Timer/Counter

MIAT STM32 Development Kit

謝昇憲
support.wuyang@gmail.com

浯陽科技有限公司
WU-YANG
Technology Co., Ltd.

Declared Version

Training Only	
Declare	
Document Number	
Document Version	1.00
Release Date	2009.06.20
Document Title	Timer / Counter
Exercise Time	■ Lecture 90 minutes ■ Operating 90 minutes
Platform	■ MIAT STM32F10x EVB ■ MIAT IOB
Peripheral	■ Text LCD ■ LED
Author	■Wu-Yang
Key Word	STM32F10x, Timer, Counter

實驗目的

嵌入式系統中Timer / Counter是相當常用的功能，本章將介紹瞭解ARM Cortex-M3 Timer / Counter ，並使讀者瞭解其使用方式。

實做重點

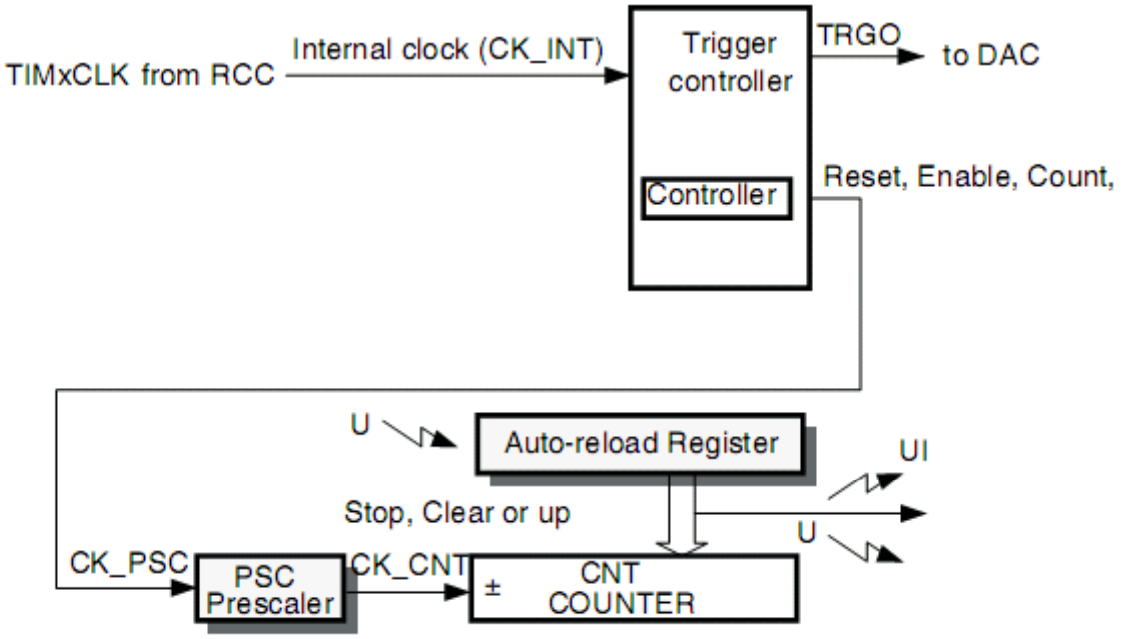
- 控制4個timer產生4個不同的時間訊號
- 控制1個timer的4個channel產生4個不同的時間訊號
- 設計一個簡易碼錶

實驗原理

- Introduction ARM Cortex-M3 Timer
- Development Flow
- ARM Configure

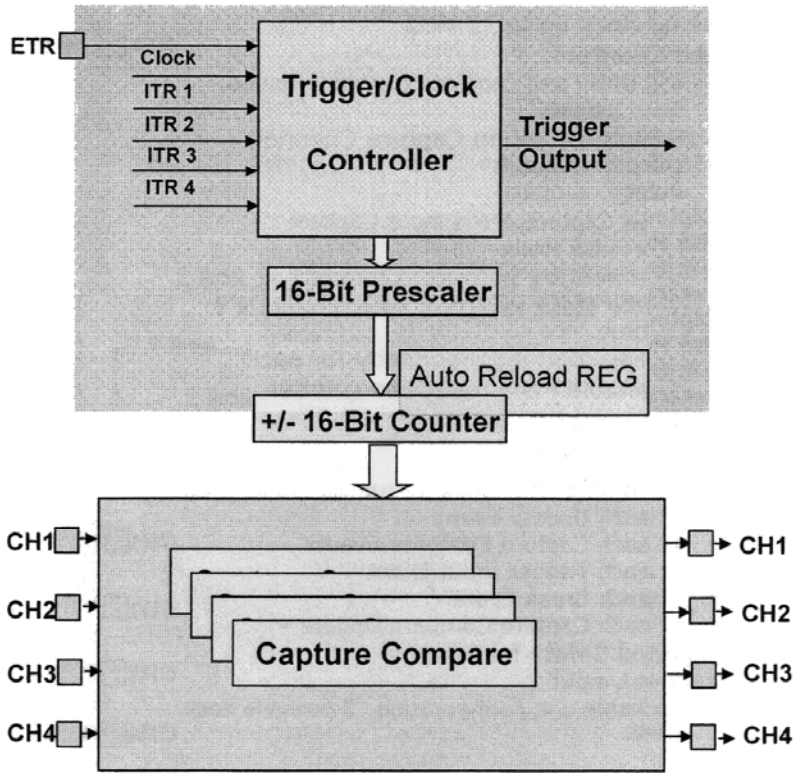
Basic timer Feature Overview

- TIM 6, 47on Low Speed APB(APB1)
- Internal clock up to 72Mhz
- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide the counter clock
- Synchronization circuit to trigger the DAC
- Interrupt/DMA generation on the update event: counter overflow



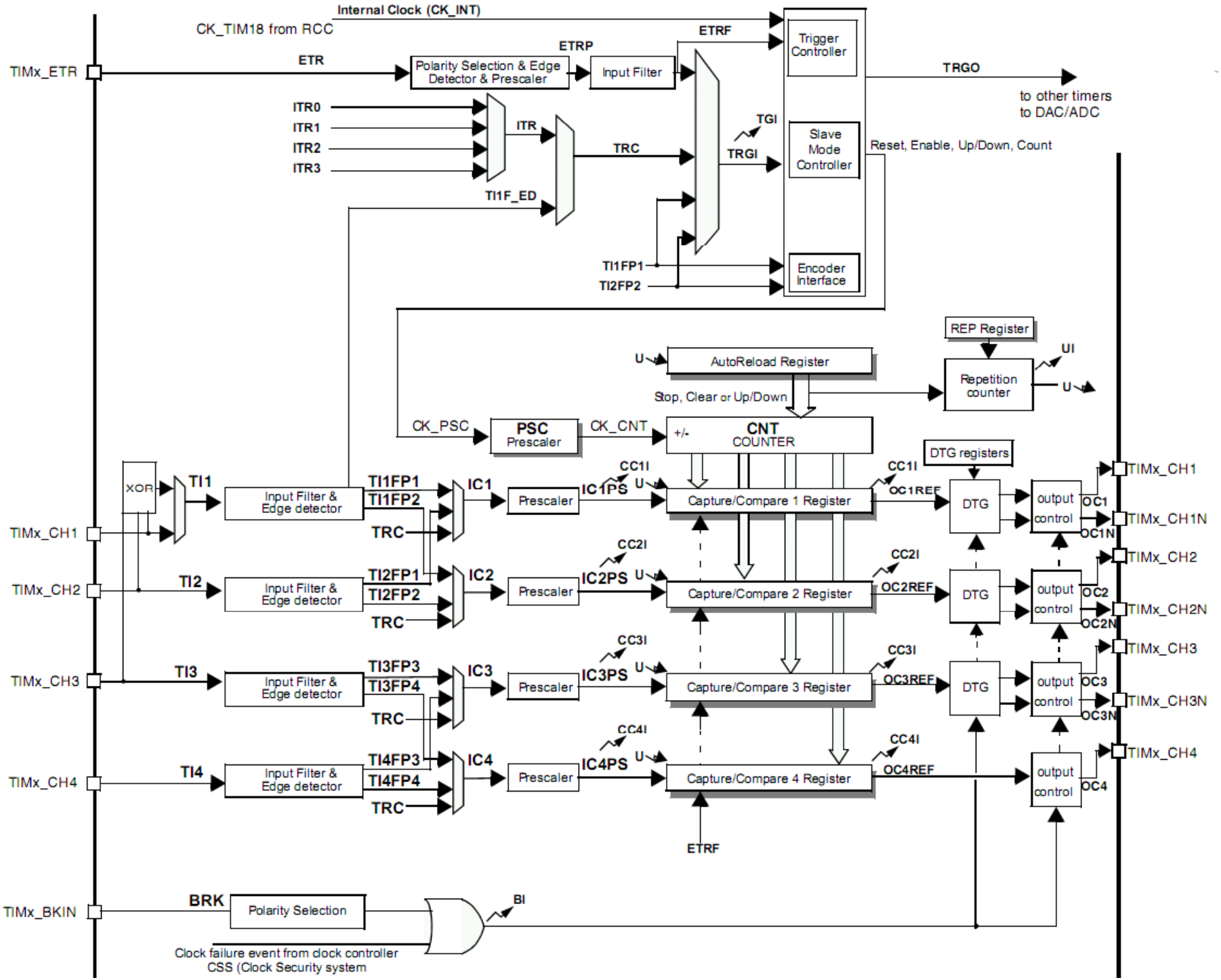
General Purpose timer Feature Overview

- TIM2, 3, 4 on Low Speed APB(APB1)
- Internal clock up to 72Mhz
- 16bit Counter
  - ✓ Up, Down, centered counting mode
  - ✓ Auto Reload
- 4 x 16 High resolution Capture Channels
  - ✓ Programmable direction of the channel inpu
  - ✓ Output Compare
  - ✓ PWM
  - ✓ Input Capture, PWM Input Capture
  - ✓ One Pulse Mode
- Synchronization
  - ✓ Timer Master/Slave
  - ✓ Triggered or gated mode
- Encoder Interface
- 6 Independent IRQ/DMA Request generation
  - ✓ At each Update Event
  - ✓ At each Capture Compare Events
  - ✓ At each Input Trigger



Advanced control timer Feature Overview

- TIM1, 8 on High Speed APB(APB2)
- Complementary Outputs with programmable dead-time
- Break input to put the timer's output signals in reset state or in a known state.

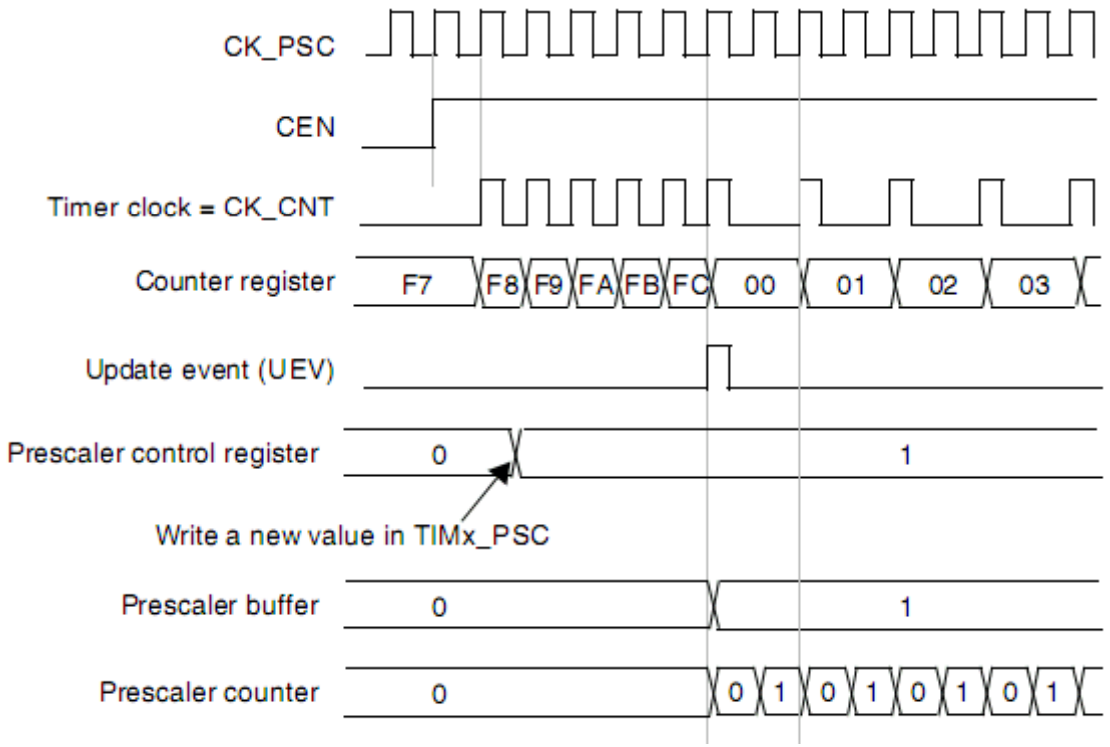

Internal Clock (CK_INT)
CK_TIM18 from RCC
TIMx_ETR
ETR
ETRP
Polarity Selection & Edge Detector & Prescaler
Input Filter
ETRF
Trigger Controller
TRGO
to other timers
to DAC/ADC
ITR0
ITR1
ITR2
ITR3
ITR
TRC
TGI
TRGI
Slave Mode Controller
Reset, Enable, Up/Down, Count
TI1F_ED
TI1FP1
TI2FP2
Encoder Interface
REP Register
AutoReload Register
UI
Repetition counter
Stop, Clear or Up/Down
CK_PSC
PSC Prescaler
CK_CNT
CNT COUNTER
DTG registers
XOR
TI1
TI2
TI3
TI4
Input Filter & Edge detector
TI1FP1
TI1FP2
TI2FP1
TI2FP2
TI3FP3
TI3FP4
TI4FP3
TI4FP4
IC1
IC2
IC3
IC4
Prescaler
IC1PS
IC2PS
IC3PS
IC4PS
CC1I
CC2I
CC3I
CC4I
Capture/Compare 1 Register
Capture/Compare 2 Register
Capture/Compare 3 Register
Capture/Compare 4 Register
OC1REF
OC2REF
OC3REF
OC4REF
DTG
output control
OC1
OC1N
OC2
OC2N
OC3
OC3N
OC4
TIMx_CH1
TIMx_CH2
TIMx_CH3
TIMx_CH4
TIMx_CH1N
TIMx_CH2N
TIMx_CH3N
TIMx_BKIN
BRK
Polarity Selection
BI
Clock failure event from clock controller
CSS (Clock Security system

Time-base unit

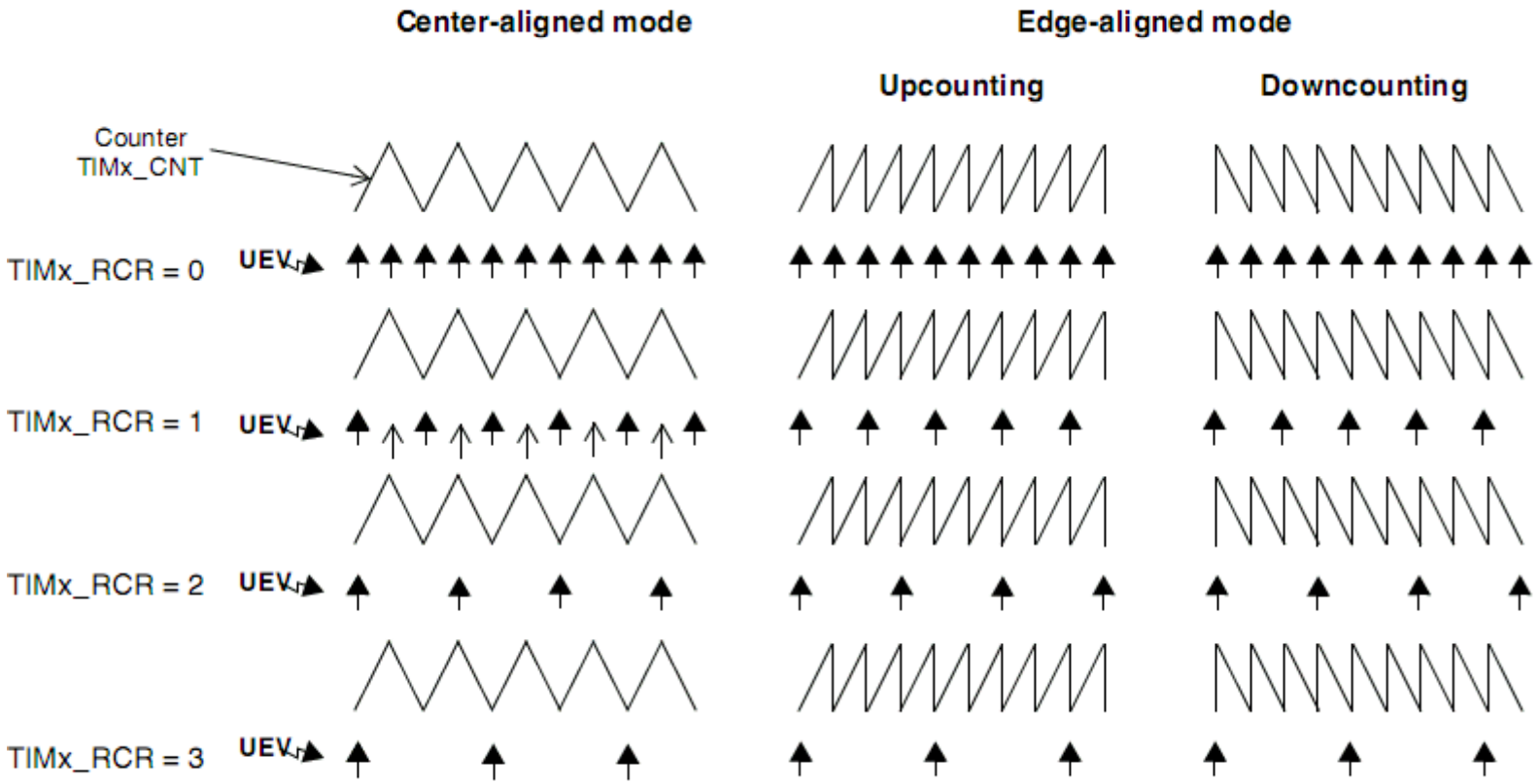
- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

Prescaler register

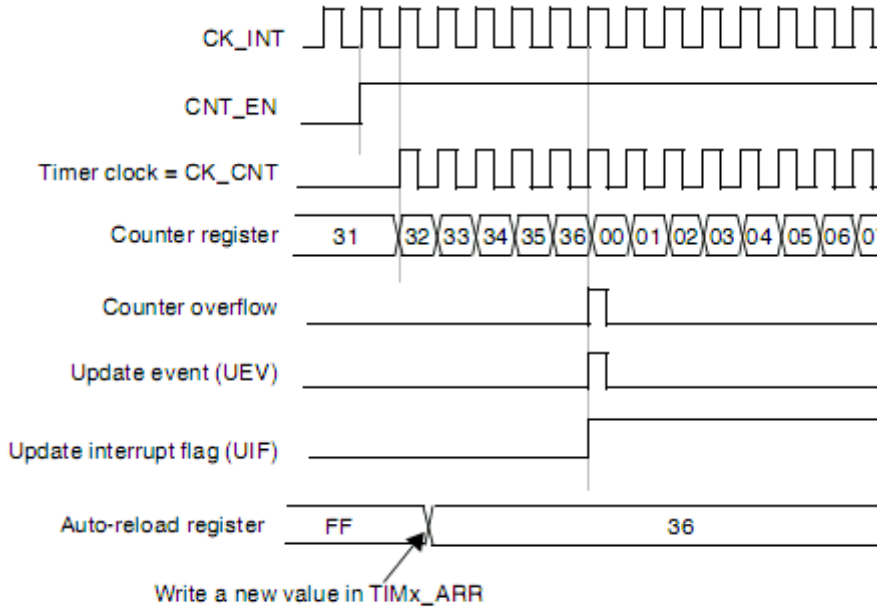
Counter timing diagram with prescaler division change from 1 to 2

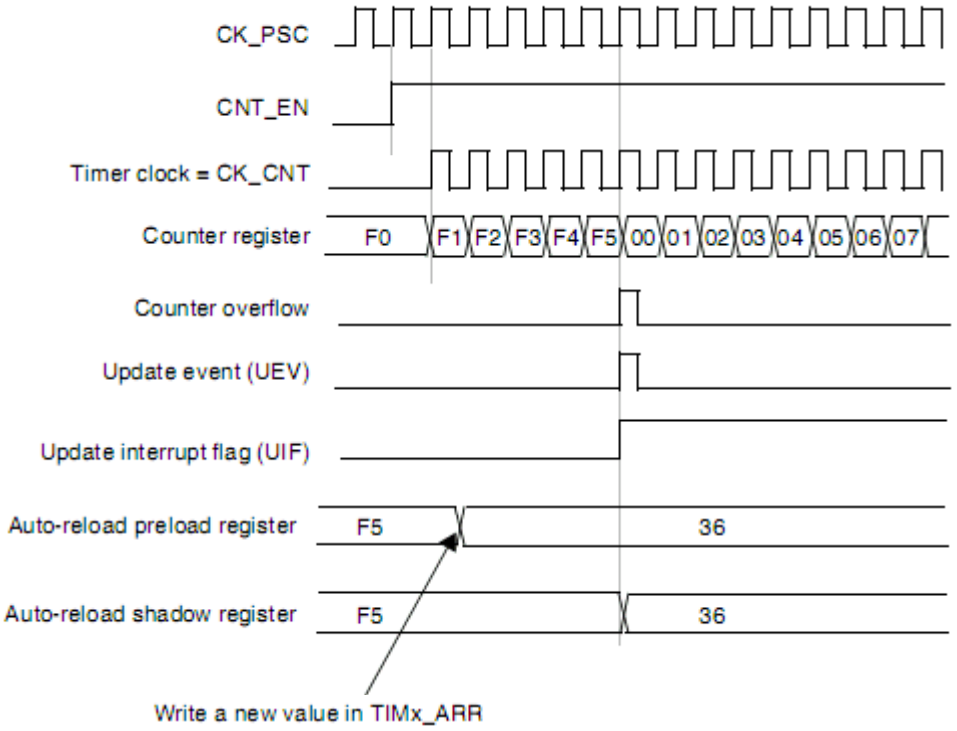


Counter modes



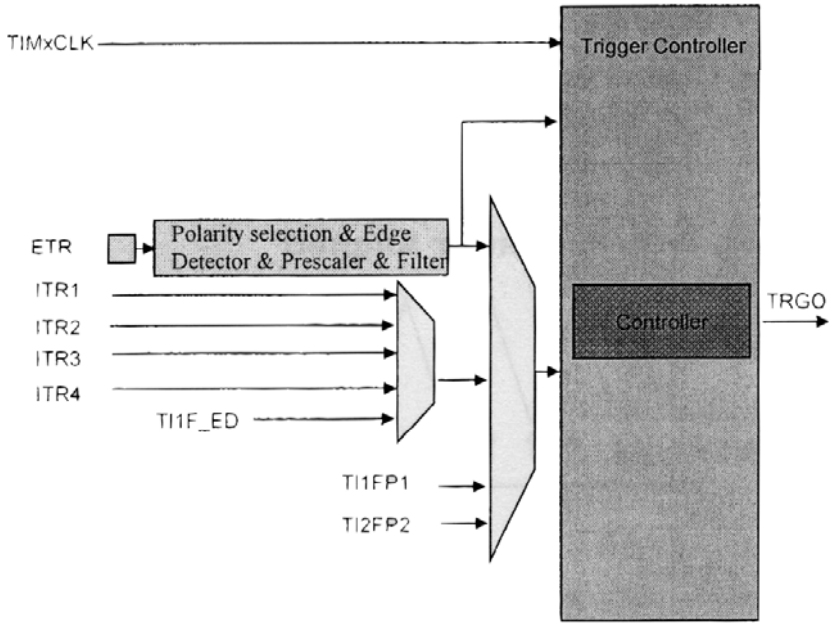
Auto-reload register buffer



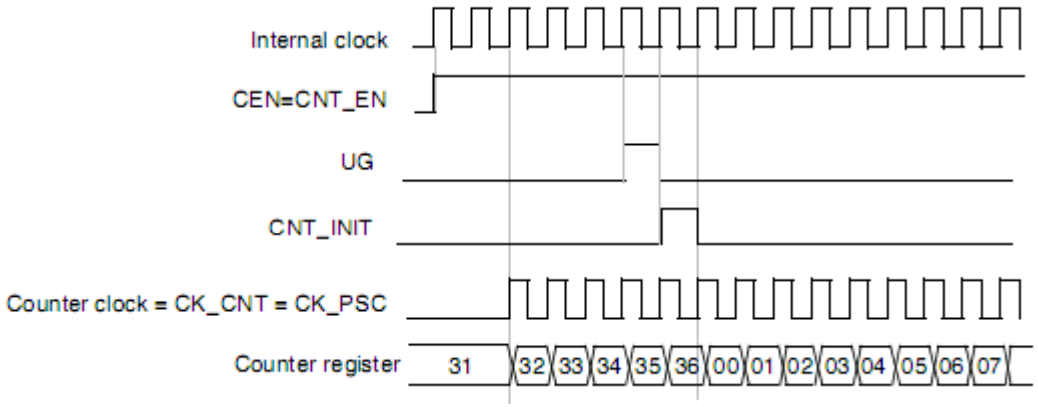


Clock selection

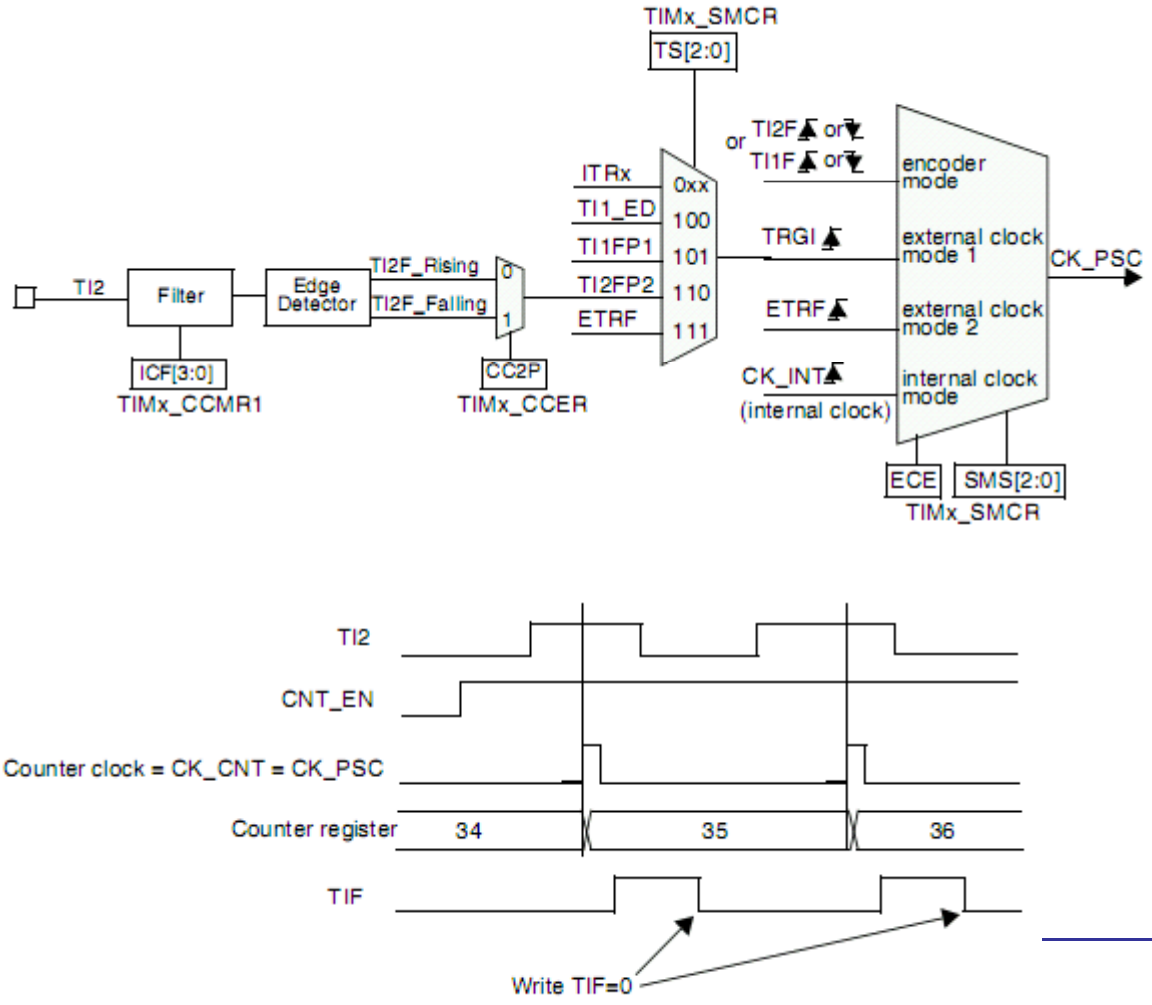
- Clock can be selected out of 8 sources
  - ✓ Internal clock TIMxCLK Provided bt the RCC
  - ✓ External pin ETR
    - Enable/Disable bit
    - Programmable polarity
    - 4 Bits External Trigger filiter
    - External Trigger Presaler
      - Presaler off
      - Division by 2
      - Division by 4
      - Division by 8
  - ✓ Internal Trigger input 1 to 4:
    - ITR1, 2, 3, 4
    - Using one timer as prescaler for an
  - ✓ External Capture Compare pins
    - TI1F_ED
    - TI1FP1, TI2FP2



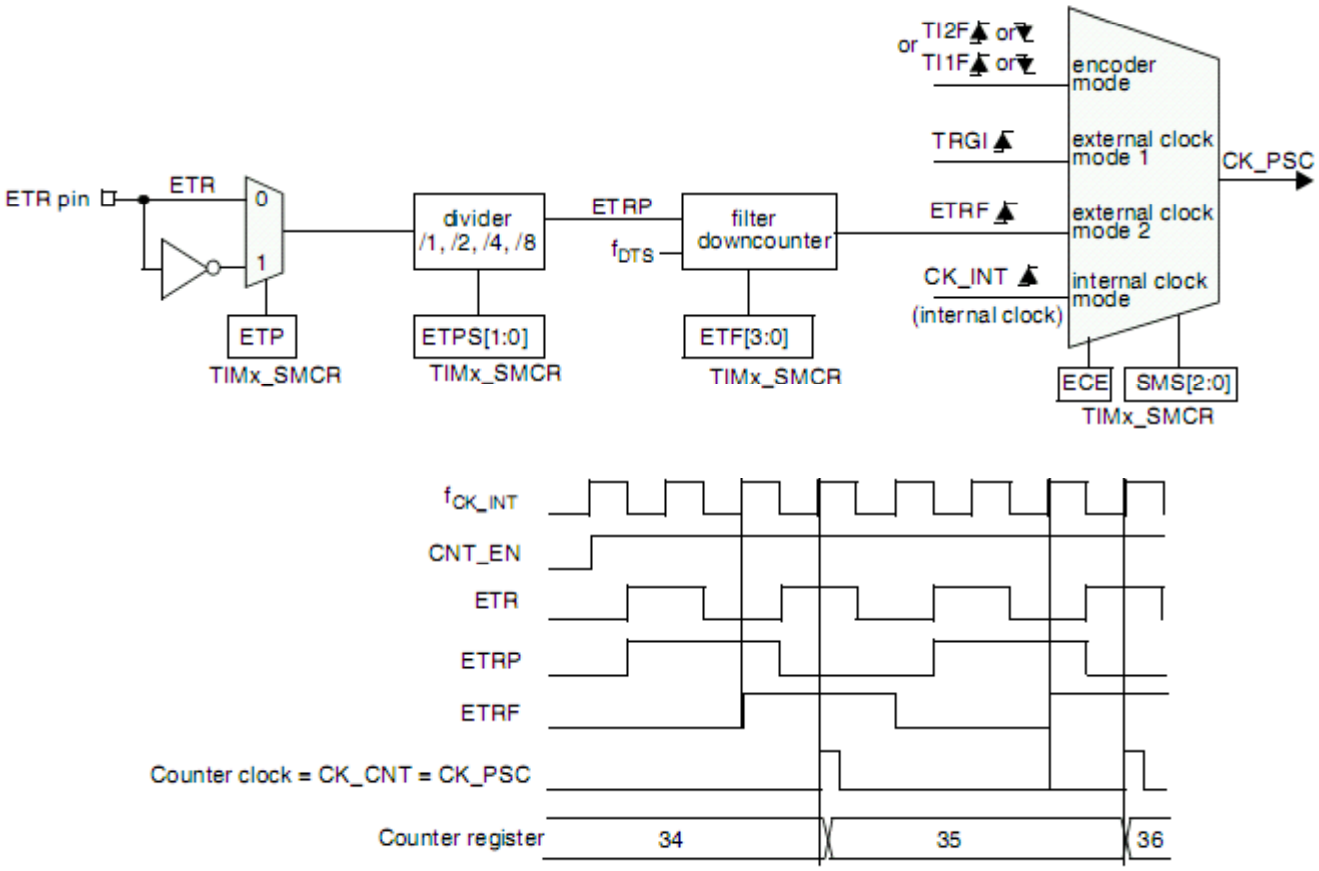
Internal clock source



External clock source mode 1

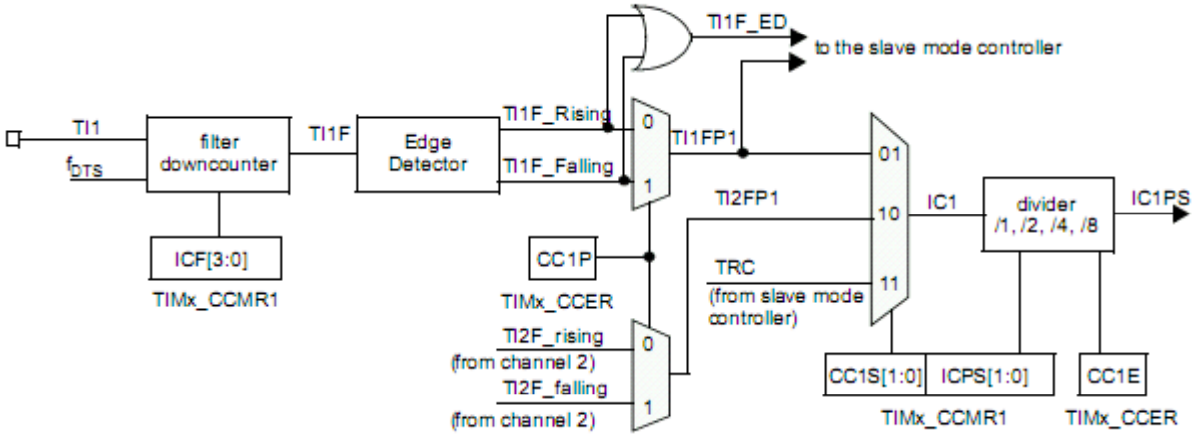


External clock source mode 2



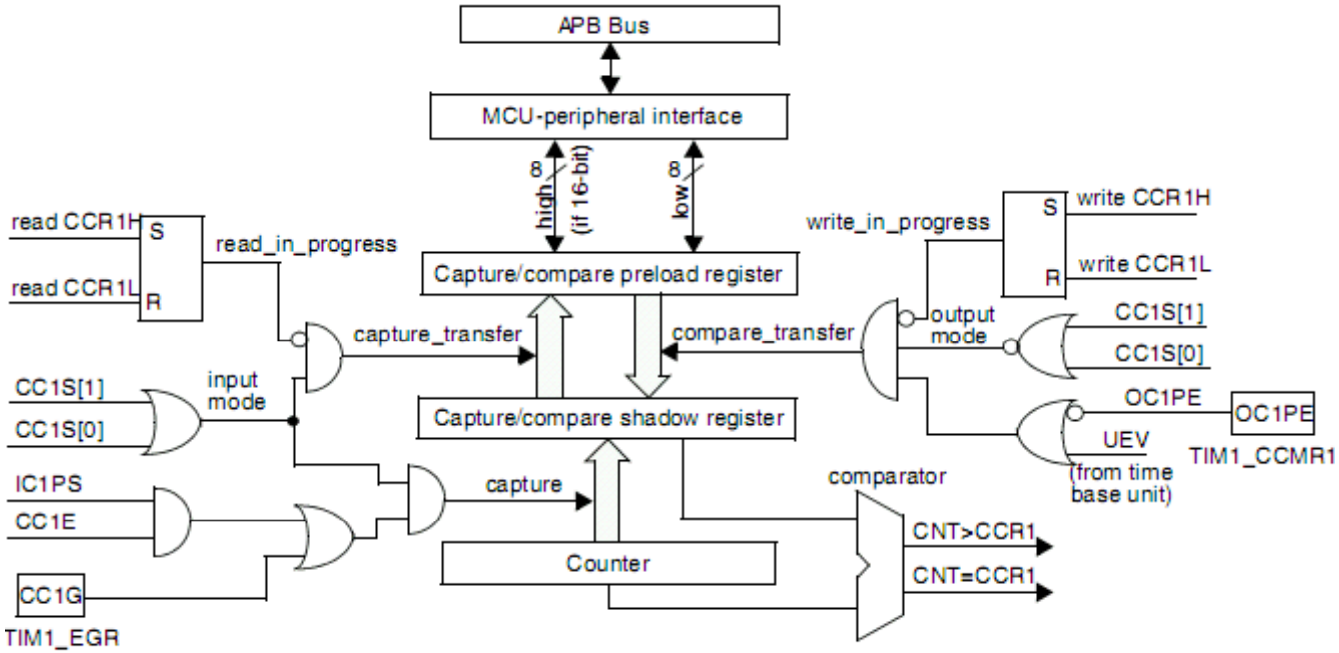
Capture/compare channels

- Capture/compare channel (example: channel 1 input stage)



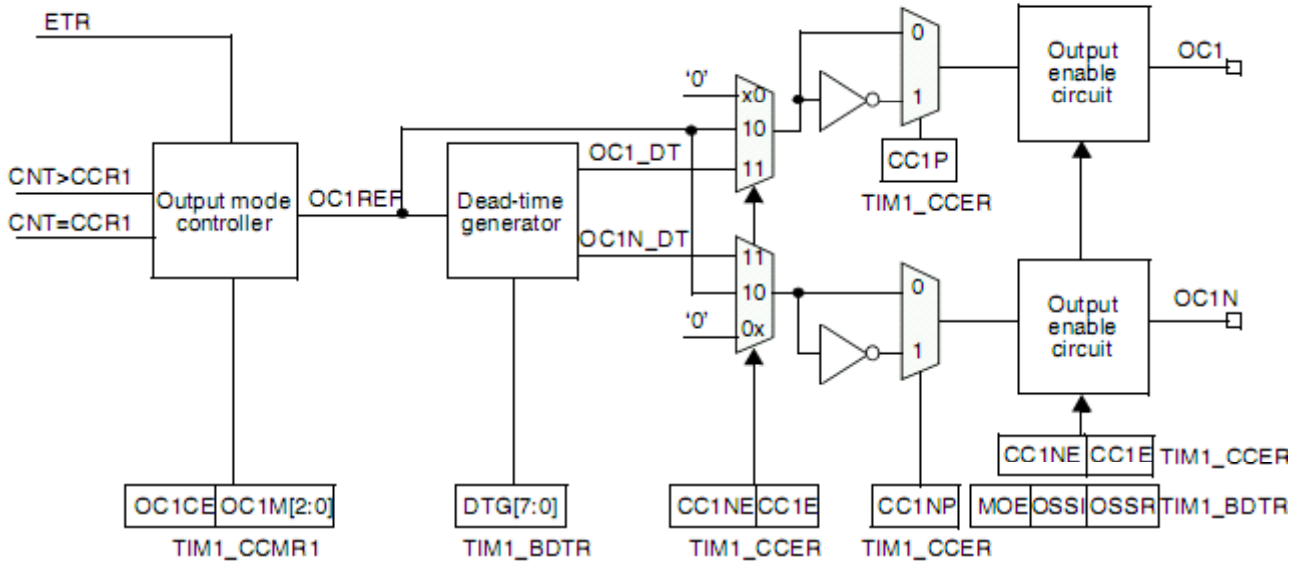
Capture/compare channels(Cont.)

- Capture/compare channel 1 main circuit



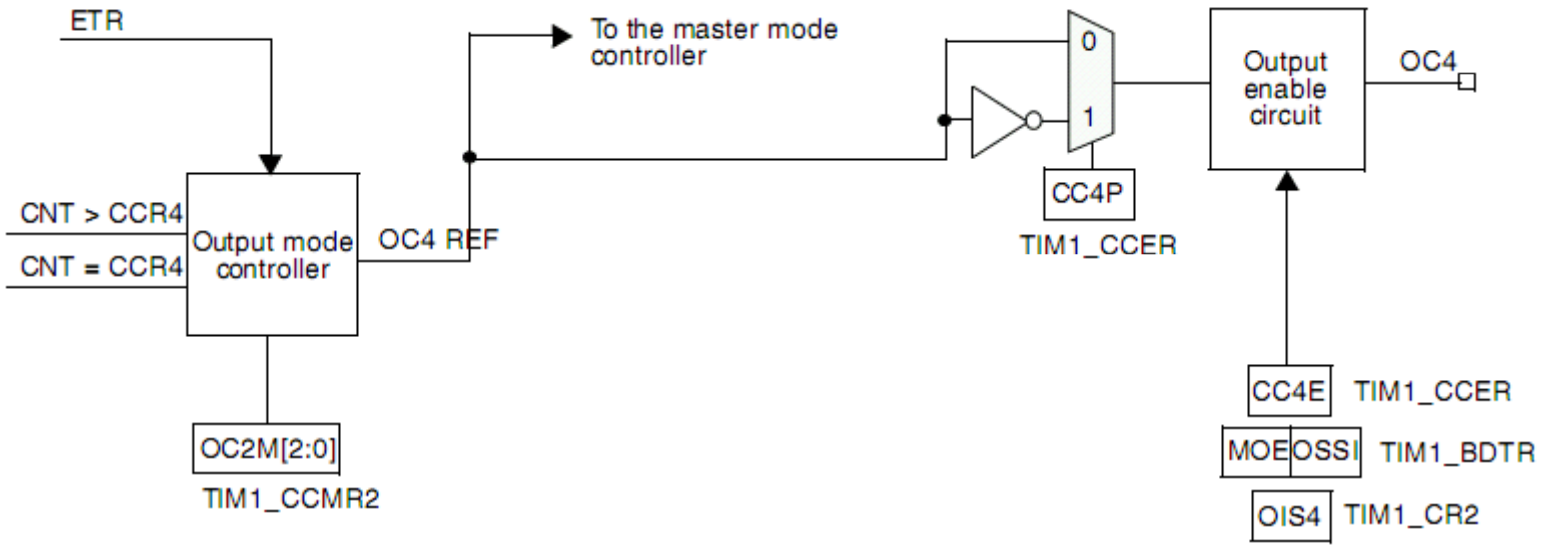
Capture/compare channels(Cont.)

- *Output stage of capture/compare channel 1~3*

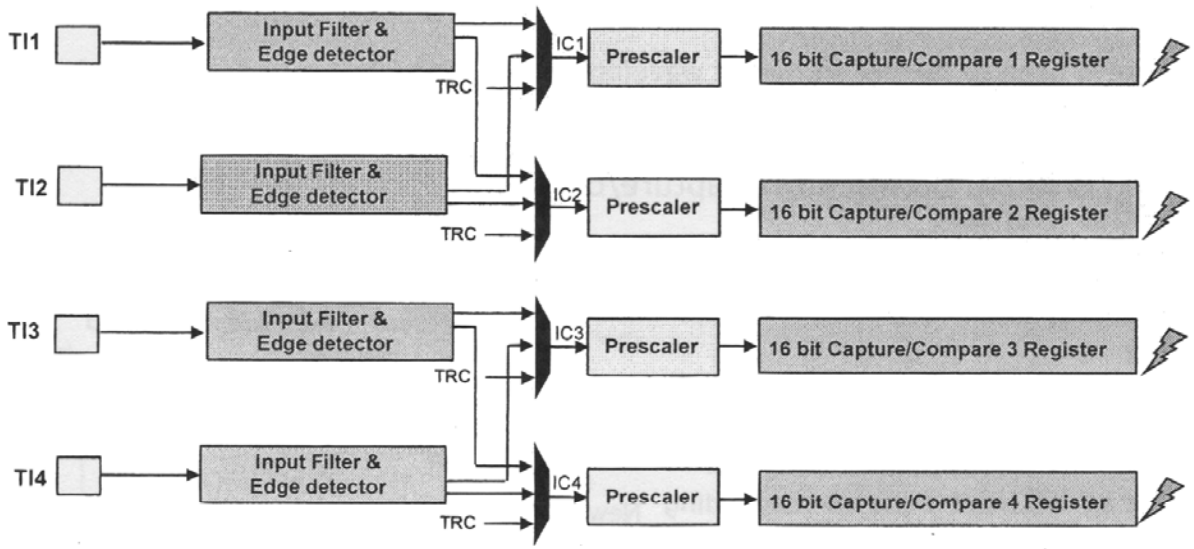


Capture/compare channels(Cont.)

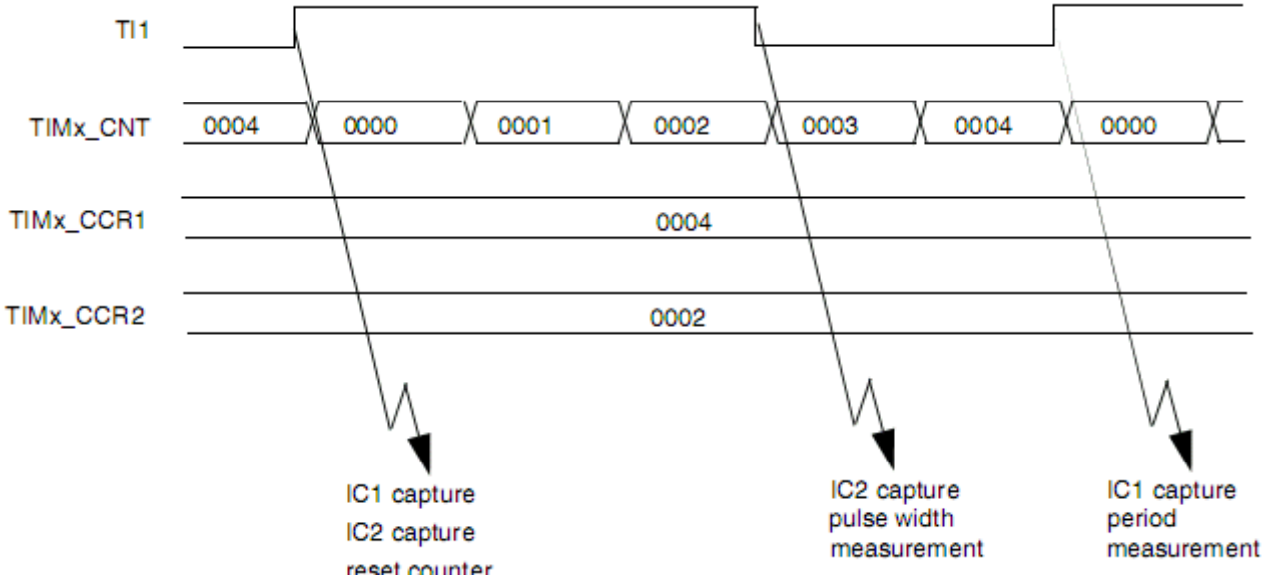
- *Output stage of capture/compare channel 4*



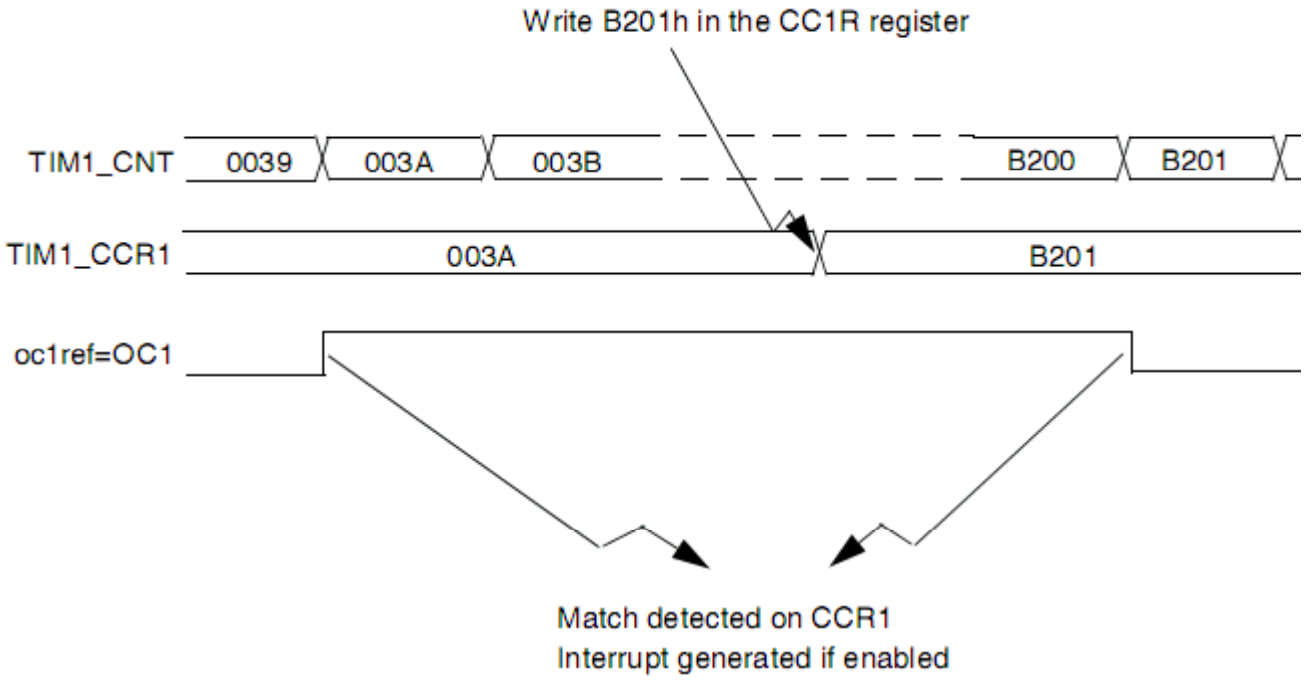
Input capture mode



PWM input mode

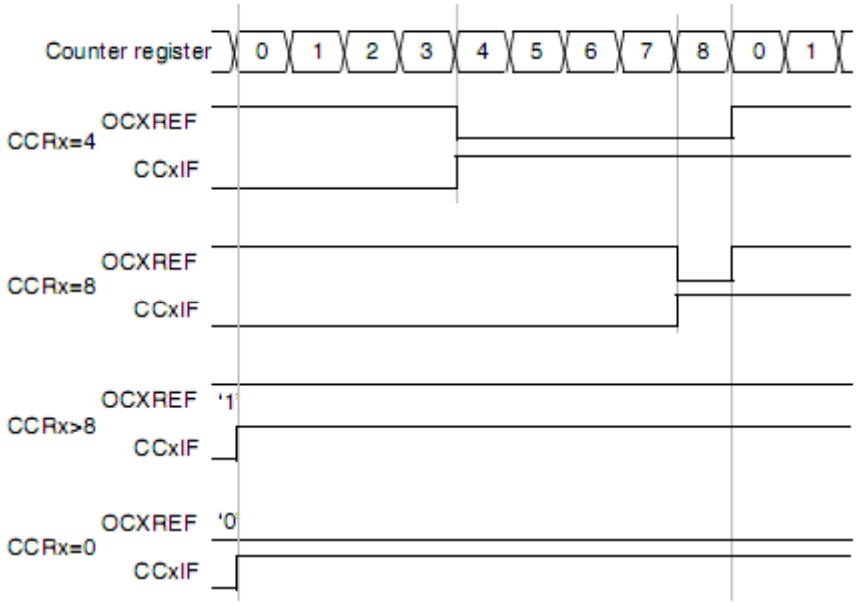


Output compare mode



PWM mode

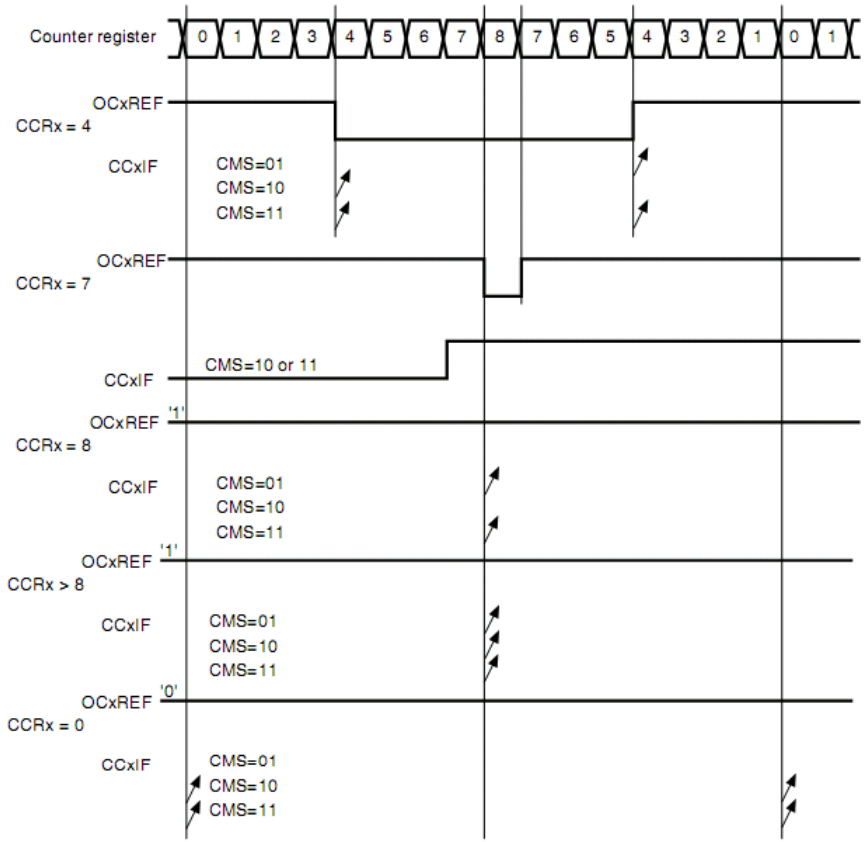
- PWM edge-aligned mode



Edge-aligned PWM waveforms (ARR=8)

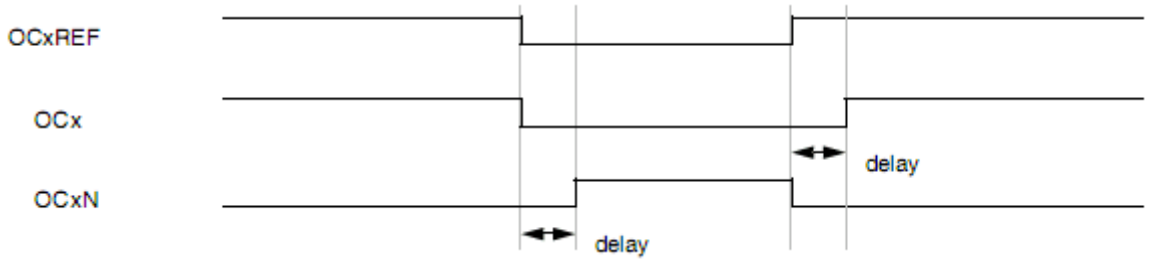
PWM mode(cont.)

- PWM center-aligned mode

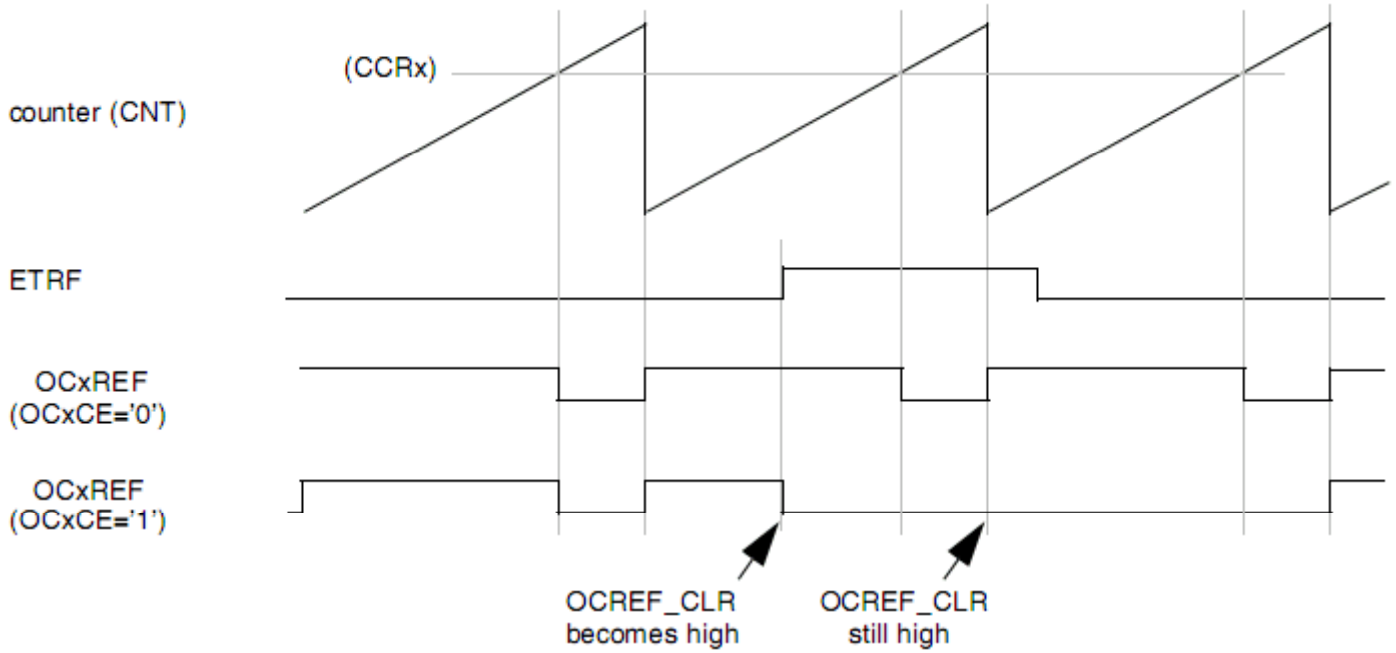


Center-aligned PWM waveforms (ARR=8)

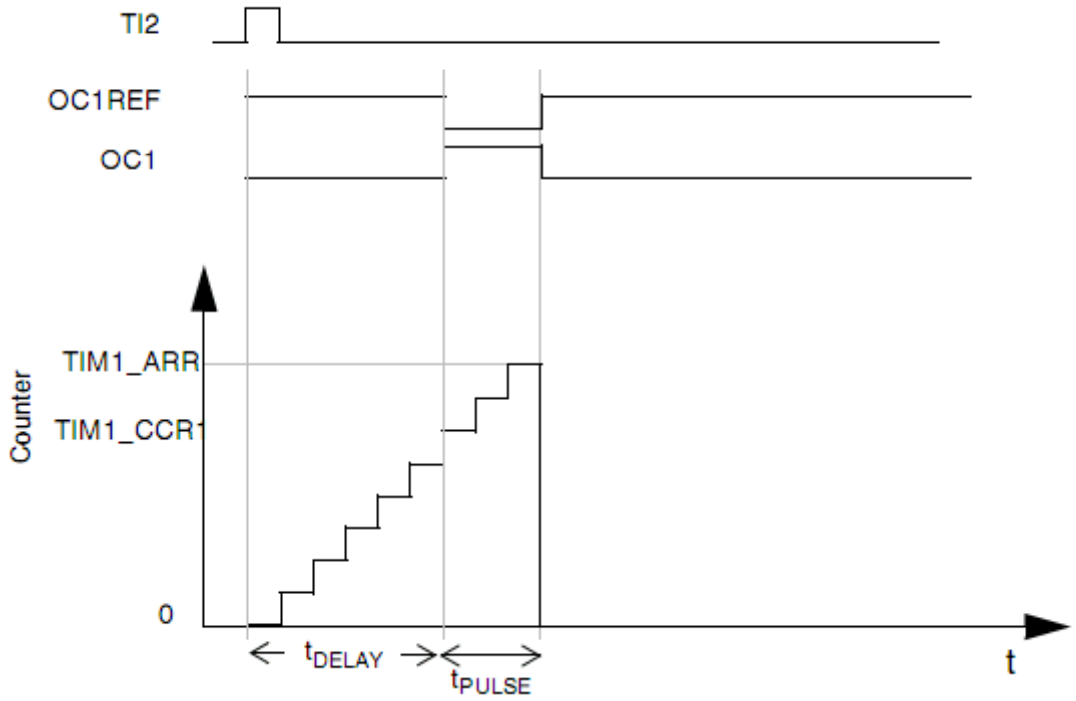
Complementary outputs and dead-time insertion



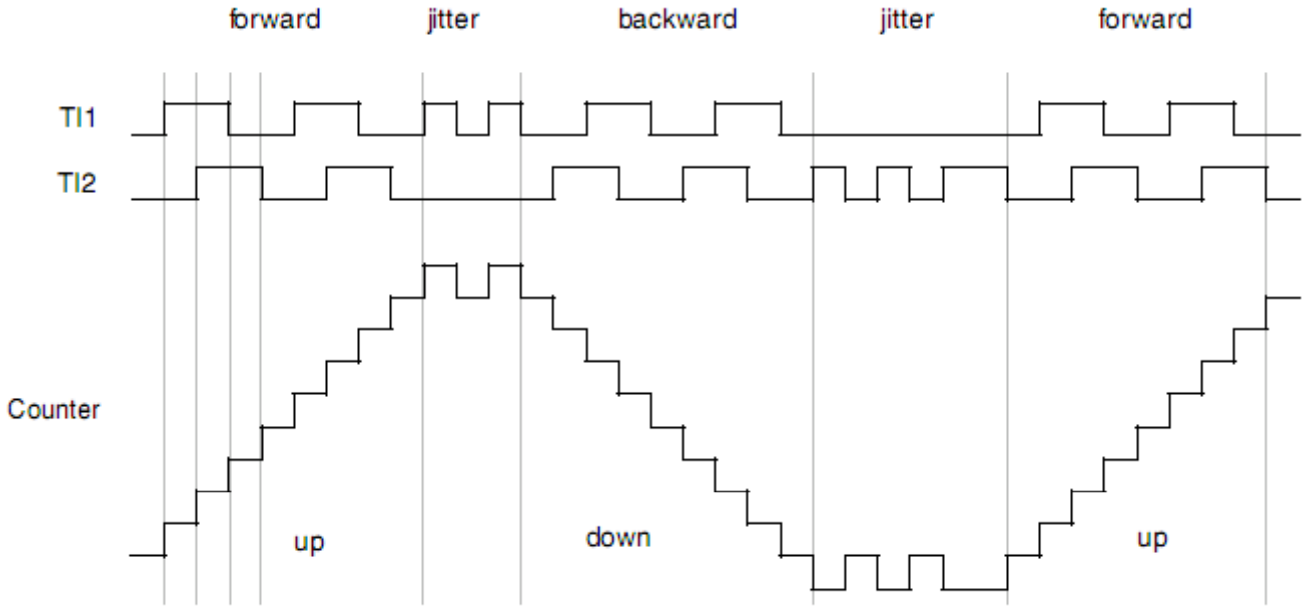
Clearing the OCxREF signal on an external event



One-pulse mode

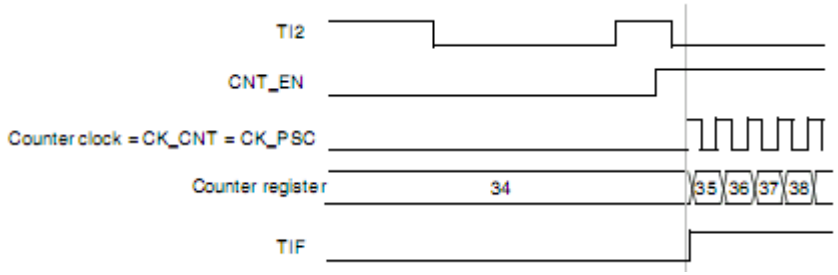


Encoder interface mode

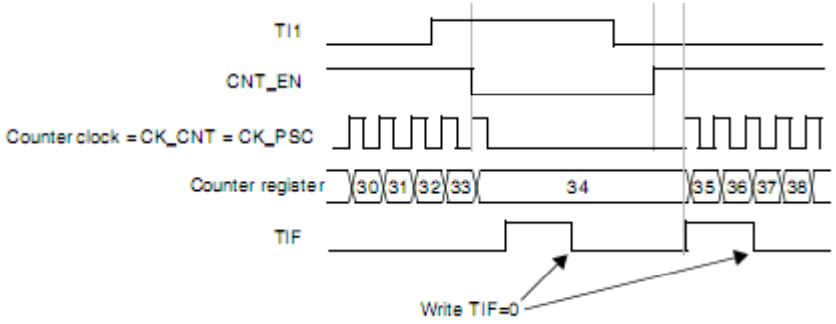


Timers and external trigger synchronization

- Slave mode: Trigger mode

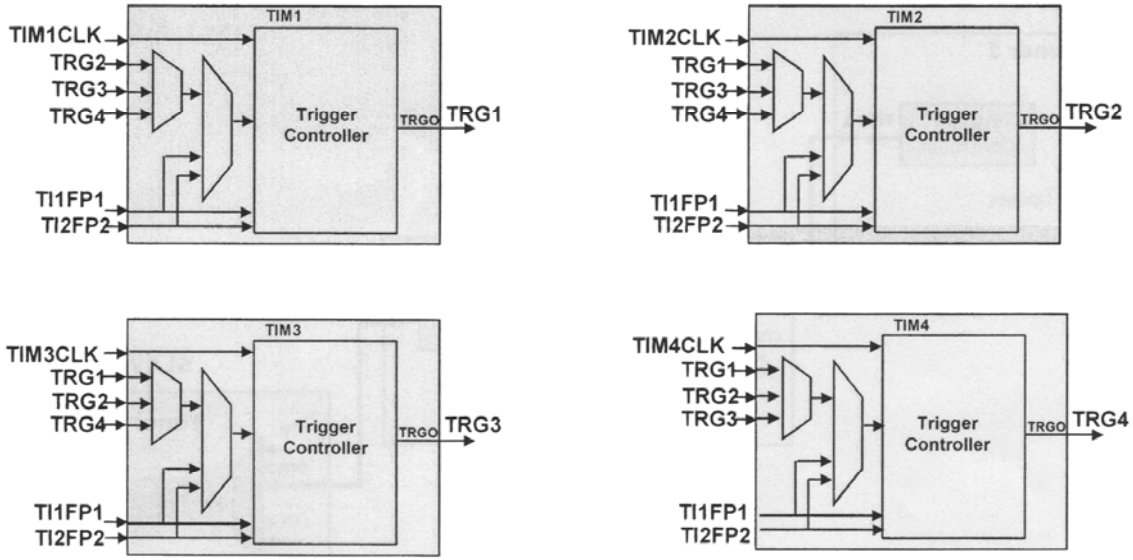


- Slave mode: Gated mode



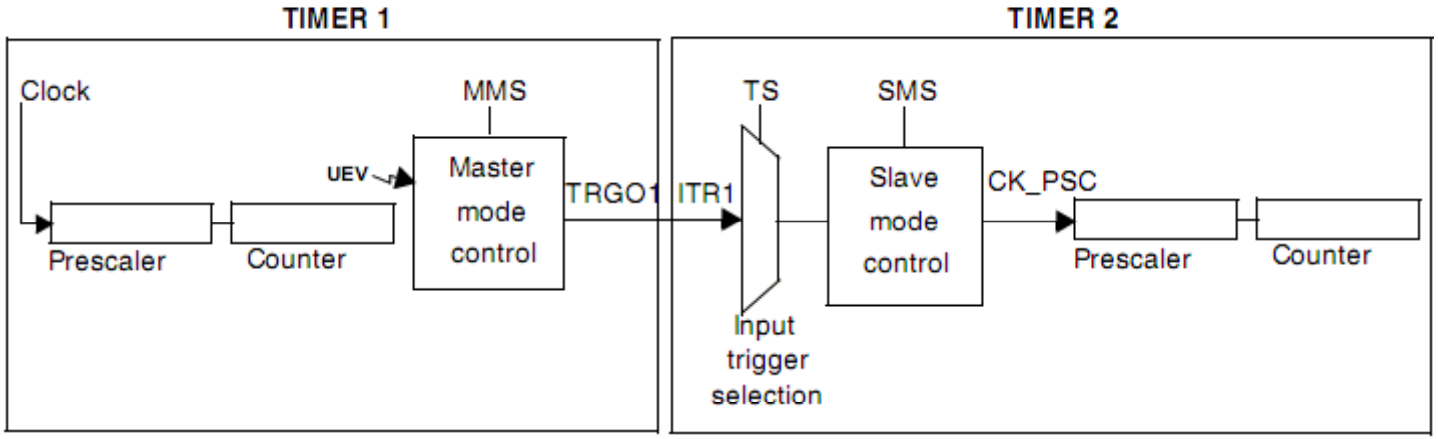
Timer synchronization

- The four Timers are link together for timers synchronization or chaining
- Using one timer as prescaler for the another



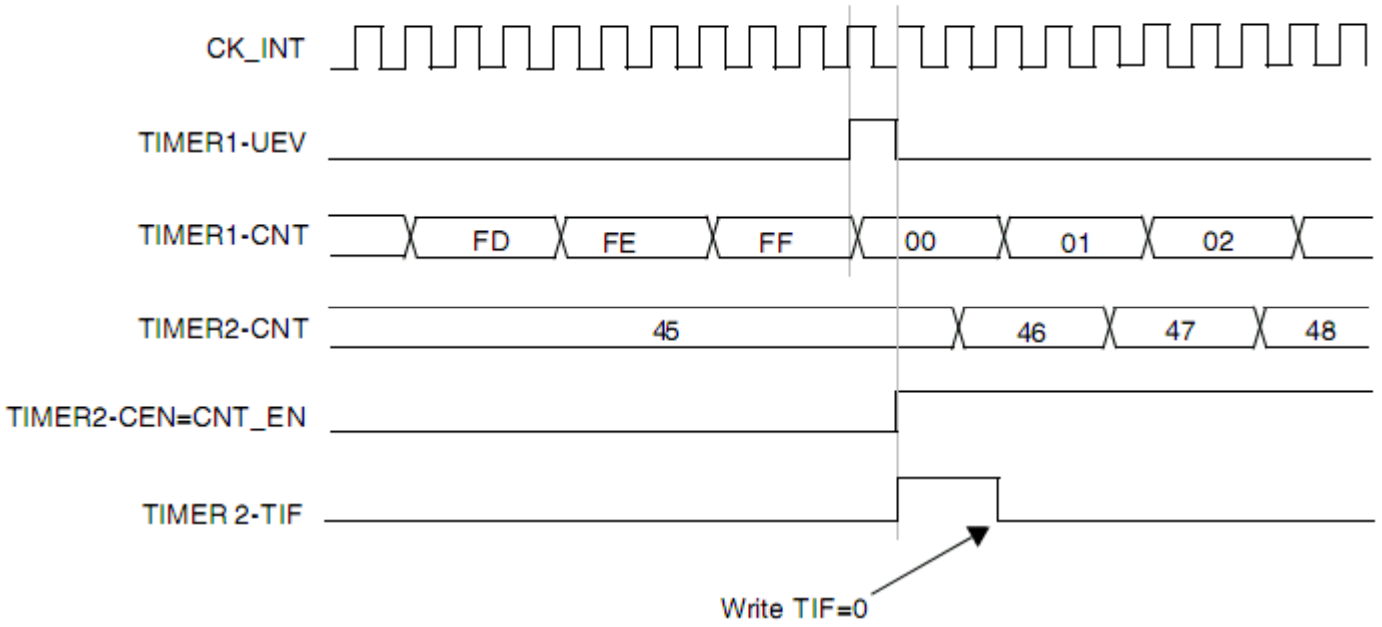
Timer synchronization(cont.)

- Using one timer as prescaler for the another



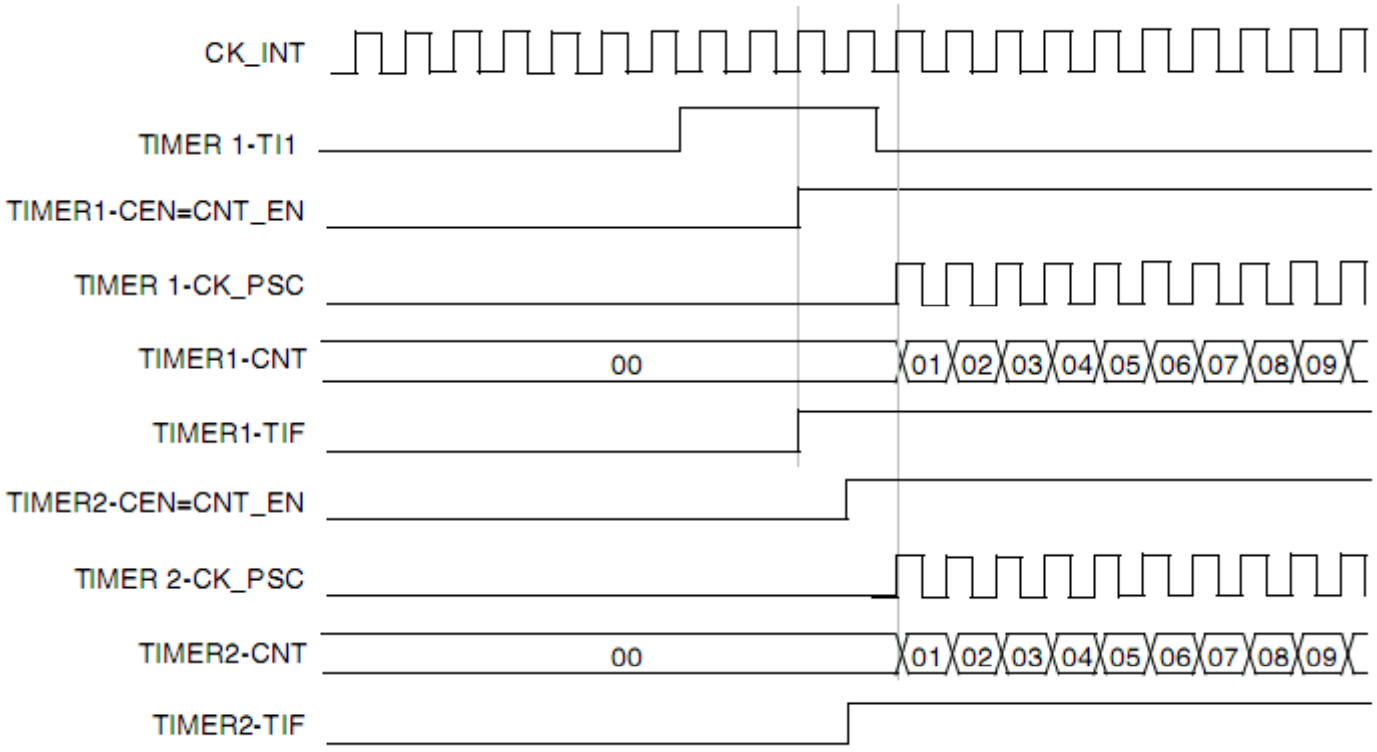
Timer synchronization(cont.)

- Using one timer to start another timer



Timer synchronization(cont.)

- Starting 2 timers synchronously in response to an external trigger



Development Flow

Embedded Software Side

Connect the EVB and the IOB

Programming

Bootup STM32F10x

- **RCC Configure**
- **GPIO Configure**
- **TIMsConfigure**
- **NVIC Configure**

```
int main(void)
{

  #ifdef DEBUG
    debug();
  #endif

  /* System clocks configuration ----------------------*/
  RCC_Configuration();

  /* NVIC configuration -------------------------------*/
  NVIC_Configuration();

 /* Configure TIMs -----------------------------------*/
  TIM_Configuration();

  /* GPIO configuration -------------------------------*/
  GPIO_Configuration();

   while(1)
   {
   }
}
```

Configure RCC

RCC FwLib Functions List

Function name	Description
RCC_DeInit	Resets the RCC clock configuration to the default reset state.
RCC_HSEConfig	Configures the External High Speed oscillator (HSE).
RCC_WaitForHSEStartUp	Waits for HSE start-up.
RCC_HCLKConfig	Configures the AHB clock (HCLK).
RCC_PCLK1Config	Configures the Low Speed APB clock (PCLK1).
RCC_PCLK2Config	Configures the High Speed APB clock (PCLK2).
RCC_PLLConfig	Configures the PLL clock source and multiplication factor.
RCC_PLLCmd	Enables or disables the PLL.
RCC_SYSCLKConfig	Configures the system clock (SYSCLK).
RCC_APB2PeriphClockCmd	Enables or disables the High Speed APB (APB2) peripheral clock.

```
void RCC_Configuration(void)
{
  /* RCC system reset(for debug purpose) */
  RCC_DeInit();
  /* Enable HSE */
  RCC_HSEConfig(RCC_HSE_ON);
  /* Wait till HSE is ready */
  HSEStartUpStatus = RCC_WaitForHSEStartUp();
  if(HSEStartUpStatus == SUCCESS) {
    /* Enable Prefetch Buffer */
    FLASH_PrefetchBufferCmd(FLASH_PrefetchBuffer_Enable);
    /* Flash 2 wait state */
    FLASH_SetLatency(FLASH_Latency_2);
    /* HCLK = SYSCLK */
    RCC_HCLKConfig(RCC_SYSCLK_Div1);
    /* PCLK2 = HCLK */
    RCC_PCLK2Config(RCC_HCLK_Div1);
    /* PCLK1 = HCLK/2 */
    RCC_PCLK1Config(RCC_HCLK_Div2);
    /* ADCCLK = PCLK2/4 */
    RCC_ADCCLKConfig(RCC_PCLK2_Div4);
    /* PLLCLK = 8MHz * 7 = 56 MHz */
    RCC_PLLConfig(RCC_PLLSource_HSE_Div1, RCC_PLLMul_7);
    /* Enable PLL */
    RCC_PLLCmd(ENABLE);
    /* Wait till PLL is ready */
    while(RCC_GetFlagStatus(RCC_FLAG_PLLRDY) == RESET) { }
    /* Select PLL as system clock source */
    RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);
    /* Wait till PLL is used as system clock source */
    while(RCC_GetSYSCLKSource() != 0x08) {  }
  }
 /* Enable peripheral clocks ------------------------------*/
 /* Enable GPIO_A clock */
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIO_LED, ENABLE);

  /* Enable TIM2, TIM3 and TIM4 and TIM5 clocks */
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2 | RCC_APB1Periph_TIM3 |
                         RCC_APB1Periph_TIM4 | RCC_APB1Periph_TIM5, ENABLE);
}
```

Configure GPIO

GPIO FwLib Functions List

Function name	Description
GPIO_DeInit	Resets the GPIOx peripheral registers to their default reset values.
GPIO_AFIODeInit	Resets the Alternate Functions (remap, event control and EXTI configuration) registers to their default reset values.
GPIO_Init	Initializes the GPIOx peripheral according to the specified parameters in the GPIO_InitStruct.
GPIO_StructInit	Fills each GPIO_InitStruct member with its default value.
GPIO_ReadInputDataBit	Reads the specified input port pin
GPIO_ReadInputData	Reads the specified GPIO input data port
GPIO_ReadOutputDataBit	Reads the specified output data port bit
GPIO_ReadOutputData	Reads the specified GPIO output data port
GPIO_SetBits	Sets the selected data port bits
GPIO_ResetBits	Clears the selected data port bits
GPIO_WriteBit	Sets or clears the selected data port bit
GPIO_Write	Writes data to the specified GPIO data port
GPIO_PinLockConfig	Locks GPIO Pins configuration registers
GPIO_EventOutputConfig	Selects the GPIO pin used as Event output.
GPIO_EventOutputCmd	Enables or disables the Event Output.
GPIO_PinRemapConfig	Changes the mapping of the specified pin.
GPIO_EXTILineConfig	Selects the GPIO pin used as EXTI Line.

```
void GPIO_Configuration(void)
{
  GPIO_InitTypeDef GPIO_InitStructure;


 /* Configure GPIO_A pin 5, 6, 7, 8, as output push-pull */
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6 | GPIO_Pin_7 | GPIO_Pin_8;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_Init(GPIO_A, &GPIO_InitStructure);}
}
```

Configure TIMs

```
void TIM_Configuration(void)
{
  TIM_TimeBaseInitTypeDef  TIM_TimeBaseStructure;
  TIM_OCInitTypeDef  TIM_OCInitStructure;

  /* TIM2 configuration */
  TIM_TimeBaseStructure.TIM_Period = 0x4AF;
  TIM_TimeBaseStructure.TIM_Prescaler = 0xEA5F;
  TIM_TimeBaseStructure.TIM_ClockDivision = 0x0;
  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
  TIM_TimeBaseStructure.TIM_RepetitionCounter = 0x0000;
  TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);

  TIM_OCStructInit(&TIM_OCInitStructure);
  /* Output Compare Timing Mode configuration: Channel1 */
  TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_Timing;
  TIM_OCInitStructure.TIM_Pulse = 0x0;
  TIM_OC1Init(TIM2, &TIM_OCInitStructure);

  /* TIM3 configuration */
  TIM_TimeBaseStructure.TIM_Period = 0x95F;

  TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);
  /* Output Compare Timing Mode configuration: Channel1 */
  TIM_OC1Init(TIM3, &TIM_OCInitStructure);
  ……

  /* TIM2 enable counter */
  TIM_Cmd(TIM2, ENABLE);
   ……

  /* Immediate load of TIM2 Precaler value */
  TIM_PrescalerConfig(TIM2, 0xEA5F, TIM_PSCReloadMode_Immediate);
   ……

  /* Clear TIM2 update pending flag */
  TIM_ClearFlag(TIM2, TIM_FLAG_Update);
   ……

  /* Enable TIM2 Update interrupt */
  TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE);
}
```

Configure NVIC

```
 /* Configure one bit for preemption priority */
 NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);

 /* Enable the TIM2 Interrupt */
 NVIC_InitStructure.NVIC_IRQChannel = TIM2_IRQChannel;
 NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
 NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
 NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
 NVIC_Init(&NVIC_InitStructure);

 /* Enable the TIM3 Interrupt */
 NVIC_InitStructure.NVIC_IRQChannel = TIM3_IRQChannel;
 NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 1;
 NVIC_Init(&NVIC_InitStructure);

 /* Enable the TIM4 Interrupt */
 NVIC_InitStructure.NVIC_IRQChannel = TIM4_IRQChannel;
 NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 2;
 NVIC_Init(&NVIC_InitStructure);

/* Enable the TIM5 Interrupt */
 NVIC_InitStructure.NVIC_IRQChannel = TIM5_IRQChannel;
 NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 3;
 NVIC_Init(&NVIC_InitStructure);
```

IRQ Service

```
void TIM2_IRQHandler(void)
{
  /* Clear TIM2 update interrupt */
  TIM_ClearITPendingBit(TIM2, TIM_IT_Update);

  /* Toggle GPIO_LED pin 5 */
  GPIO_WriteBit(GPIO_A, GPIO_Pin_5, (BitAction)(1 - GPIO_ReadOutputDataBit(GPIO_A, GPIO_Pin_5)));
}
```

硬體電路配置

- LED電路配置
- 硬體接線示意圖

實驗步驟

- 軟體設置
- 原始碼檔案瀏覽
- 編譯燒錄程式並觀察結果

檔案目錄結構

<目錄>/檔案	說明
	<..\Timer_Counter\E1>
<project>	單元實驗Project目錄
<source>	程式碼目錄
<include>	引入檔目錄
<library>	函式庫目錄
<image>	燒錄配置檔目錄
	<..\\Timer_Counter\E1\image>
Lab.dfu	燒錄配置檔
	<..\Timer_Counter\E1\source>
hw_config.c	硬體配置程式
lcd_func.c	Text LCD 控制程式
stm32f10x_it.c	中斷服務程式

編譯燒錄程式並觀察結果

- 完成系統硬體設置之工作後，依 MIAT STM32 user manual (ch3, ch4)之操作指示，將編譯後的hex檔轉換為dfu
- 透過USB 燒錄dfu檔
- 觀看4顆LED是否Delay 1s, 2s, 3s, 4s後閃爍

實作重點提示

- 觀察部份原始碼檔案，藉以了解程式架構
- 將程式改為控制1個Timer的4個channel達成與E1相同的效果
- 修改IRQ，將時間結果顯示於Text LCD，完成簡易碼錶。

實際操作

操作時間~***(90min)***

習作及參考資料

- 參考資料

  [1] MIAT_STM32_user_manual_V1.00.pdf

  [2] STM32F10xxx reference manual.pdf

  [3] STM32F103XX firmware library.pdf

  [4] Code-O-Rama Design Contest Training

Q & A