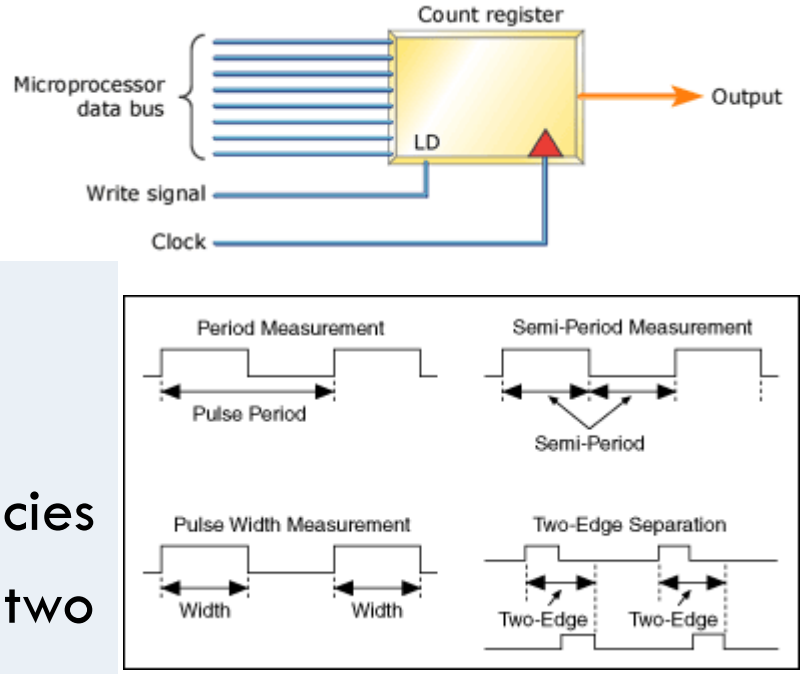
STM32F3 TIMERS

Cuauhtémoc Carbajal

22/10/2013

Introduction

- Hardware timers are used to:
  - Generate
    - signals of various frequencies
    - Generate pulse-width-modulated (PWM) outputs
    - Accurate time base
  - Trigger events at known frequencies
  - Measure elapsed time between two events
  - Count events

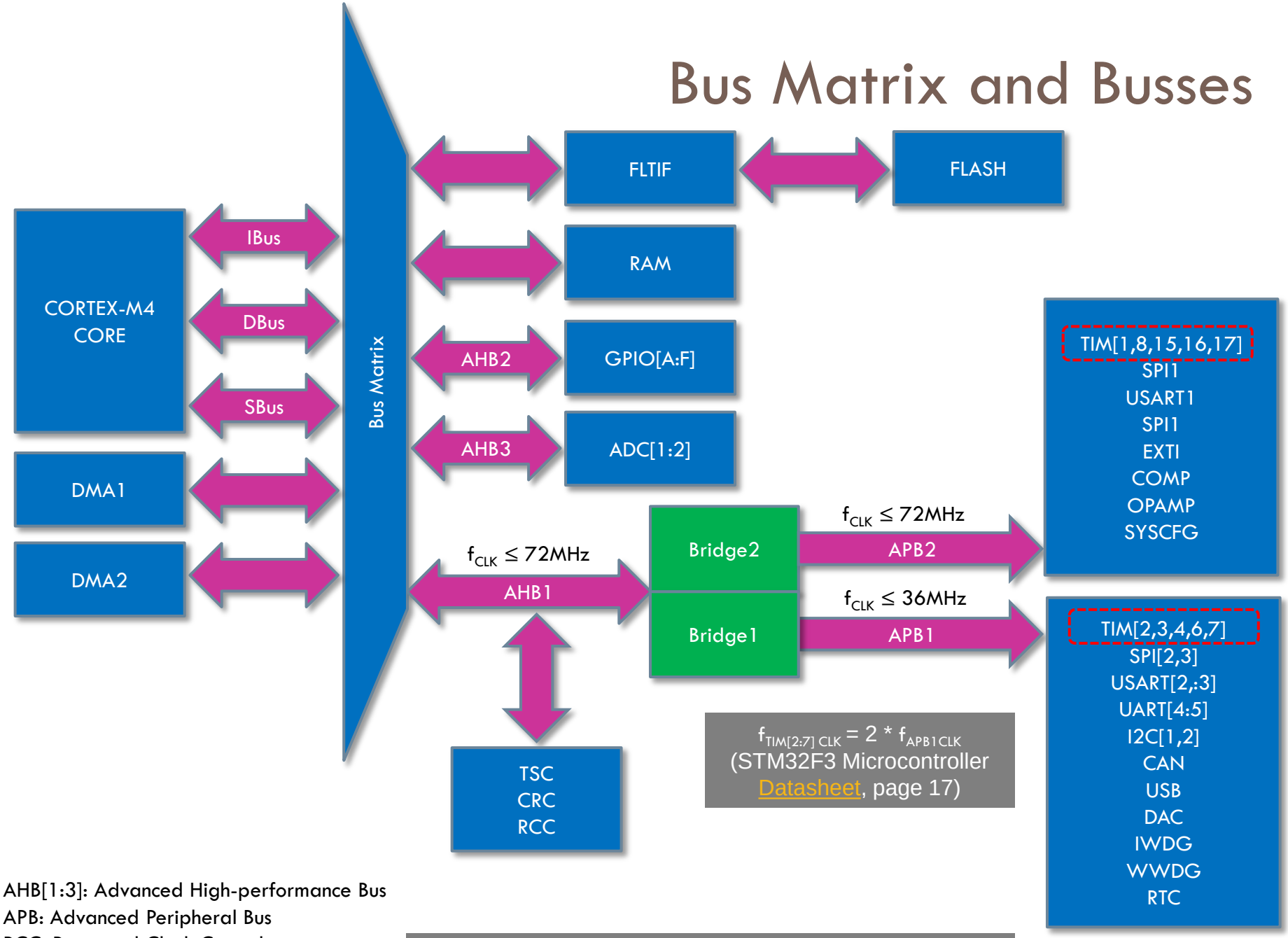


- Without accurate timing, digital control engineering is not possible – the control signals (controller action) have to happen at the exact right moment in time, e.g. timing control of an engine, etc.

STM32 Timers

- The STM32F30x has up to ten timer units
  - Timer 1 and Timer 8 are advanced timers intended for motor control.
  - Timers 2-4 and 15-17 are general purpose timer units.
  - Timers 6-7 are basic timers which are used to provide a time base to trigger the digital to analog converters.
- All of the timers have a common architecture; the advanced timer simply has additional hardware features.
- We will look at the basic timer first and then move on to the general-purpose timer.

Bus Matrix and Busses



$f_{TIM[2:7]\ CLK} = 2 * f_{APB1\ CLK}$
(STM32F3 Microcontroller Datasheet, page 17)

AHB[1:3]: Advanced High-performance Bus
APB: Advanced Peripheral Bus
RCC: Reset and Clock Control

STM32F3 Microcontroller Reference Manual, pages 41-44

Timers and IRQn

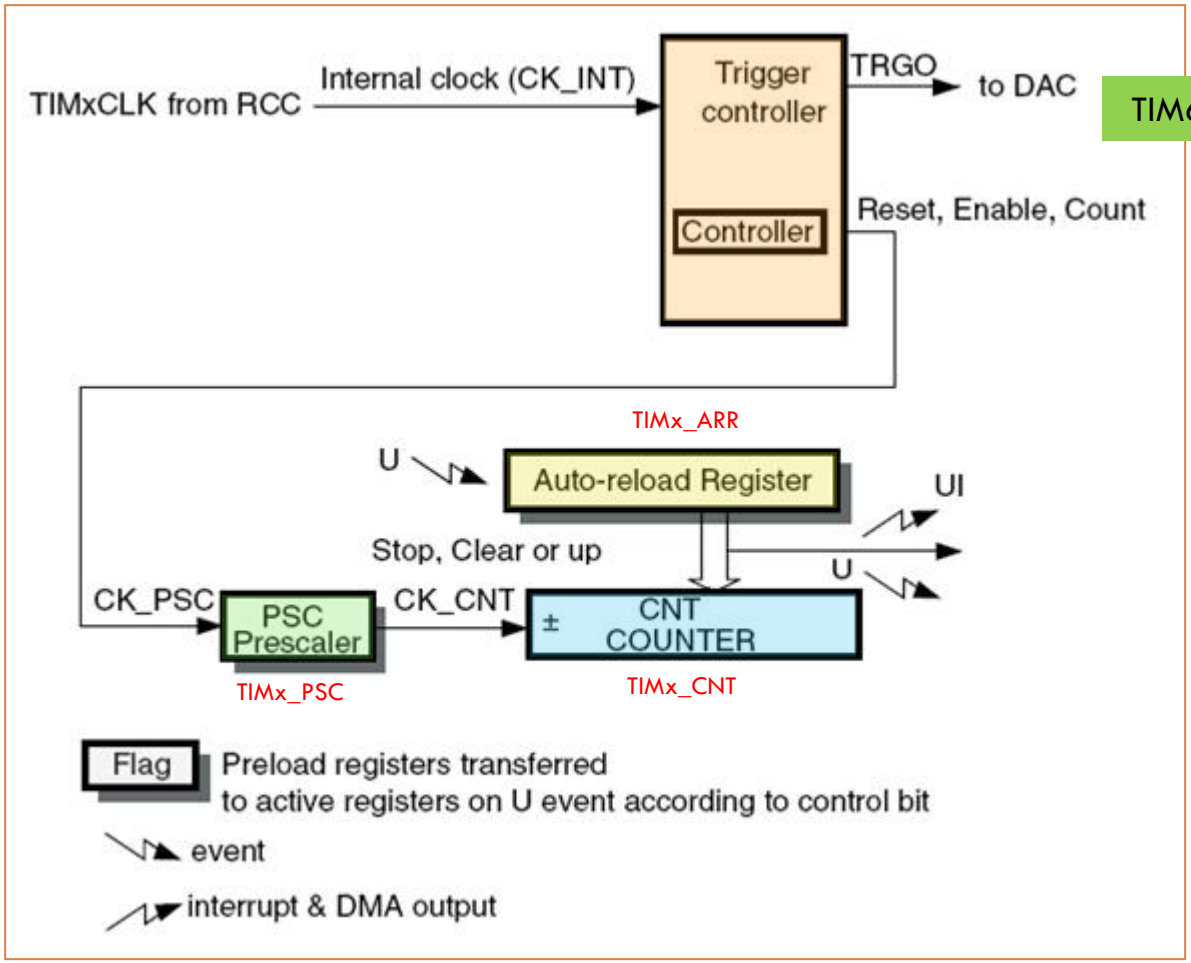
IRQn	Peripheral
24	TIM1_BRK_TIM15
25	TIM1_UP_TIM16
26	TIM1_TRG_COM_TIM17
27	TIM1_CC
28	TIM2
29	TIM3
30	TIM4
43	TIM8_BRK
44	TIM8_UP
45	TIM8_TRG_COM
46	TIM8_CC
54	TIM6_DAC
55	TIM7

STM32F3 Microcontroller Reference Manual (pages 183-186)

Timer Feature Comparison

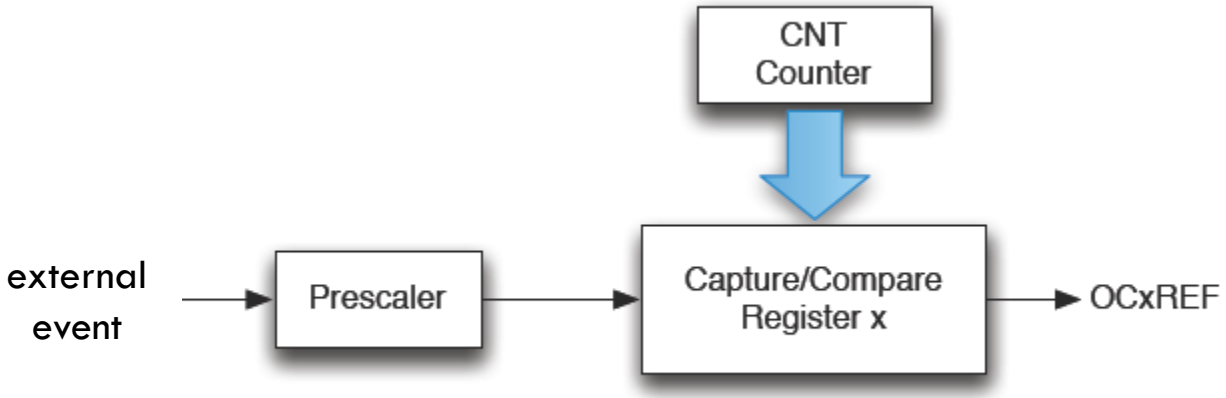
Timer	16 Bits	32 Bits	Up	Down	Up/Down	Auto-Reload	Input Capture	Output Compare	Edge-aligned PWM	Center-aligned PWM	One-pulse mode output	Complementary outputs with programmable dead-time	Synchronization circuit to control the timer with external signals and to interconnect several timers together	Repetition counter to update the timer registers only after a given number of cycles of the counter	Break inputs to put the timer's output signals in a safe user selectable configuration	Interrupt/DMA generation	Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes	Trigger input for external clock or cycle-by-cycle current management	Synchronization circuit to trigger the DAC
1,8	x		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
2		x	x	x	x	x	x	x	x	x	x		x			x	x	x	
3,4	x		x	x	x	x	x	x	x	x	x		x			x	x	x	
15	x		x			x	x	x	x		x	x	x	x	x	x			
16,17	x		x			x	x	x	x		x	x		x	x	x			
6, 7	x		x			x										x			x

Basic Timer Block Diagram



Output Compare / Input Capture

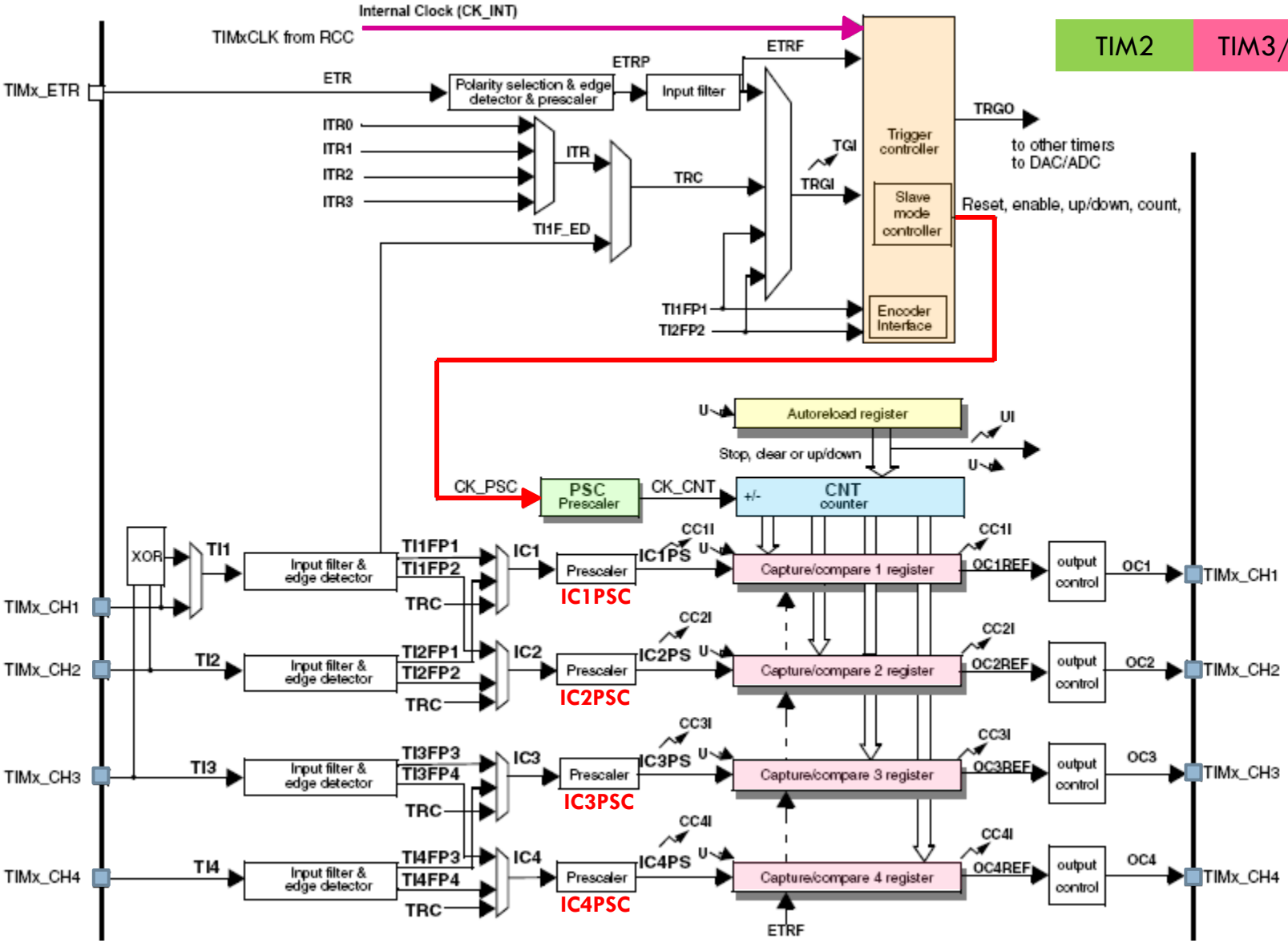
- Many timers extend the basic module with the addition of counter channels. The “x” refers to the channel.
- With this modest additional hardware, an output can be generated whenever the count register reaches a specific value **or** the counter register can be captured when a specific input event occurs (possibly a prescaled input clock).



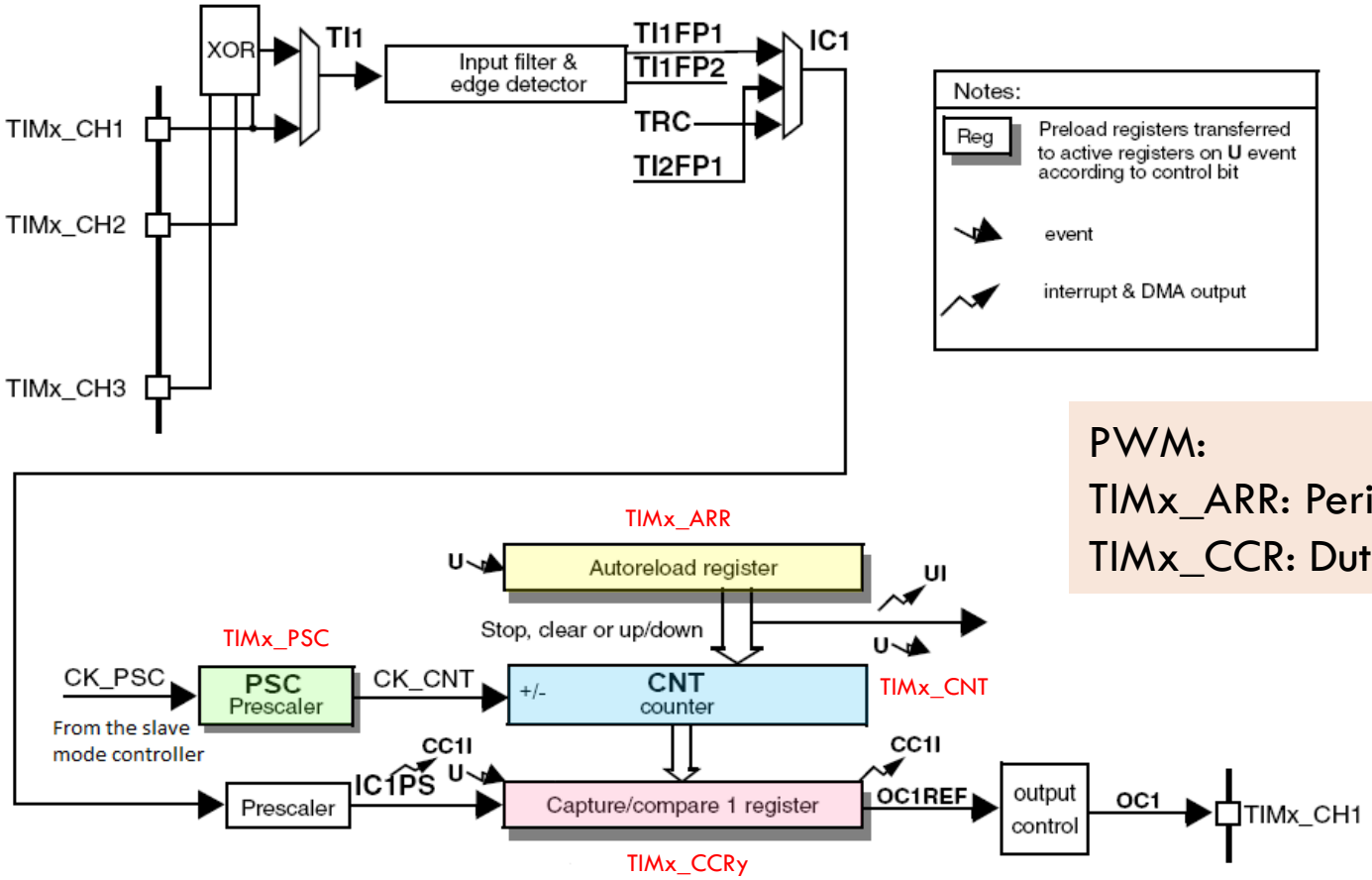
Timer Channel

General-purpose timer block diagram

TIM2 TIM3/4



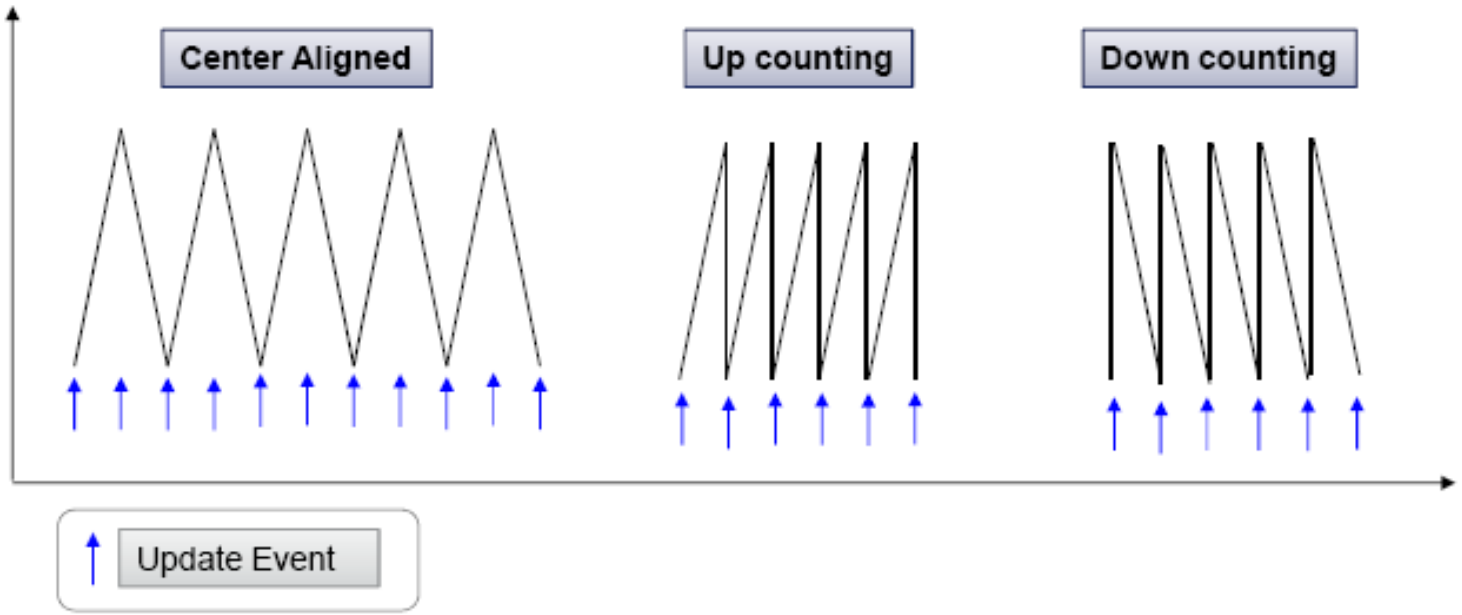
General-purpose timer block diagram (Detail)



PWM:
TIMx_ARR: Period
TIMx_CCR: Duty

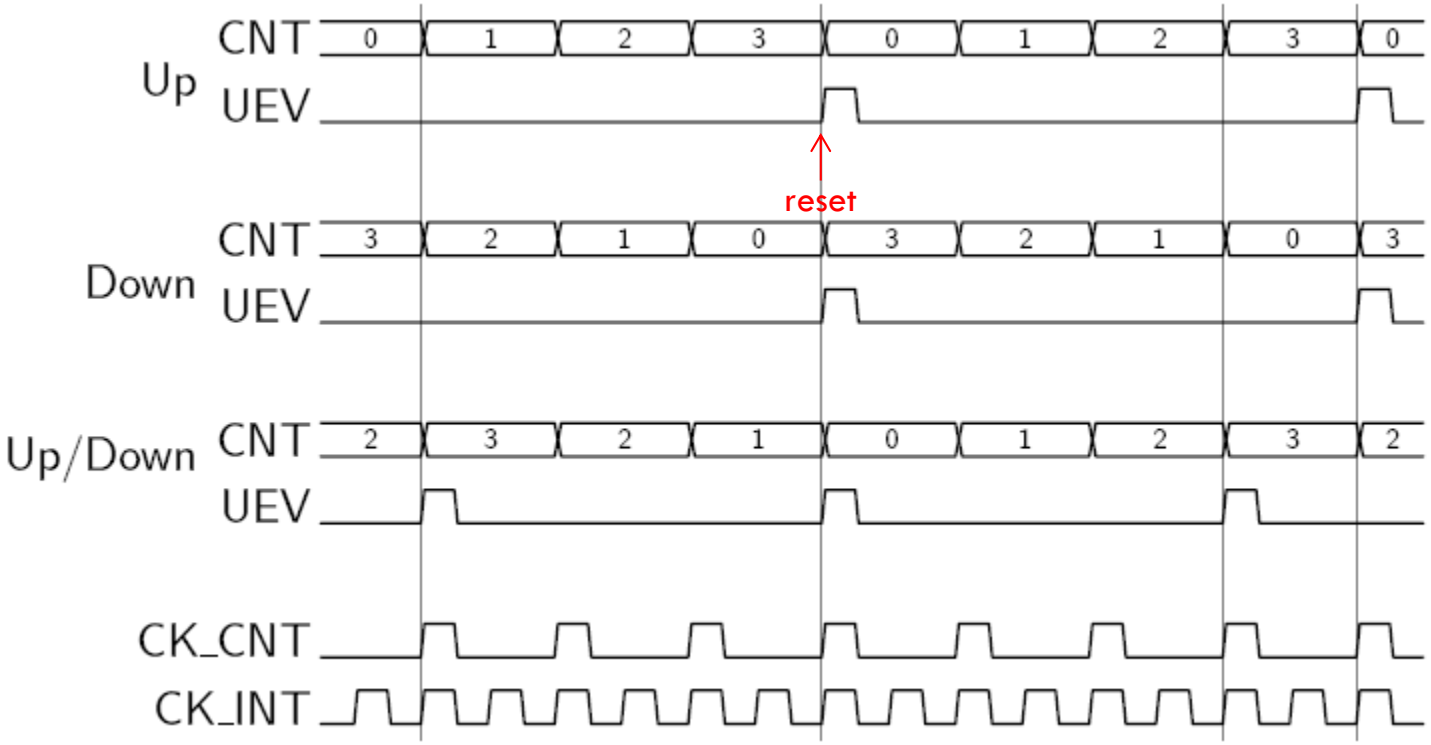
Counting Modes (1/2)

- There are three counter modes:
  - Up counting mode
  - Down counting mode
  - Center-aligned mode



TIM2/5 TIM3/4/19

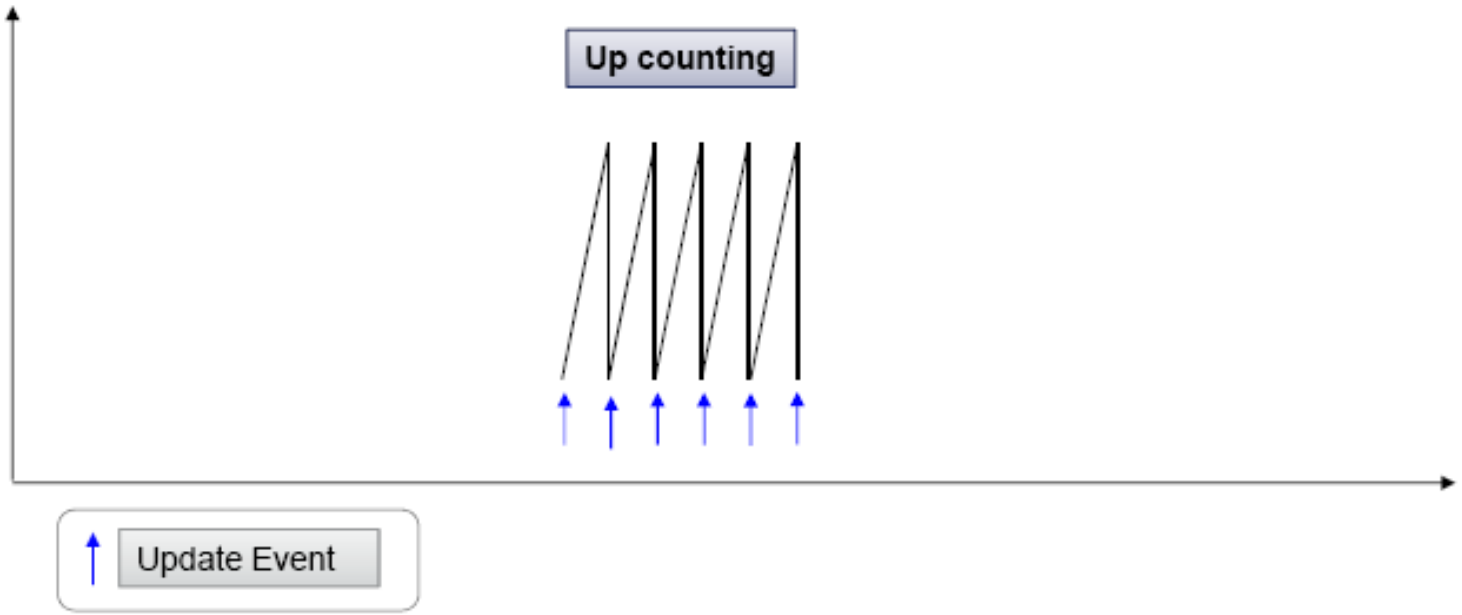
Counter Modes



Counter Modes (ARR=3, PSC=1)

Counting Modes (2/2)

- There is only one counting mode:
  - Up counting mode



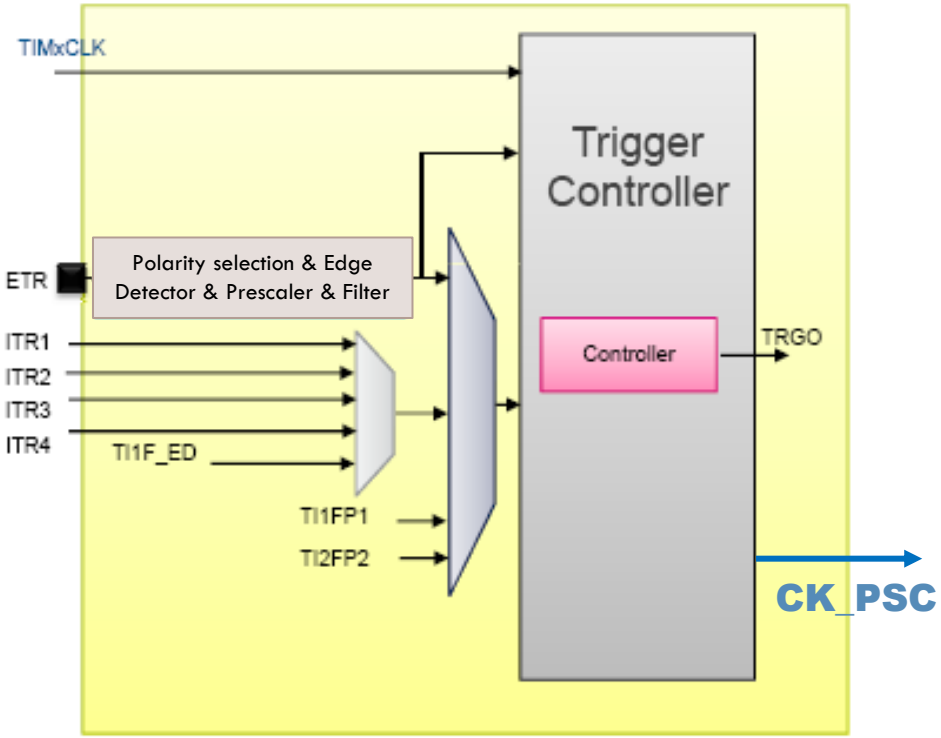
TIM12 TIM15 TIM13/14 TIM16/17 TIM6/7/18

Update Event

- The content of the preload register is transferred into the shadow register
  - depends on the *Auto-reload Preload* feature if enabled or not ARPE
    - If enabled, at each Update Event the transfer occurs
    - If not enabled, the transfer occurs Immediately
- The Update Event is generated
  - For each counter overflow/underflow
  - Through software, by setting the UG bit (Update Generation)
- The Update Event (UEV) request source can be configured to be
  - Next to counter overflow/underflow event
  - Next to Counter overflow/underflow event plus the following events
    - Setting the UG bit by software
    - Trigger active edge detection (through the slave mode controller)

Counter Clock Selection

- Clock can be selected out of 8 sources
  - Internal clock TIMxCLK provided by the RCC
  - Internal trigger input 1 to 4:
    - ITR1 / ITR2 / ITR3 / ITR4
    - Using one timer as prescaler for another timer
  - External Capture Compare pins
    - Pin 1: TI1FP1 or TI1F_ED
    - Pin 2: TI2FP2
  - External pin ETR
    - Enable/Disable bit
    - Programable polarity
    - 4 Bits External Trigger Filter
    - External Trigger Prescaler:
      - Prescaler off
      - Division by 2
      - Division by 4
      - Division by 8

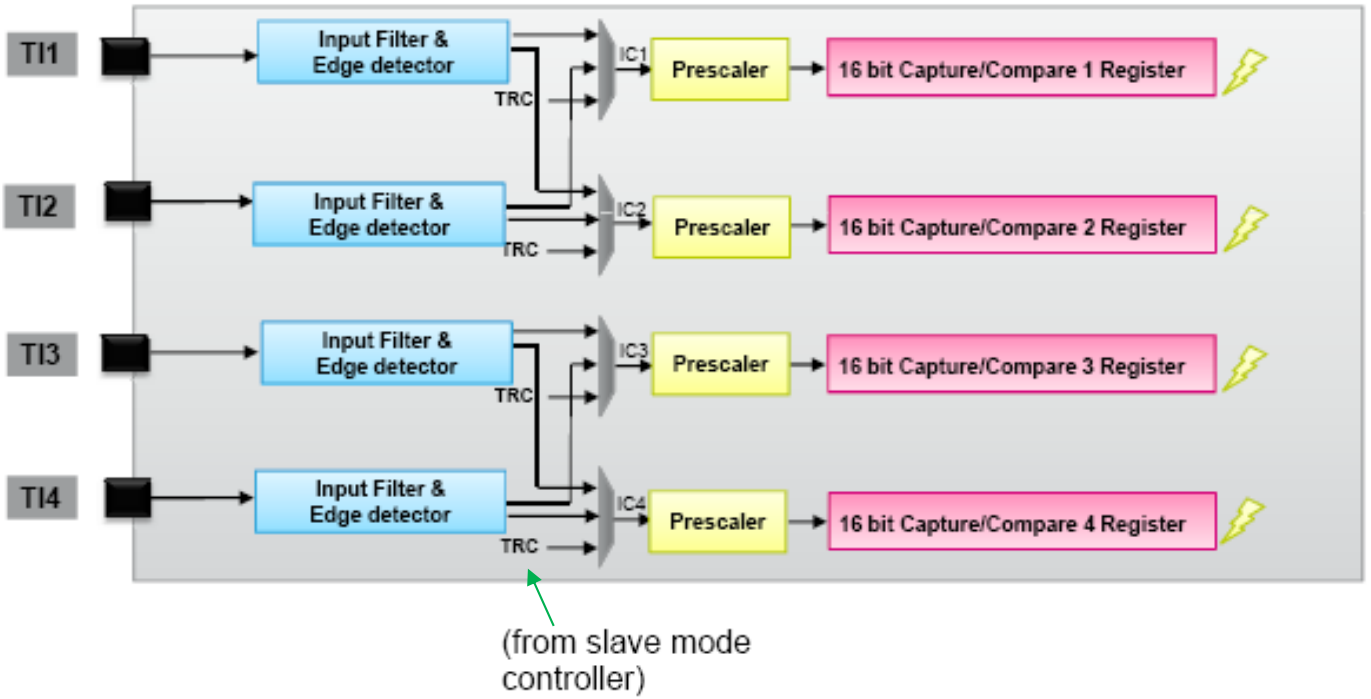


Capture Compare Array presentation

- Up to 4 channels
  - TIM2/3/4/5/19 have 4 channels
  - TIM12/15 have 2 channels
  - TIM13/14/16/17 have one channel
  - TIM6/7/18 have no channels
- Programmable bidirectional channels
  - Input direction: channel configured in **Capture** mode
  - Output direction: Channel configured in **Compare** mode
- **Channel's main functional blocks**
  - Capture/Compare register
  - Input stage for capture
    - 4-bit digital filter
    - Input Capture Prescaler:
  - Output stage for Compare
    - Output control block

Input Capture Mode (1/2)

- Capture stage architecture



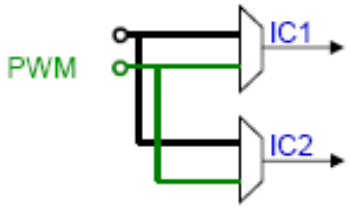
Input Capture Mode (2/2)

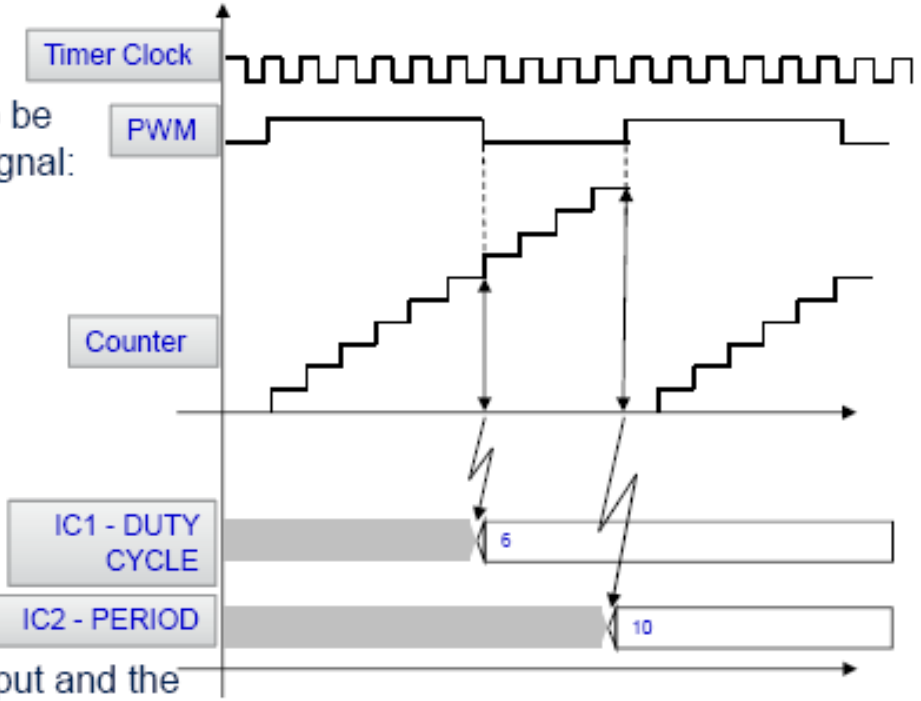
- Flexible mapping of TIx inputs to channels' inputs ICx
  - {TI1->IC1}, {TI1->IC2}, {TI2->IC1} and {TI2->IC2} are possible
- When an active Edge is detected on ICx input, the counter value is latched in the corresponding CCR register.
- When a Capture Event occurs, the corresponding CCXIF flag is set and an interrupt or a DMA request can be sent if they are enabled.
- An over-capture flag for over-capture signaling
  - Takes place when a Capture Event occurs while the CCxIF flag was already high

PWM Input Mode

- IC1 and IC2 must be configured to be connected together to the PWM signal:

⇨ IC1 and IC2 are redirected internally to be mapped to the same external pin TI1 or TI2.



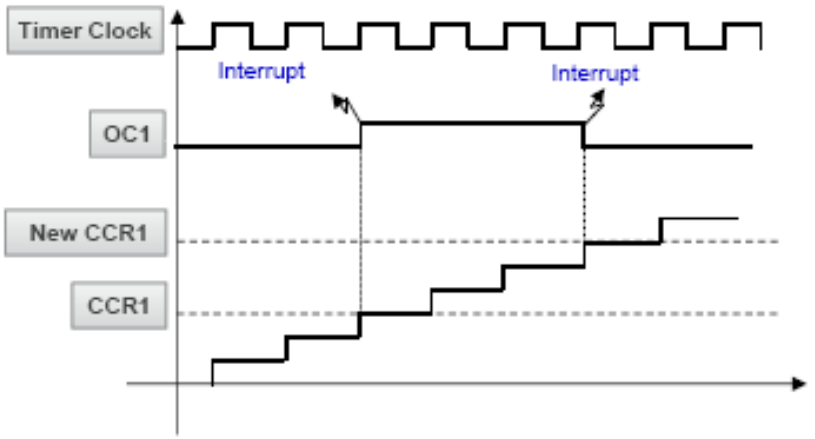


- IC1 and IC2 active edges must have opposite polarity.
- IC1 or IC2 is selected as trigger input and the slave mode controller is configured in reset mode.
- The PWM Input functionality enables the measurement of the period and the pulse width of an external waveform.

Output Compare Mode

- The Output Compare is used to control an output waveform or indicate when a period of time has elapsed.

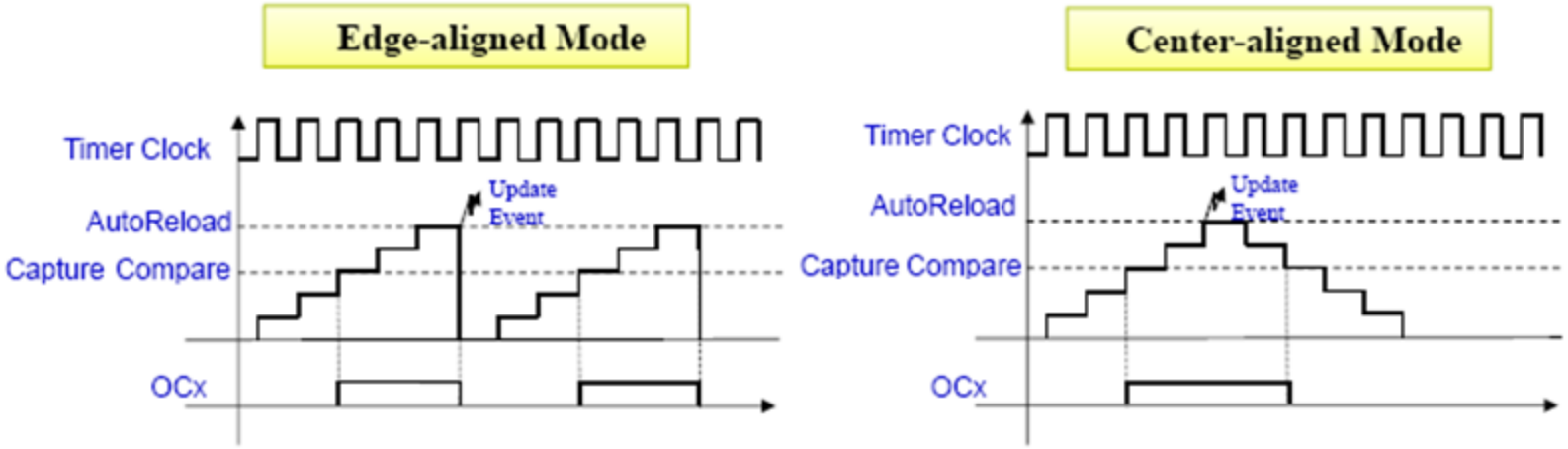
- When a match is found between the capture/compare register and the counter:
  - The corresponding output pin is assigned to the programmable Mode, it can be:
    - Set
    - Reset
    - Toggle
    - Remain unchanged
  - Set a flag in the interrupt status register
  - Generates an interrupt if the corresponding interrupt mask is set
  - Send a DMA request if the corresponding enable bit is set



- The CCRx registers can be programmed with or without preload registers

PWM Mode

- Available on all channels
- Two PWM mode available
  - PWM mode 1
  - PWM mode 2
  - Each PWM mode behavior (waveform shape) depends on the counting direction

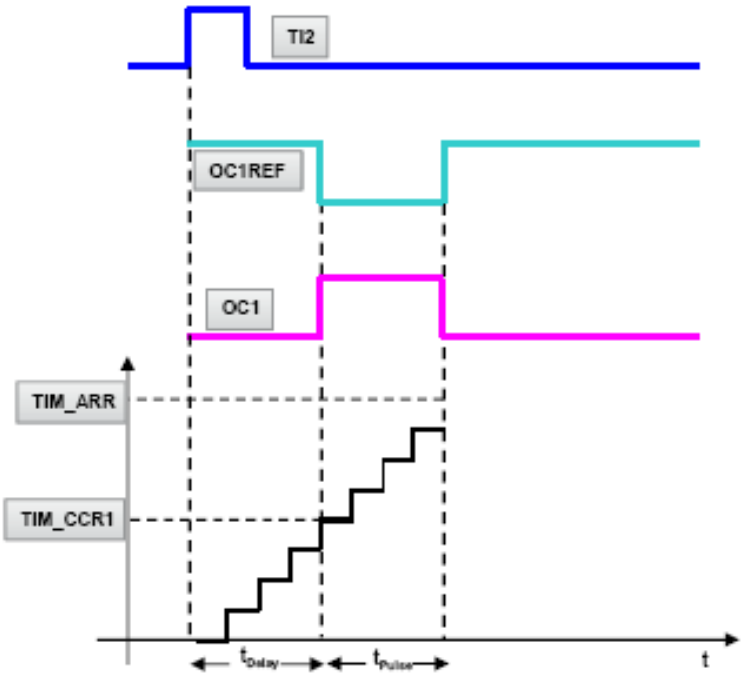


PWM mode 2

TIM2/5 TIM3/4/19 TIM12 TIM15 TIM13/14 TIM16/17

One Pulse Mode (1/2)

- One Pulse Mode (OPM) is a particular case of Output Compare mode
- It allows the counter to be started in response to a stimulus and to generate a pulse
  - With a programmable length
  - After a programmable delay
- There are two One Pulse Mode waveforms selectable by software:
  - Single Pulse
  - Repetitive Pulse



TIM2/5 TIM3/4/19 TIM12 TIM15 TIM13/14 TIM16/17

One Pulse Mode (2/2)

Exercise:

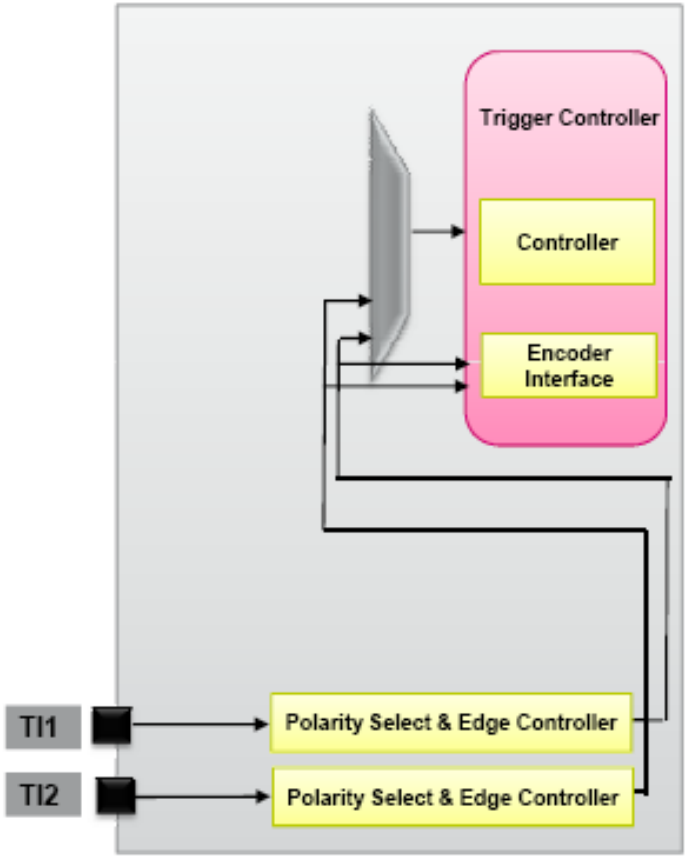
How to configure One Pulse Mode to generate a repetitive Pulse in response to a stimulus ?

One Pulse Mode configuration steps

1. Input Capture Module Configuration:
   - i. Map TIxFPx on the corresponding TIx.
   - ii. TIxFPx Polarity configuration.
   - iii. TIxFPx Configuration as trigger input.
   - iv. TIxFPx configuration to start the counter (Trigger mode)
2. Output Compare Module Configuration:
   - i. OCx configuration to generate the corresponding waveform.
   - ii. OCx Polarity configuration.
   - iii. t_{Delay} and t_{Pulse} definition.
3. **One Pulse Module Selection: Set or Reset the corresponding bit (OPM) in the Configuration register (CR1).**

Encoder Interface (1/2)

- Encoders are used to measure position and speed of mobile systems (either linear or angular)
- The encoder interface mode acts as an external clock with direction selection
- Encoders and Microcontroller connection example:
  - A can be connected directly to the MCU without external interface logic.
  - The third encoder output which indicates the mechanical zero position, may be connected to an external interrupt and trigger a counter reset.
- Encoder enhancement
  - A copy of the Update Interrupt Flag (UIF) is copied into bit 31 of the counter register
  - Simultaneous read of the Counter value and the UIF flag : Simplify the position determination



TIM2/5 TIM3/4/19 TIM12 TIM15

Encoder Interface (2/2)

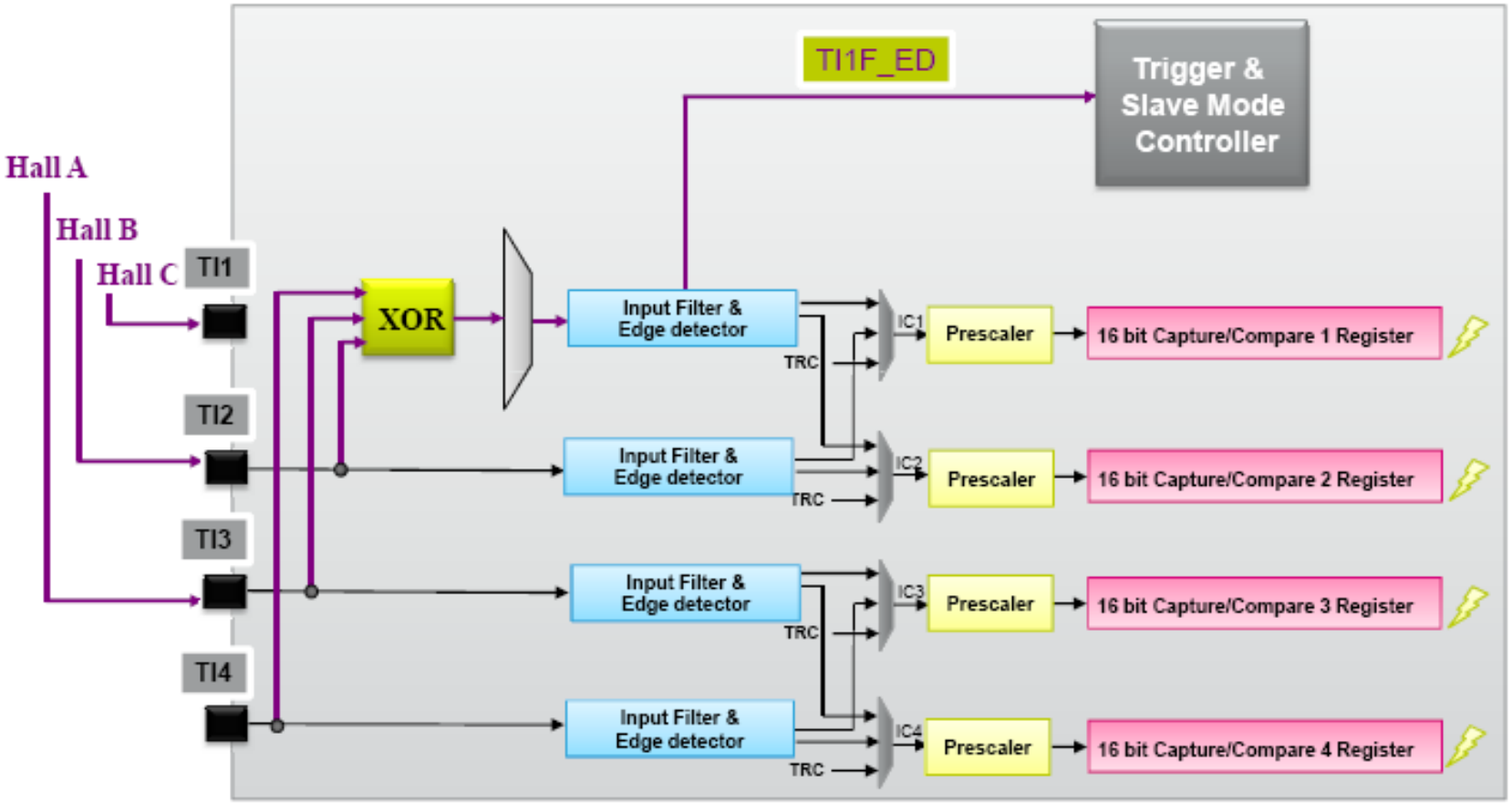
Exercise:

How to configure the Encoder interface to detect the rotation direction of a motion system?

Encoder interface configuration steps:

1. Select the active edges: example counting on TI1 and TI2.
2. Select the polarity of each input: example TI1 and TI2 polarity not inverted.
3. Select the corresponding Encoder Mode.
4. Enable the counter.

Hall sensor Interface (1/2)



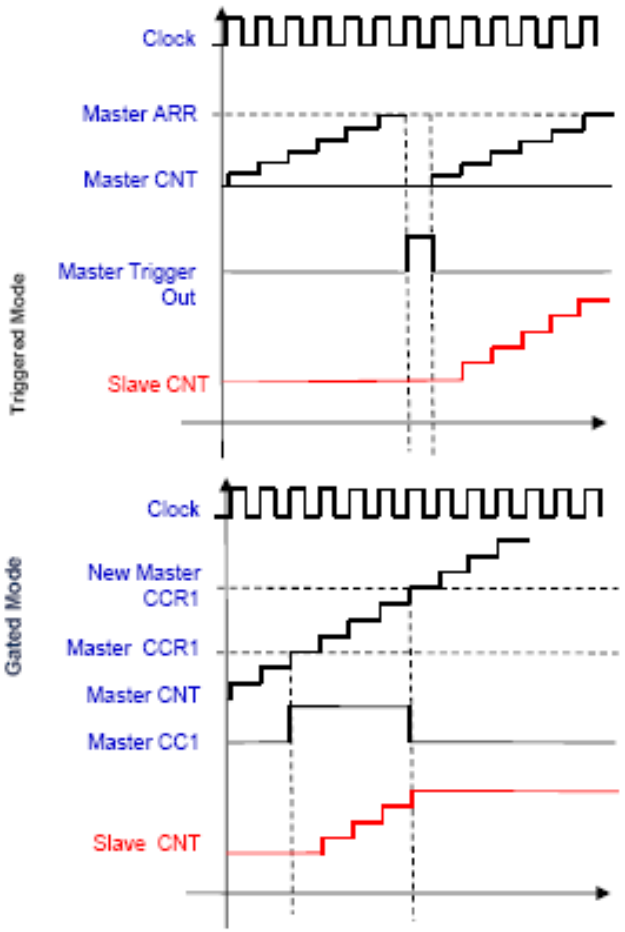
TIM2/5 TIM3/4/19

Hall sensor Interface (2/2)

- Hall sensors are used for:
  - Speed detection
  - Position sensor
  - Brushless DC Motor Sensor
- How to configure the TIM to interface with a Hall sensor?
  - Select the hall inputs for TI1: TI1S bit in the CR2 register
  - The slave mode controller is configured in reset mode
  - TI1F_ED is used as input trigger
- To measure a motor speed:
  - Use the Capture/Compare Channel 1 in Input Capture Mode
  - The Capture Signal is the TRC signal
  - The captured value which correspond to the time elapsed between 2 changes on the inputs, gives an information about the motor speed

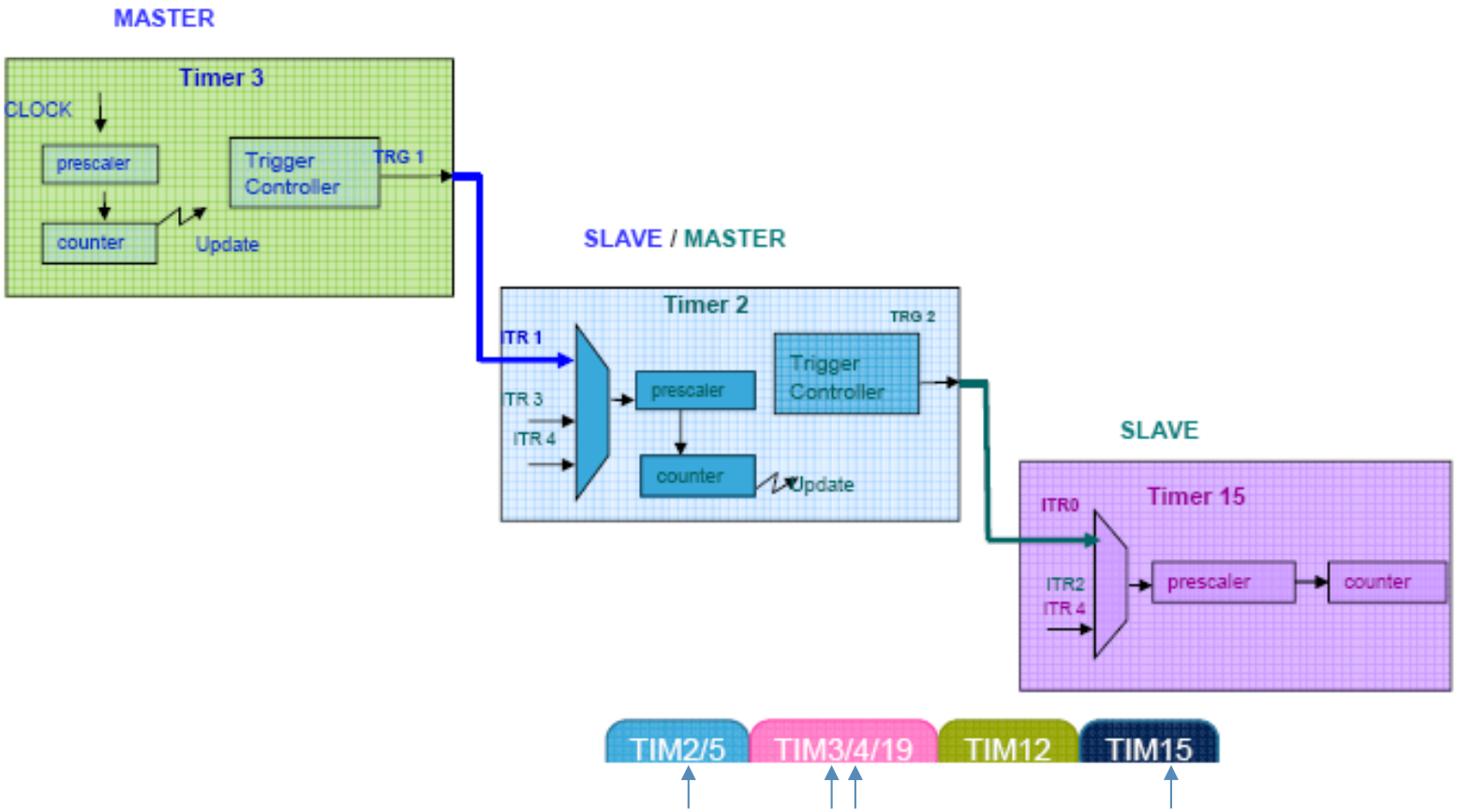
Synchronization Mode Configuration

- The Trigger Output can be controlled on:
  - Counter reset
  - Counter enable
  - Update event
  - OC1 / OC1Ref / OC2Ref / OC3Ref / OC4Ref signals
- The slave timer can be controlled in two modes:
  - Triggered mode : only the start of the counter is controlled
  - Gated Mode: Both start and stop of the counter are controlled
  - Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter



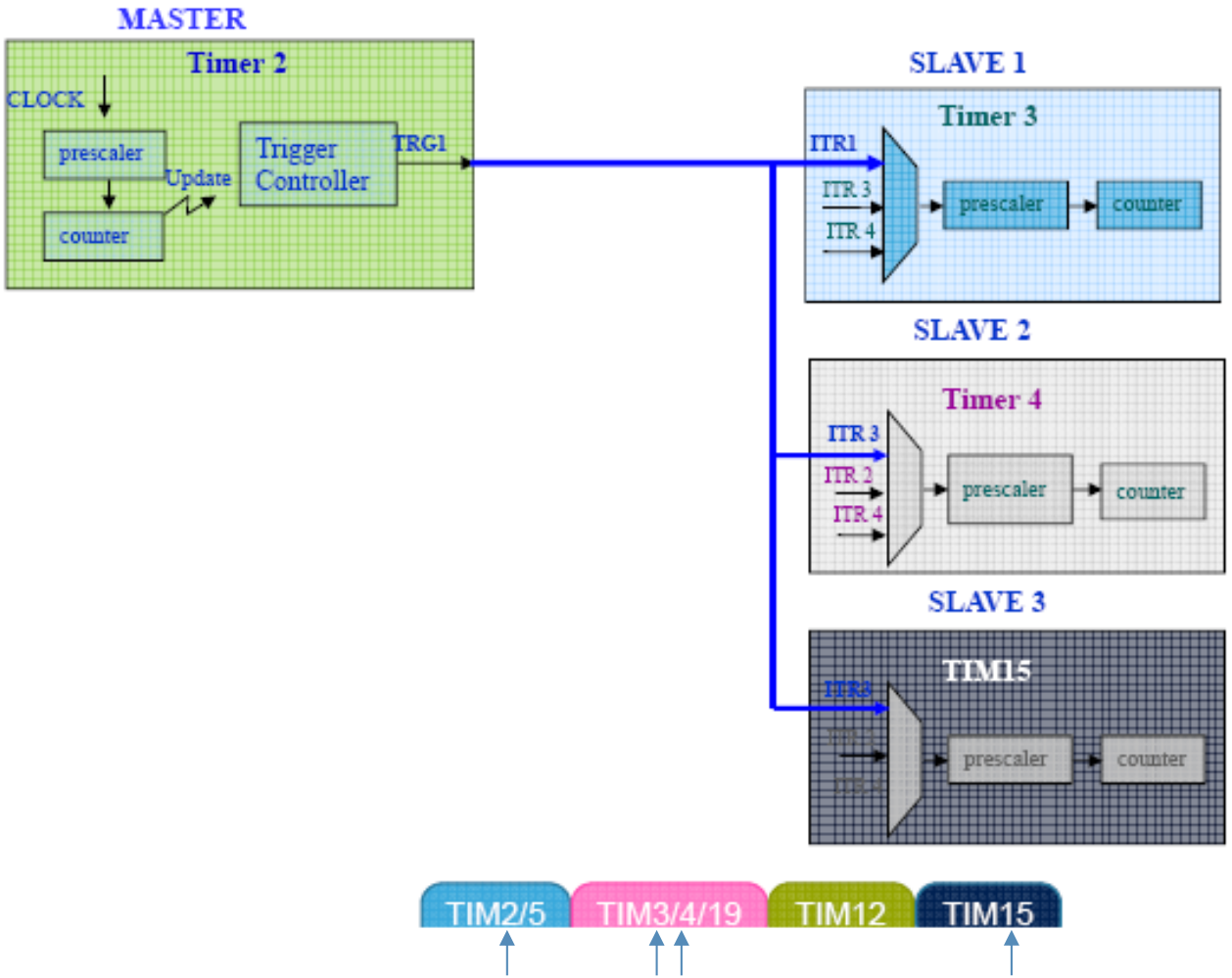
Synchronization: Configuration examples (1/3)

- Cascade mode:
  - TIM3 used as master timer for TIM2
  - TIM2 configured as TIM3 slave, and master for TIM15



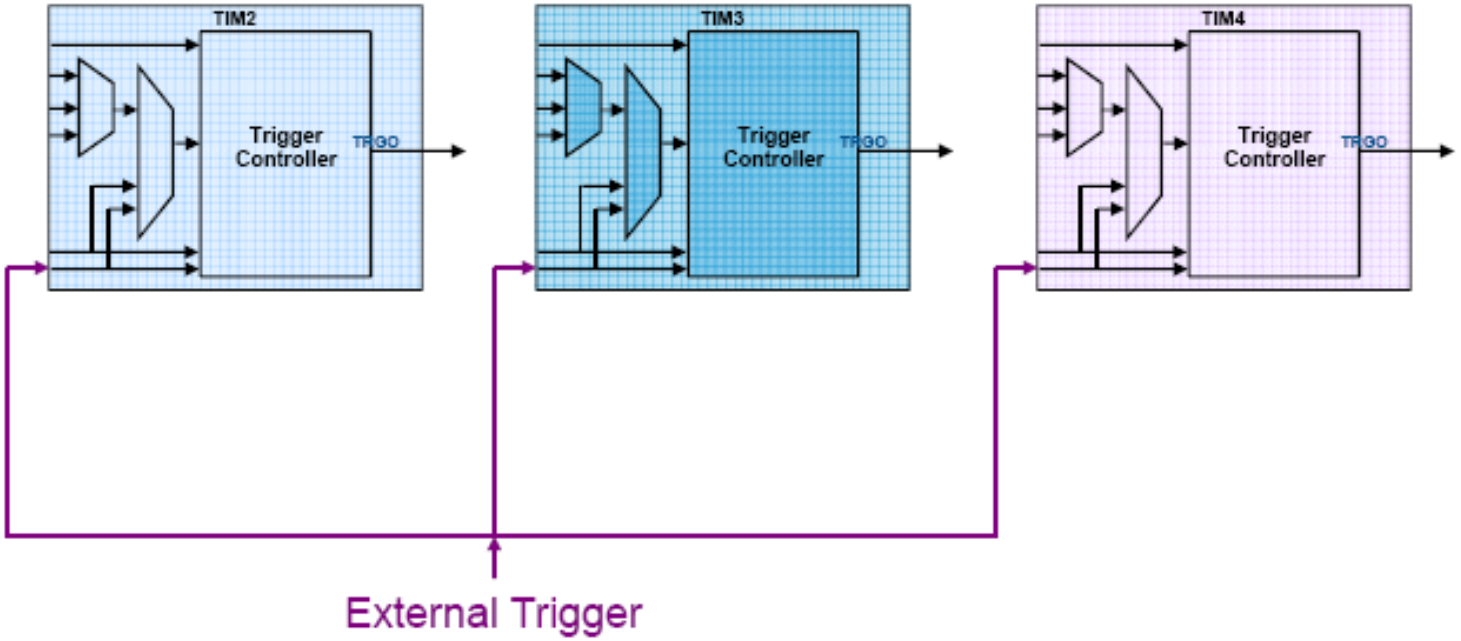
Synchronization: Configuration examples (2/3)

- One Master several slaves: TIM2 used as master for TIM3, TIM4 and TIM15



Synchronization: Configuration examples (3/3)

- Timers and external trigger synchronization
  - TIM2, TIM3 and TIM4 are slaves for an external signal connected to respective Timers inputs



Basic timers (TIM6/TIM7)

- The main block of the programmable timer is a 16-bit, up counter with its related auto-reload register. The counter clock can be divided by a prescaler.
- The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.
- They may be used as generic timers for time-base generation but they are also specifically used to drive the digital-to-analog converter (DAC).
- The timers are completely independent, and do not share any resources.

STM32F3 Microcontroller Reference Manual (pages 587-599)

TIM6/TIM7 main features

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also "on the fly") the counter clock frequency by any factor between 1 and 65536
- Synchronization circuit to trigger the DAC
- Interrupt/DMA generation on the update event: counter overflow

TIM6/TIM7 registers

Description	Name	Offset
Control Register 1	TIMx_CR1	0x00
Control Register 2	TIMx_CR2	0x04
DMA/Interrupt Enable Register	TIMx_DIER	0x0C
Status Register	TIMx_SR	0x10
Event Generation Register	TIMx_EGR	0x14
Counter	TIMx_CNT	0x24
Prescaler	TIMx_PSC	0x28
Auto-Reload Register	TIMx_ARR	0x2C

•STM32F3 Microcontroller Datasheet, page 593-598

TIM6/TIM7 Registers Relevant Bits

Reg	Bits	Name	Description	Mask
TIMx_CR1	11	UIFREMAP	UIF status bit remapping	0x00000800
	7	ARPE	Auto-reload preload enable	0x00000080
	3	OPM	One-pulse mode	0x00000008
	2	URS	Update request source	0x00000004
	1	UDIS	Update disable	0x00000002
	0	**CEN**	**Counter enable**	**0x00000001**
TIMx_CR2	6:4	MMS	Master mode selection	
TIMx_DIER	8	UDE	Update DMA request enable	0x00000100
	0	**UIE**	**Update interrupt enable**	**0x00000001**
TIMx_SR	**0**	**UIF**	**Update interrupt flag**	**0x00000001**
TIMx_EGR	0	UG	Update generation	0x00000001

TIM6/TIM7 register map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMAP	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																					0				0				0	0	0	0
0x04	TIMx_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS[2:0]			Res.	Res.	Res.	Res.
	Reset value																										0	0	0				
0x08	Reserved																																
0x0C	TIMx_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UDE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIE
	Reset value																								0								0
0x10	TIMx_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
	Reset value																																0
0x14	TIMx_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UG
	Reset value																																0
0x18-0x20	Reserved																																
0x24	TIMx_CNT	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIMx_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Code Snippet


```
//Timer7 Prescaler :550; Preload = 65455-1;
// Actual Interrupt Time = 1000 ms

#define UIE 0x00000001 // Update interrupt enable
#define CEN 0x00000001 // Counter enable
#define UIF 0x00000001 // Update interrupt flag
#define RCC_APB1ENR_TIM7EN 0x00000020

void InitTimer7(void){
  RCC->APB1ENR |= RCC_APB1ENR_TIM7EN; // Enable clock for TIM7
  TIM7->CR1 &= ~CEN; // Disable TIM7 interrupt
  TIM7->PSC = 550;
  TIM7->ARR = 65454;
  NVIC_EnableIRQ(TIM7_IRQn);
  TIM7->DIER |= UIE; // Enable TIM7 interrupt
  TIM7->CR1 |= CEN; // TIM7 enable
}

void TIM7_IRQHandler (void) {
  TIM7->SR &= ~UIF; // Clear UIF
  //Enter your code here
}
```

$36{,}000{,}000/2^6=550$

$36{,}000{,}000/550=65454.54545$

PRESCALER: 550

PRELOAD: 65455

General-purpose timers (TIM2/TIM3/TIM4)

TIM2 TIM3/4

- The general-purpose timers consist of a 16-bit or 32-bit auto-reload counter driven by a programmable prescaler.
- They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare and PWM).
- Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.
- The timers are completely independent, and do not share any resources. They can be synchronized together.

STM32F3 Microcontroller Reference Manual (pages 447-513)

TIM2/TIM3/TIM4 main features

TIM2 TIM3/4

- 16-bit (TIM3 and TIM4) or 32-bit (TIM2) up, down, up/down auto-reload counter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65536.
- Up to 4 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (Edge- and Center-aligned modes)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers.

TIM2/TIM3/TIM4 main features

TIM2 TIM3/4

- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

Most Important TIM3 Registers

Description	Name	TIM2	TIM3/4
Control Register 1	TIMx_CR**1**		
Control Register 2	TIMx_CR**2**		
DMA/Interrupt Enable Register	TIMx_DIER		
Status Register	TIMx_SR		
Event Generation Register	TIMx_EGR		
Capture/Compare Mode Register 1	TIMx_CCMR**1**		
Capture/Compare Mode Register 2	TIMx_CCMR**2**		
Capture/Compare Enable Register	TIMx_CCER		
Counter	TIMx_CNT		
Prescaler	TIMx_PSC		
Auto-Reload Register	TIMx_ARR		
Capture/Compare Register 1	TIMx_CCR1		
Capture/Compare Register 2	TIMx_CCR2		
Capture/Compare Register 3	TIMx_CCR3		
Capture/Compare Register 4	TIMx_CCR4		

TIM2/3/4
TIM6/7

TIM3 Some Important Bits

TIM2 TIM3/4

Reg	Bits	Name	Description
TIMx_CR1	11	UIFREMAP	UIF status bit remapping
	7	ARPE	Auto-reload preload enable
	3	OPM	One-pulse mode
	2	URS	Update request source
	1	UDIS	Update disable
	0	**CEN**	**Counter enable**
TIMx_CR2	6:4	MMS	Master mode selection
TIMx_DIER	8	UDE	Update DMA request enable
	4	CC4IE	Capture/Compare 4 interrupt enable
	3	CC3IE	Capture/Compare 4 interrupt enable
	2	CC2iE	Capture/Compare 4 interrupt enable
	1	CC1IE	Capture/Compare 4 interrupt enable
	0	**UIE**	**Update interrupt enable**
TIMx_SR	**0**	**UIF**	**Update interrupt flag**
TIMx_EGR	0	UG	Update generation

TIM2/3/4
TIM6/7

capture/compare mode register z (TIMx_CCMRz) z={1,2}

TIM2 TIM3/4

- The channels can be used in input (capture mode) or in output (compare mode).
- The direction of a channel is defined by configuring the corresponding CCyS bits.
- All the other bits of this register have a different function in input and in output mode.

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1C	**TIMx_CCMR2** *Output Compare mode*	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]	O24CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2 *Input Capture mode*	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TIMx_CCMRz (Output Compare Mode)

TIM2 TIM3/4

Field	Description	Operation
OCyM[3:0]	Output Compare y Mode	define the behavior of the output reference signal OCyREF from which OCy and OCyN are derived.
OCyCE	Output compare y clear enable	
OCyPE	Output compare y preload enable	
OCyFE	Output compare y fast enable	
CCyS[1:0]	Capture/Compare y selection	**00:** CCy channel is configured as output **01:** CCy channel is configured as input, ICy is mapped on TIy **10:** CCy channel is configured as input; if y is odd, ICy is mapped on TIy+1, else ICy is mapped on TIy-1

OCyM[3:0] field of TIMx_CCMRz

TIM2 TIM3/4

Value	Action (in Output Compare)
0000	Frozen
0001	OCyREF = 1 when the counter CNT = CCRy
0010	OCyREF = 0 when the counter CNT = CCRy
0011	OCyREF toggles when CNT = CCRy
0100	OCyREF is forced 0
0101	OCyREF is forced 1
0110	PWM mode 1: When ↑ if CNT<CCRy then OCyREF=1 else OCyREF= 0. When ↓ if CNT>CCRy then OCyREF= 0 else OCyREF=1
0111	PWM mode 2: When ↑ if CNT<CCRy then OCyREF=0 else OCyREF=1. When ↓ CNT>CRy then OCyREF=1else OCyREF=0.
1000	Retriggerable OPM mode 1
1001	Retriggerable OPM mode 2
1100	Combined PWM mode 1
1101	Combined PWM mode 2
1110	Asymmetric PWM mode 1
1111	Asymmetric PWM mode 2

TIMx_CCMRz (Input Capture Mode)

TIM2 TIM3/4

Field	Description	Operation
ICyF	Input capture y filter	
ICyPSC[1:0]	Input capture y prescaler	
CCyS[1:0]	Capture/Compare **y** selection	**00:** CCy channel is configured as output **01:** CCy channel is configured as input, ICy is mapped on TIy **10:** CCy channel is configured as input; if y is odd, ICy is mapped on TIy+1, else ICy is mapped on TIy-1

Capture/Compare Enable Register (TIMx_CCER) (CCy channel as output)

TIM2 TIM3/4

Bits	Description	Operation
CCyNP	Capture/Compare y output Polarity	CC1NP must be kept cleared in this case.
CCyP	Capture/Compare y output Polarity	0: OCy active high 1: OCy active low
CCyE	Capture/Compare y output enable.	0: Off - OCy is not active 1: On - OCy signal is output on the corresponding output pin

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x20	TIMx_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
	Reset value																	0		0	0	0		0	0	0		0	0	0		0	0

Capture/Compare Enable Register (TIMx_CCER) (CCy channel as input)

TIM2 TIM3/4

Bits	Description	Operation
CCyNP MSB	Capture/Compare y output Polarity	This bit is used in conjunction with CCyP to define TIyFP1 polarity.
CCyP LSB	Capture/Compare y output Polarity	CCyNP/CCyP bits select TIyFP1 polarity for trigger or capture operations. **00:** rising edge/noninverted • rising edge (capture, trigger in reset, external clock or trigger mode) • not inverted (trigger in gated mode, encoder mode) **01:** falling edge/inverted • falling edge (capture, trigger in reset, external clock or trigger mode) • inverted (trigger in gated mode, encoder mode) **10:** reserved, do not use this configuration **11:** both edges/noninverted • edges (capture, trigger in reset, external clock or trigger mode) • not inverted (trigger in gated mode). This configuration must not be used for encoder mode.
CCyE	Capture/Compare y output enable	This bit determines if a capture of the counter value can actually be done into CCRy or not. **0:** Capture disabled **1:** Capture enabled

TIM2/3/4 Registers

TIM2 TIM3/4

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMAP	Reserved	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x04	**TIMx_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
	Reset value																									0	0	0	0	0			
0x08	**TIMx_SMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMS[3]	ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x0C	**TIMx_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value																		0	0	0	0	0	0	0		0		0	0	0	0	0
0x10	**TIMx_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value																				0	0	0	0			0		0	0	0	0	0
0x14	**TIMx_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																										0		0	0	0	0	0
0x18	**TIMx_CCMR1** *Output Compare mode*	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR1 *Input Capture mode*	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]	CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TIM2/3/4 Registers

TIM2 TIM3/4

		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1C	**TIMx_CCMR2** *Output Compare mode*	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]	O24CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2 *Input Capture mode*	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**TIMx_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
	Reset value																	0		0	0	0		0	0	0		0	0	0		0	0
0x24	**TIMx_CNT**	CNT[31] or UIFCPY	CNT[30:16] (TIM2 only, reserved on the other timers)															CNT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIMx_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIMx_ARR**	ARR[31:16] (TIM2 only, reserved on the other timers)																ARR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	Reserved																																

TIM2/3/4 Registers

TIM2 TIM3/4

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x34	**TIMx_CCR1**	CCR1[31:16] (TIM2 only, reserved on the other timers)																CCR1[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**TIMx_CCR2**	CCR2[31:16] (TIM2 only, reserved on the other timers)																CCR2[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	**TIMx_CCR3**	CCR3[31:16] (TIM2 only, reserved on the other timers)																CCR3[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	**TIMx_CCR4**	CCR4[31:16] (TIM2 only, reserved on the other timers)																CCR4[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	Reserved																																
0x48	**TIMx_DCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x4C	**TIMx_DMAR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STM32F3 Microcontroller Reference Manual (pages 512-513)

GP Timer: Code Example

TIM2 TIM3/4

```
#include "stm32f30x.h"
int main(void) {
  // At this stage the microcontroller clock setting is already configured

  // GPIOE clock enable
  RCC->AHBENR |= RCC_AHBENR_GPIOEEN;

  // Configure PE15 in output push-pull mode
  GPIOE->MODER   |= 1UL << 15*2;  // Output
  GPIOE->OTYPER  |= 0L  << 15;    // Push-pull
  GPIOE->OSPEEDR |= 3UL << 15*2;  // 50 MHz
  GPIOE->PUPDR   |= 0L  << 15*2;  // No pull-up resistance

  // TIM3 clock enable
  RCC->APB1ENR |= RCC_APB1ENR_TIM3EN;

  // delay = 0.5 = (PSC+1)*ARR/FAPB1 = 60000*600/72000000
  TIM3->PSC = 599;          // Set pre-scaler to 600 (PSC + 1)
  TIM3->ARR = 60000;        // Auto reload value 600000
  TIM3->CR1 = TIM_CR1_CEN;  // Enable timer

  while (1) {
    if(TIM3->SR & TIM_SR_UIF) {  // if UIF flag is set
      TIM3->SR &= ~TIM_SR_UIF;   // clear UIF flag
      GPIOE->ODR ^= 1L << 15;    // toggle LED state
    }
  }
}
```

GP Timer: Code Example (using ISR) (1)

TIM2 TIM3/4

```
#include "stm32f30x.h"

int main(void)
{
  // At this stage the microcontroller clock setting is already configured

  // GPIOE clock enable
  RCC->AHBENR |= RCC_AHBENR_GPIOEEN;

  // Configure PE15 in output push-pull mode
  GPIOE->MODER   |= 1   <<(15*2); // Output
  GPIOE->OTYPER  |= 0   << 15;    // Push-pull
  GPIOE->OSPEEDR |= 3UL <<(15*2); // 50 MHz
  GPIOE->PUPDR   |= 0   <<(15*2); // No pull-up resistance

  // TIM3 clock enable
  RCC->APB1ENR |= RCC_APB1ENR_TIM3EN;

  // delay = 0.5 = (PSC+1)*ARR/FAPB1 = 60,000*600/72,000,000
  TIM3->PSC = 599;          // Set pre-scaler to 600 (PSC + 1)
  TIM3->ARR = 60000;        // Auto reload value 600000
  TIM3->CR1 = TIM_CR1_CEN;  // Enable timer

  TIM3->DIER    |= 1 << 0;   // enable interrupt
  NVIC->ISER[0] |= 1 << 29;  // enable TIM3 interrupt in NVIC

  while (1);
}
```

GP Timer: Code Example (using ISR) (2)

TIM2 TIM3/4

```
void TIM3_IRQHandler (void)
{
  if(TIM3->SR & TIM_SR_UIF)     // if UIF flag is set
  {
    TIM3->SR &= ~TIM_SR_UIF;    // clear UIF flag
    GPIOE->ODR ^= 1L << 15;     // toggle LED state
  }
}
```

PWM

TIM2 TIM3/4

- Each of the timers 2 to 4 has four output channels. For example, the output channels for timer 3 are mapped as follows:

TIM3_CH1	TIM3_CH2	TIM3_CH3	TIM3_CH4
PA6 (AF2)	PA4 (AF2)	PB0 (AF2)	PB1 (AF2)
PB4 (AF2)	PA7 (AF2)	PC8 (AF2)	PB7 (AF10)
PC6 (AF2)	PB5 (AF2)	PE4 (AF2)	PC9 (AF2)
PE2 (AF2)	PC7 (AF2)		PE5 (AF2)
	PE3 (AF2)		

•STM32F3 Microcontroller Datasheet (pages 41-48)

Timer Configuration for PWM

TIM2 TIM3/4

- Pulse width modulation mode allows you to generate a signal with a period determined by the value of the ARR register and a duty cycle determined by the value of the CCRy register.
- The PWM mode can be selected independently on each channel (one PWM per OCy output) by writing 110 (PWM mode 1) or 111 (PWM mode 2) in CCMRz.OCyM bits.

PWM mode (cont)

- OCy polarity is software programmable using the CCER.CCyP bit.
  - It can be programmed as active high or active low.
- OCy output is enabled by the CCER.CCyE bit.
- In PWM mode (1 or 2), CNT and CCRy are always compared to determine whether CCRy ≤ CNT or CNT ≤ CCRy (depending on the direction of the counter).
- The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CR1.CMS bits.

PWM Code Example

TIM2 TIM3/4

```
#include "stm32f30x.h"

int main(void)
{

  RCC->AHBENR  |= RCC_AHBENR_GPIOCEN; // Enable GPIOC clock
  RCC->APB1ENR |= RCC_APB1ENR_TIM3EN; // Enable Timer 3 clock

  // PC8 configuration
  GPIOC->MODER   |= 2 << (8*2);      // Alternate function mode
  GPIOC->OTYPER  |= 0 << 8;          // Output push-pull (reset state)
  GPIOC->OSPEEDR |= 0 << (8*2);      // 2 MHz High speed
  GPIOC->AFR[1]  |= 2 << ((8-8)*4); // Select AF2 for PC8: TIM3_CH3

  // Period = 600*6000/72000000 = 50ms, Duty = 25ms
  TIM3->PSC    = 5999;  // Set prescaler to 6000 (PSC + 1)
  TIM3->ARR    = 600;   // Auto reload value 600
  TIM3->CCR3   = 600/5; // Start PWM duty for channel 3
  TIM3->CCMR2 |= TIM_CCMR2_OC3M_2 | TIM_CCMR2_OC3M_1; // PWM mode 1 on channel 3
  TIM3->CCER  |= TIM_CCER_CC3E; // Enable compare on channel 3
  TIM3->CR1   |= TIM_CR1_CEN;   // Enable timer

  while (1) {}
}
```

pwm2.c D:\PROJECTS\STM32\TESTS\PWM\pwm.uvproj

Input capture mode

- In Input capture mode, the Capture/Compare Registers (TIMx_CCRy) are used to latch the value of the counter after a transition detected by the corresponding ICy signal.
- When a capture occurs, the corresponding CCyIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled.
- If a capture occurs while the CCyIF flag was already high, then the over-capture flag CCyOF (TIMx_SR register) is set.
- CCyIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRy register. CCyOF is cleared when you write it to 0.

Input Capture Procedure (1)

Example: capturing the counter value in TIMx_CCR1 when TI1 input rises.

- Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register.
  - As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
- Select the edge of the active transition on the TI1 channel by writing the CC1P and CC1NP and CC1NP bits to 000 in the TIMx_CCER register (rising edge in this case).

Input Capture Procedure (2)

TIM2 TIM3/4

- Program the input prescaler.
  - In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).
- Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
- If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

Input Capture Procedure (3)

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag).
  - CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

Output compare mode

- This function is used to control an output waveform or indicating when a period of time has elapsed.
- When a match is found between the capture/compare register and the counter, the output compare function:
  - Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCyM bits in the TIMx_CCMRz register) and the output polarity (CCyP bit in the TIMx_CCER register).
  - The output pin can keep its level (OCyM=000), be set (OCyM=001), be cleared (OCyM=010) or can toggle (OCyM=011) on match.
  - Sets a flag in the interrupt status register (CCyIF bit in the TIMx_SR register).
  - Generates an interrupt if the corresponding interrupt mask is set (CCyIE bit in the TIMx_DIER register).

Output compare mode (2)

- In output compare mode, the update event UEV has no effect on OCyREF and OCy output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Output Compare: Procedure

TIM2 TIM3/4

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRy registers.
3. Set the CCyIE and/or CCyDE bits if an interrupt and/or a DMA request is to be generated.
4. Select the output mode. For example, you must write OCyM=011, OCyPE=0, CCyP=0 and CCyE=1 to toggle OCy output pin when CNT matches CCRy, OCy is enabled and active high.
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

Forced output mode (1)

- In output mode (CCyS bits = 00 in the TIMx_CCMRz register), each output compare signal (OCyREF and then OCy) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.
- To force an output compare signal (OCyREF/OCy) to its active level, you just need to write 101 in the OCyM bits in the corresponding TIMx_CCMRz register. Thus OCyREF is forced high (OCxREF is always active high) and OCy get opposite value to CCyP polarity bit.
  - e.g.: CCyP=0 (OCy active high) => OCy is forced to high level.

Forced output mode (2)

- OCyREF signal can be forced low by writing the OCyM bits to 100 in the TIMx_CCMRz register.
- Anyway, the comparison between the TIMx_CCRy shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly.

Pins connected to TIM2

TIM2_CH1_ETR	TIM2_CH2	TIM2_CH3	TIM2_CH4
PA0 (AF1)	PA1 (AF1)	PA2 (AF1)	PA3 (AF1)
PA5 (AF1)	PB3 (AF1)	PA9 (AF10)	PA10 (AF10)
PA15 (AF1)	PD4 (AF2)	PB10 (AF1)	PB11 (AF1)
PD3 (AF2)		PD7 (AF2)	PD6 (AF2)

SMS[3:0] Field of TIMx Slave Mode Control Register (TIMx_SMCR)

SMS: Slave mode selection

Bits	Mode	Description
0000	Slave mode disabled	if CEN = ‘1 then the prescaler is clocked directly by the internal clock. Counter counts up/down on TI2FP2 edge depending on TI1FP1 level.
0001	Encoder mode 1	Counter counts up/down on TI2FP2 edge depending on TI1FP1 level.
0010	Encoder mode 2	Counter counts up/down on TI1FP1 edge depending on TI2FP2 level.
0011	Encoder mode 3	Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.
0100	Reset Mode	0Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.
0101	Gated Mode	The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.
0110	Trigger Mode	The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.
0111	External Clock Mode 1	Rising edges of the selected trigger (TRGI) clock the counter.
1000	Combined reset + trigger mode	Rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers and starts the counter.

TS[1:0] Field of TIMx Slave Mode Control Register (TIMx_SMCR)

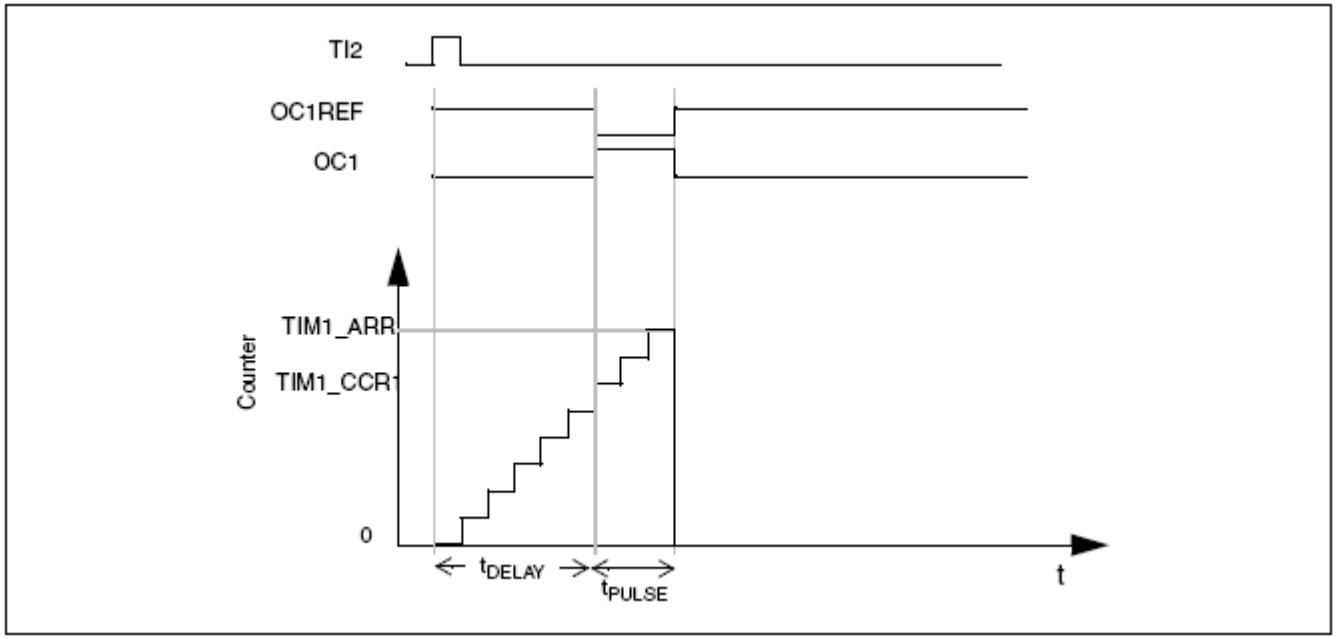
- **TS:** Trigger selection
- This bit-field selects the trigger input to be used to synchronize the counter.

Bits	Identification
000	Internal Trigger 0 (ITR0); reserved
001	Internal Trigger 1 (ITR1)
010	Internal Trigger 2 (ITR2)
011	Internal Trigger 3 (ITR3); reserved
100	TI1 Edge Detector (TI1F_ED)
101	Filtered Timer Input 1 (TI1FP1)
110	Filtered Timer Input 2 (TI2FP2)
111	(ETRF) External Trigger input

One-pulse mode

- It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.
- Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. You select One-pulse mode by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.
- A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:
  - CNT<CCRy≤ ARR (in particular, 0<CCRy),

One-pulse mode



One-pulse mode

- For example you may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.
- Let's use TI2FP2 as trigger 1:
  - Map TI2FP2 on TI2 by writing IC2S=01 in the TIMx_CCMR1 register.
  - TI2FP2 must detect a rising edge, write CC2P=0 and CC2NP=0 in the TIMx_CCER register.
  - Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS=110 in the TIMx_SMCR register.
  - TI2FP2 is used to start the counter by writing SMS to '110 in the TIMx_SMCR register (trigger mode).

One-pulse mode

- The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).
  - The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
  - The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
  - Let's say you want to build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the auto-reload value.
    - To do this you enable PWM mode 2 by writing OC1M=111 in the TIMx_CCMR1 register. You can optionally enable the preload registers by writing OC1PE=1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case you have to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '0 in this example.

One-pulse mode

- In our example, the DIR and CMS bits in the TIMx_CR1 register should be low.
- You only want 1 pulse (Single mode), so you write 1 in the OPM bit in the TIMx_CR1register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to 0, the Repetitive Mode is selected.