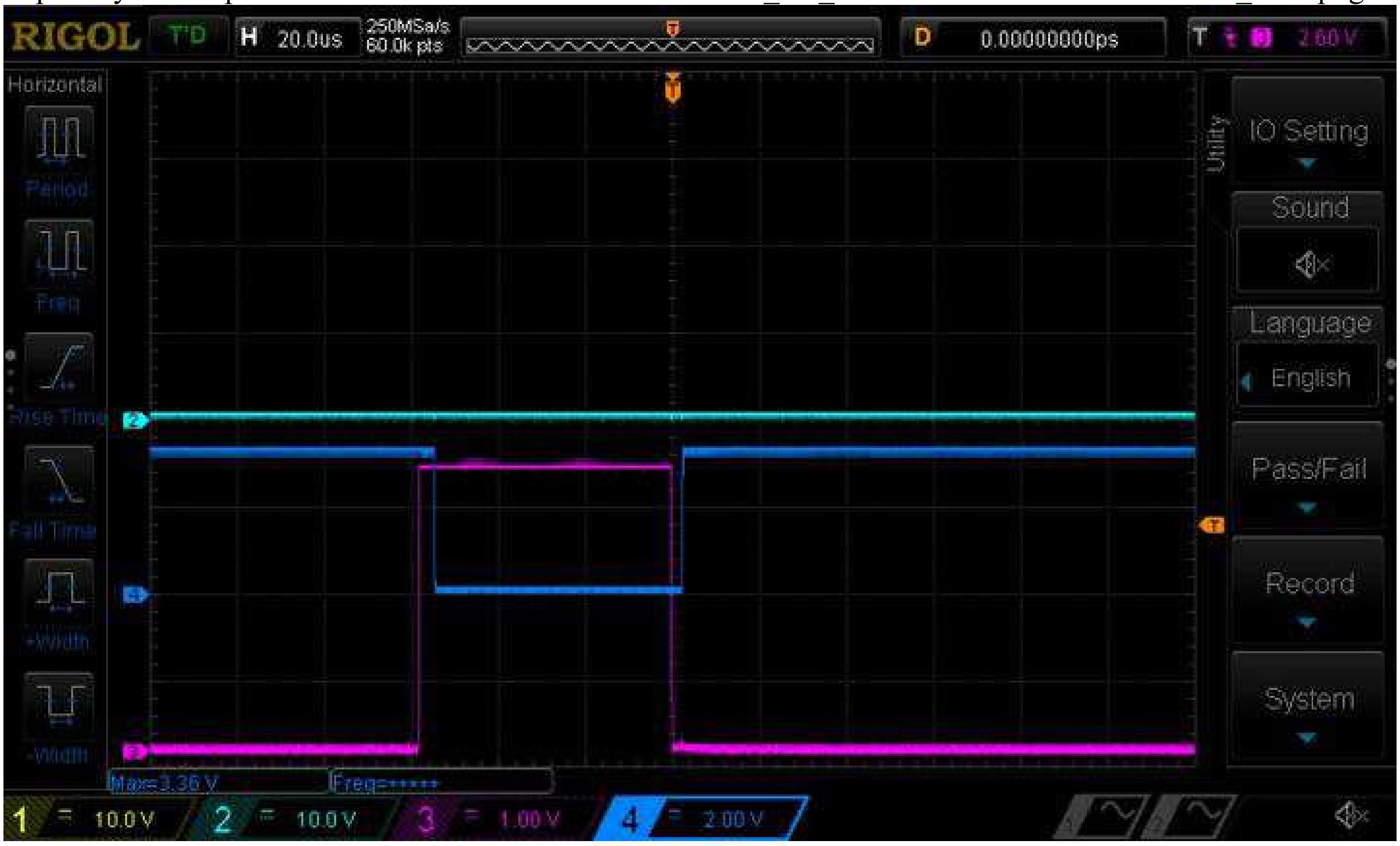
STM32F103RC
I am trying to trigger ADC1 from Timer 1 CC 2, but nothing happens.
Timer 1 is configured with PWM and CC interrupt.
Triggering from Timer 1 CC 1 causes EOC Interrupt to execute on timer update (PWM rising edge) for some reason.
ADC interrupt works properly when triggering conversion in software (also from CC interrupt).

See attached scope capture (blue debug pin, pink PWM):
https://my.st.com/public/STe2ecommunities/mcu/Lists/cortex_mx_stm32/Attachments/70530/t1cc1_adc1.png



Is there more documentation on this than what I found in the Reference Manual (DocID13902 Rev 16) ?

> 11.7 Conversion on external trigger
> Conversion can be triggered by an external event (e.g. timer capture, EXTI line). If the EXTTRIG control bit is set then external events are able to trigger a conversion. The EXTSEL[2:0] and JEXTSEL[2:0] control bits allow the application to select decide which out of 8 possible events can trigger conversion for the regular and injected groups.
> Note: When an external trigger is selected for ADC regular or injected conversion, only the rising edge of the signal can start the conversion.
> Table 67. External trigger for regular channels for ADC1 and ADC2

The SPL code examples and everything else I found so far use DMA.
The Errata does not seem to mention anything that applies to my issue.

So aside from the different behaviour of the two compare units, I must be forgetting to configure something.

```
// attempting to trigger ADC conversion with Timer 1 Compare 2
//
// Softtrigger works as expected
// Trigger with CC1:
//   ADC interrupt occurs just after timer update/reset (when PWM output pin goes high)
// Trigger with CC2 does not work at all

// device: STM32F103RCT6 / 0x414
//   LQFP64
//   rev Y / 0x1003 (ST-Link Utility 3.9: rev X)
//   datecode 321
//

// 8 MHz crystal -> 72 MHz
// compiler
/*
   C:\Program Files (x86)\EmBitz\1.00\share\em_armgcc\bin>arm-none-eabi-gcc-5.4.1.exe --version
   arm-none-eabi-gcc-5.4.1.exe (GNU Tools for ARM Embedded Processors) 5.4.1 20160609 (release) [ARM/embedded-5-branch revision 237715]
   Copyright (C) 2015 Free Software Foundation, Inc.
   This is free software; see the source for copying conditions.  There is NO
   warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
*/
// SPL 3.5

#include "stm32f10x_conf.h"


//#define SOFTTRIGGER
#define CC1
//#define CC2



int main(void)
{

    SystemInit();


    // Interrupts
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_4);

    NVIC_InitTypeDef NVIC_InitStructure;

    NVIC_InitStructure.NVIC_IRQChannel = ADC1_2_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0x00;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x01;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);


    NVIC_InitStructure.NVIC_IRQChannel = TIM1_CC_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0x00;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x00;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);



    // Clocks

    RCC_ADCCLKConfig(RCC_PCLK2_Div6);

    RCC_APB2PeriphClockCmd(
        RCC_APB2Periph_GPIOA |
        RCC_APB2Periph_GPIOB |
        //RCC_APB2Periph_AFIO  |
        RCC_APB2Periph_TIM1  |
        RCC_APB2Periph_ADC1  ,
        ENABLE
        );



    // IO

    GPIO_InitTypeDef GPIO_InitStructure;

    GPIO_StructInit(&GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;       // adc
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;

    GPIO_Init(GPIOA, &GPIO_InitStructure);


    GPIO_StructInit( &GPIO_InitStructure );

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;     // timer 1 pwm
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;

    GPIO_Init(GPIOA, &GPIO_InitStructure);


    GPIO_StructInit( &GPIO_InitStructure );

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_14;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;

    GPIO_Init(GPIOB, &GPIO_InitStructure);



    // PWM Timer 1

    TIM_TimeBaseInitTypeDef TIM_TimeBase_InitStructure;

    TIM_TimeBaseStructInit( &TIM_TimeBase_InitStructure );

    TIM_TimeBase_InitStructure.TIM_CounterMode =    TIM_CounterMode_Up;
    TIM_TimeBase_InitStructure.TIM_ClockDivision =  TIM_CKD_DIV1;
    TIM_TimeBase_InitStructure.TIM_Prescaler =      32;
    TIM_TimeBase_InitStructure.TIM_Period =         1023;

    TIM_TimeBaseInit(TIM1, &TIM_TimeBase_InitStructure);



    // CC1 PWM
    TIM_OCInitTypeDef   TIM_OC_InitStructure;

    TIM_OCStructInit( &TIM_OC_InitStructure );

    TIM_OC_InitStructure.TIM_OCMode =       TIM_OCMode_PWM1;
    TIM_OC_InitStructure.TIM_OCIdleState =  TIM_OCIdleState_Reset;
    TIM_OC_InitStructure.TIM_OCNIdleState = TIM_OCNIdleState_Reset;
    TIM_OC_InitStructure.TIM_OCPolarity =   TIM_OCPolarity_High;
    TIM_OC_InitStructure.TIM_OCNPolarity =  TIM_OCNPolarity_Low;
    TIM_OC_InitStructure.TIM_Pulse =        127;
    TIM_OC_InitStructure.TIM_OutputState =  TIM_OutputState_Enable;
    TIM_OC_InitStructure.TIM_OutputNState = TIM_OutputNState_Disable;
    TIM_OC1Init(TIM1, &TIM_OC_InitStructure);

    TIM_OC1PreloadConfig(TIM1, TIM_OCPreload_Enable);

#ifdef CC1
```

```
    TIM_ClearITPendingBit(TIM1, TIM_IT_CC1);
    TIM_ITConfig(TIM1, TIM_IT_CC1, ENABLE);
#endif


    // CC2 ADC
    TIM_OCStructInit( &TIM_OC_InitStructure );
    TIM_OC_InitStructure.TIM_Pulse =        511;
    TIM_OC2Init(TIM1, &TIM_OC_InitStructure);

    TIM_OC2PreloadConfig(TIM1, TIM_OCPreload_Enable);

#ifdef CC2
    TIM_ClearITPendingBit(TIM1, TIM_IT_CC2);
    TIM_ITConfig(TIM1, TIM_IT_CC2, ENABLE);
#endif



    TIM_Cmd(TIM1, ENABLE);

    TIM_CtrlPWMOutputs(TIM1, ENABLE);



    // ADC

    ADC_InitTypeDef ADC_InitStructure;

    ADC_StructInit(&ADC_InitStructure);

    ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;



#ifdef SOFTTRIGGER
    ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
#else
    #ifdef CC1
    ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_T1_CC1;
    #endif
    #ifdef CC2
    ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_T1_CC2;
    #endif
#endif
    ADC_InitStructure.ADC_ContinuousConvMode = DISABLE;
    ADC_InitStructure.ADC_ScanConvMode = DISABLE;
    ADC_InitStructure.ADC_NbrOfChannel = 1;
    ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;

    ADC_Init(ADC1, &ADC_InitStructure);

    ADC_RegularChannelConfig(
         ADC1,
         ADC_Channel_1,
         1,                         // The rank in the regular group sequencer. This
parameter must be between 1 to 16.
         ADC_SampleTime_1Cycles5
         );

#ifndef SOFTTRIGGER
    ADC_ExternalTrigConvCmd(ADC1, ENABLE);
#endif

    ADC_ClearITPendingBit(ADC1, ADC_IT_EOC);
    ADC_ITConfig(ADC1, ADC_IT_EOC, ENABLE);

    ADC_Cmd(ADC1, ENABLE);


    // calibrate
    ADC_ResetCalibration(ADC1);
    while(ADC_GetResetCalibrationStatus(ADC1));

    ADC_StartCalibration(ADC1);
    while(ADC_GetCalibrationStatus(ADC1));



/*
#ifdef SOFTTRIGGER
    ADC_SoftwareStartConvCmd(ADC1, ENABLE);
    //while(!ADC_GetFlagStatus(ADC1, ADC_FLAG_EOC));
    //ADC_ClearFlag(ADC1, ADC_FLAG_EOC);
    //uint16_t value = ADC_GetConversionValue(ADC1);
#endif
*/


  while(1)
  {

  }
}



void ADC1_2_IRQHandler(void){


        if( ADC_GetITStatus(ADC1, ADC_IT_EOC) == SET){
            ADC_ClearITPendingBit(ADC1, ADC_IT_EOC);

           // GPIO_SetBits(GPIOB, GPIO_Pin_14);
            GPIO_ResetBits(GPIOB, GPIO_Pin_14);

        }else
        {
            asm("NOP");
        }
}



void TIM1_CC_IRQHandler(void)
{
    if( TIM_GetITStatus(TIM1, TIM_IT_CC1) == SET ){
        TIM_ClearITPendingBit(TIM1, TIM_IT_CC1);
        GPIO_SetBits(GPIOB, GPIO_Pin_14);

        #ifdef SOFTTRIGGER
        ADC_SoftwareStartConvCmd(ADC1, ENABLE);
        #endif
    }


    if( TIM_GetITStatus(TIM1, TIM_IT_CC2) == SET ){
        TIM_ClearITPendingBit(TIM1, TIM_IT_CC2);
        GPIO_SetBits(GPIOB, GPIO_Pin_14);
       // GPIO_ResetBits(GPIOB, GPIO_Pin_14);

```

```
        #ifdef SOFTTRIGGER
        ADC_SoftwareStartConvCmd(ADC1, ENABLE);
        #endif
    }
}


void NMI_Handler(void){asm("NOP");}                  //
void HardFault_Handler(void){while(1);}              //
void MemManage_Handler(void){asm("NOP");}            //
void BusFault_Handler(void){while(1);}               //
void UsageFault_Handler(void){while(1);}             //
void SVC_Handler(void){asm("NOP");}                  //
void DebugMon_Handler(void){asm("NOP");}             //
void PendSV_Handler(void){asm("NOP");}               //
void SysTick_Handler(void){asm("NOP");}              //
void Default_Handler(void){asm("NOP");}              //

// External Interrupts
void WWDG_IRQHandler(void){asm("NOP");}             // Window Watchdog
void PVD_IRQHandler(void){asm("NOP");}              // PVD through EXTI Line detect
void TAMPER_IRQHandler(void){asm("NOP");}           // Tamper
void RTC_IRQHandler(void){asm("NOP");}              // RTC
void FLASH_IRQHandler(void){asm("NOP");}            // Flash
void RCC_IRQHandler(void){asm("NOP");}              // RCC
void EXTI0_IRQHandler(void){asm("NOP");}            // EXTI Line 0
void EXTI1_IRQHandler(void){asm("NOP");}            // EXTI Line 1
void EXTI2_IRQHandler(void){asm("NOP");}            // EXTI Line 2
void EXTI3_IRQHandler(void){asm("NOP");}            // EXTI Line 3
void EXTI4_IRQHandler(void){asm("NOP");}            // EXTI Line 4
void DMA1_Channel1_IRQHandler(void){asm("NOP");}    // DMA1 Channel 1
void DMA1_Channel2_IRQHandler(void){asm("NOP");}    // DMA1 Channel 2
void DMA1_Channel3_IRQHandler(void){asm("NOP");}    // DMA1 Channel 3
void DMA1_Channel4_IRQHandler(void){asm("NOP");}    // DMA1 Channel 4
void DMA1_Channel5_IRQHandler(void){asm("NOP");}    // DMA1 Channel 5
void DMA1_Channel6_IRQHandler(void){asm("NOP");}    // DMA1 Channel 6
void DMA1_Channel7_IRQHandler(void){asm("NOP");}    // DMA1 Channel 7
//void ADC1_2_IRQHandler(void){asm("NOP");}           // ADC1 & ADC2
void USB_HP_CAN1_TX_IRQHandler(void){asm("NOP");}   // USB High Priority or CAN1 TX
void USB_LP_CAN1_RX0_IRQHandler(void){asm("NOP");} // USB Low  Priority or CAN1 RX0
void CAN1_RX1_IRQHandler(void){asm("NOP");}         // CAN1 RX1
void CAN1_SCE_IRQHandler(void){asm("NOP");}         // CAN1 SCE
void EXTI9_5_IRQHandler(void){asm("NOP");}          // EXTI Line 9..5
void TIM1_BRK_IRQHandler(void){asm("NOP");}         // TIM1 Break
void TIM1_UP_IRQHandler(void){asm("NOP");}          // TIM1 Update
void TIM1_TRG_COM_IRQHandler(void){asm("NOP");}     // TIM1 Trigger and Commutation
//void TIM1_CC_IRQHandler(void){asm("NOP");}          // TIM1 Capture Compare
void TIM2_IRQHandler(void){asm("NOP");}             // TIM2
void TIM3_IRQHandler(void){asm("NOP");}             // TIM3
void TIM4_IRQHandler(void){asm("NOP");}             // TIM4
void I2C1_EV_IRQHandler(void){asm("NOP");}          // I2C1 Event
void I2C1_ER_IRQHandler(void){asm("NOP");}          // I2C1 Error
void I2C2_EV_IRQHandler(void){asm("NOP");}          // I2C2 Event
void I2C2_ER_IRQHandler(void){asm("NOP");}          // I2C2 Error
void SPI1_IRQHandler(void){asm("NOP");}             // SPI1
void SPI2_IRQHandler(void){asm("NOP");}             // SPI2
void USART1_IRQHandler(void){asm("NOP");}           // USART1
void USART2_IRQHandler(void){asm("NOP");}           // USART2
void USART3_IRQHandler(void){asm("NOP");}           // USART3
void EXTI15_10_IRQHandler(void){asm("NOP");}        // EXTI Line 15..10
void RTCAlarm_IRQHandler(void){asm("NOP");}         // RTC Alarm through EXTI Line
void USBWakeUp_IRQHandler(void){asm("NOP");}        // USB Wakeup from suspend
void TIM8_BRK_IRQHandler(void){asm("NOP");}         // TIM8 Break
void TIM8_UP_IRQHandler(void){asm("NOP");}          // TIM8 Update
void TIM8_TRG_COM_IRQHandler(void){asm("NOP");}     // TIM8 Trigger and Commutation
void TIM8_CC_IRQHandler(void){asm("NOP");}          // TIM8 Capture Compare
void ADC3_IRQHandler(void){asm("NOP");}             // ADC3
void FSMC_IRQHandler(void){asm("NOP");}             // FSMC
void SDIO_IRQHandler(void){asm("NOP");}             // SDIO
void TIM5_IRQHandler(void){asm("NOP");}             // TIM5
void SPI3_IRQHandler(void){asm("NOP");}             // SPI3
void UART4_IRQHandler(void){asm("NOP");}            // UART4
void UART5_IRQHandler(void){asm("NOP");}            // UART5
void TIM6_IRQHandler(void){asm("NOP");}             // TIM6
void TIM7_IRQHandler(void){asm("NOP");}             // TIM7
void DMA2_Channel1_IRQHandler(void){asm("NOP");}    // DMA2 Channel1
void DMA2_Channel2_IRQHandler(void){asm("NOP");}    // DMA2 Channel2
void DMA2_Channel3_IRQHandler(void){asm("NOP");}    // DMA2 Channel3
void DMA2_Channel4_5_IRQHandler(void){asm("NOP");} // DMA2 Channel4 & Channel5
```