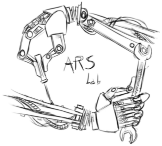
The Timers of the STM32 Microcontrollers

Corrado Santoro

ARSLAB - Autonomous and Robotic Systems Laboratory
Dipartimento di Matematica e Informatica - Università di Catania, Italy
`santoro@dmi.unict.it`



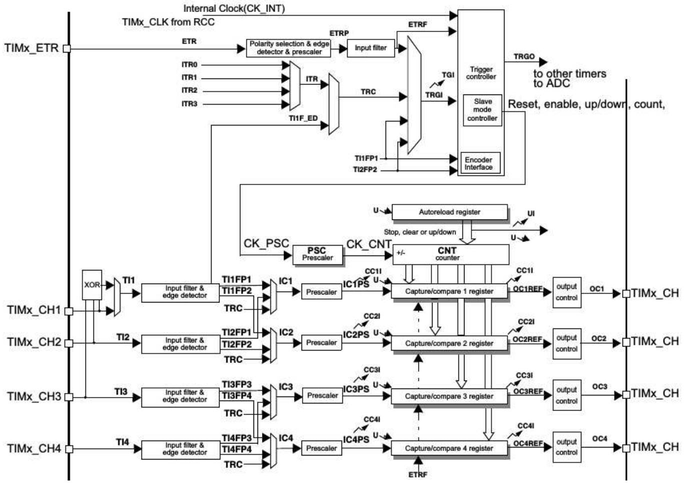
L.A.P. 1 Course

The Timers of the STM32 MCUs

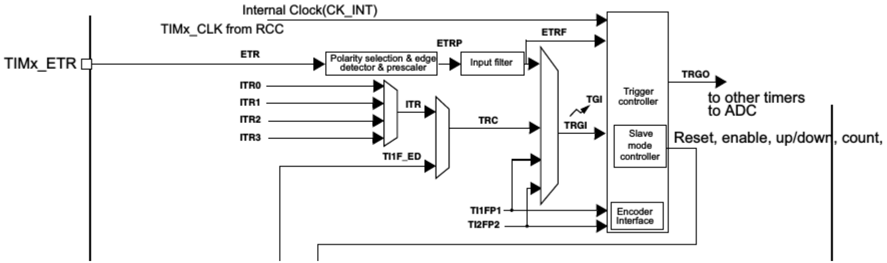
STM32 MCUs offer up to 11 different **timer/counters** with the following features:

- Clock selection (internal, external, other)
- 16/32-bit counter resolution
- Programmable prescaler
- Four independent channels configurable as:
  - Input Capture
  - Output Compare
  - PWM Mode
  - One-pulse Output
- Interrupt generation on the basis of the various events that can occur

Block Schematics of the Timers



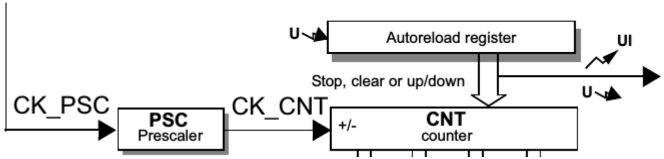
Timer Clock Source



Clock source can be:

- Internal (System Peripheral Clock)
- External (External Pin)
- External in QEI mode (Quadrature-encoder interface)
- Several Gate/Trigger inputs can be configured in order to start/stop the clock on the basis of events

Time-Base Part



Counting is handled in the time-base by the following registers:

- `TIMx->PSC`: the prescaler register, it directly specified the **division factor**
- `TIMx->CNT`: the counter register, it holds the counter value and increments according to the input clock
- `TIMx->ARR`: the auto-reload register, `CNT` counts from 0 to `ARR`, then `CNT` is set to 0 again

HAL Functions for Clock Source and Time Base

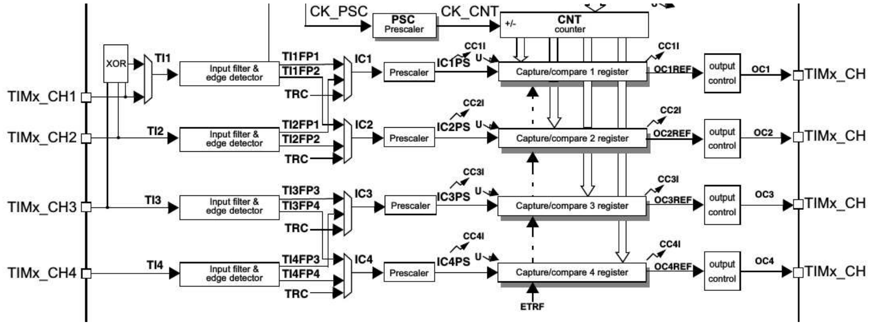
- `HAL_StatusTypeDef HAL_TIM_Base_Init(TIM_HandleTypeDef *htim);`
  Time-base initialization (counting mode, prescaler, auto-reload)

- `HAL_StatusTypeDef HAL_TIM_ConfigClockSource(TIM_HandleTypeDef *htim, TIM_ClockConfigTypeDef * sClockSourceConfig);`
  Clock source configuration

- `HAL_StatusTypeDef HAL_TIM_Base_Start(TIM_HandleTypeDef *htim);`
  Timer start

- `HAL_StatusTypeDef HAL_TIM_Base_Stop(TIM_HandleTypeDef *htim);`
  Timer stop

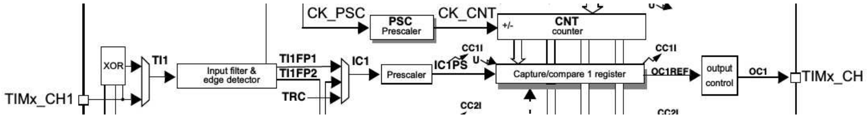
Timer Channels



Each timer can drive up to four different channels that can be configured as:

- Input Capture
- Output Compare
- PWM Mode
- One-pulse Output
- Each channel has a specific register `CCRy`

Input Capture

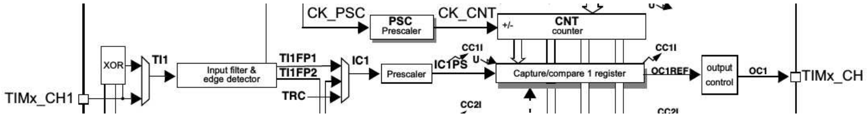


- An external pin is used as event source
- When an **edge** is detected, the value of **`CNT`** is stored into **`CCRy`**
- A bit in a flag register is set
- An (optional) interrupt is then generated
- The edge type can be programmed
- The circuit can also handle both edges in order to perform pulse width measurement

HAL Functions for Input Capture

- `HAL_StatusTypeDef HAL_TIM_IC_Init(TIM_HandleTypeDef *htim);`
  Input capture circuit initialization
- `HAL_StatusTypeDef HAL_TIM_IC_ConfigChannel(TIM_HandleTypeDef *htim, TIM_IC_InitTypeDef* sConfig, uint32_t Channel);`
  Capture channel configuration
- `HAL_StatusTypeDef HAL_TIM_IC_Start(TIM_HandleTypeDef *htim, uint32_t Channel);`
  Capture start
- `HAL_StatusTypeDef HAL_TIM_IC_Stop(TIM_HandleTypeDef *htim, uint32_t Channel);`
  Capture stop

Output Compare

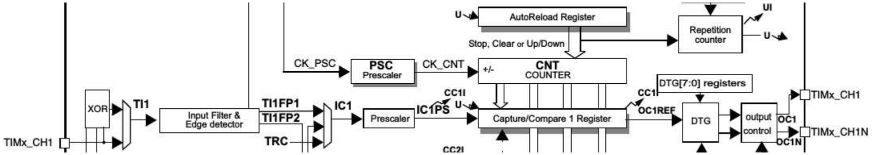


- An external pin is used as output
- When **`CNT`** reaches the value stored into **`CCRy`** a *compare event* is generated and, on the basis of a configuration, ...
- the output is **set**, or
- the output is **reset**, or
- the output is **toggled**

HAL Functions for Output Compare

- `HAL_StatusTypeDef HAL_TIM_OC_Init(TIM_HandleTypeDef *htim);`
  Output compare circuit initialization
- `HAL_StatusTypeDef HAL_TIM_OC_ConfigChannel(TIM_HandleTypeDef *htim, TIM_OC_InitTypeDef* sConfig, uint32_t Channel);`
  Compare channel configuration
- `HAL_StatusTypeDef HAL_TIM_OC_Start(TIM_HandleTypeDef *htim, uint32_t Channel);`
  Compare start
- `HAL_StatusTypeDef HAL_TIM_OC_Stop(TIM_HandleTypeDef *htim, uint32_t Channel);`
  Compare stop

PWM Mode



- An external pin is used as output
- The PWM signal is generated at that external pin
- **`ARR`** specifies the PWM period (in count units)
- **`CCRy`** specifies the PWM duty-cycle (in count units)

HAL Functions for PWM Mode

- `HAL_StatusTypeDef HAL_TIM_PWM_Init(TIM_HandleTypeDef *htim);`
  PWM initialization
- `HAL_StatusTypeDef HAL_TIM_PWM_ConfigChannel(TIM_HandleTypeDef *htim, TIM_OC_InitTypeDef* sConfig, uint32_t Channel);`
  PWM channel configuration
- `HAL_StatusTypeDef HAL_TIM_PWM_Start(TIM_HandleTypeDef *htim, uint32_t Channel);`
  PWM start
- `HAL_StatusTypeDef HAL_TIM_PWM_Stop(TIM_HandleTypeDef *htim, uint32_t Channel);`
  PWM stop

The Timers of the STM32 Microcontrollers

Corrado Santoro

ARSLAB - Autonomous and Robotic Systems Laboratory
Dipartimento di Matematica e Informatica - Università di Catania, Italy
`santoro@dmi.unict.it`

L.A.P. 1 Course