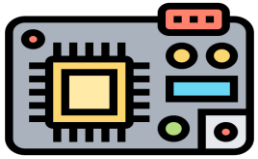
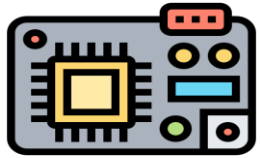


General Purpose Timer Module (GPTM)

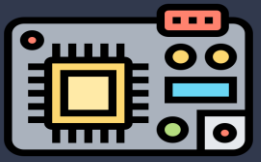
CSE 3105 - Computer Interfacing & Embedded System

Md. Nasif Osman Khansur
Lecturer, Dept. of CSE
Rajshahi University of Engineering & Technology, Rajshahi - 6204

Learning Objectives

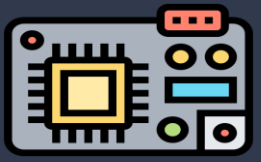


- To cover the main features of timer which are useful in handling any time related events.
- To learn how timer participate in generating waveforms and measuring the time characteristics of input signals.



Overview of Timers

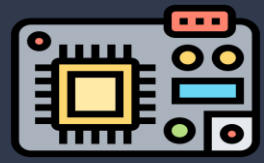
- STM32 embeds multiple timers providing timing resources for software and hardware tasks.
- Software tasks include time bases timeout event generation, and time triggers. (**Internal**)
- Hardware tasks are related to iOS, the timers can generate waveforms (PWM) on their outputs, measure the incoming signal parameters (frequency or timing), and react to external events on their inputs. (**External**)



Benefits of Timers

- STM32 Timers are very versatile and provide multiple operating modes to offload the CPU from repetitive and time-critical tasks while minimizing the interfacing circuitry needs.
- They are based on the same scalable architecture *(Once the timer operating principles are known, they are valid for any of the timers).*
- They are fully featured for motor control and digital power conversion applications.
- **Multiple timers can be linked and synchronized.**

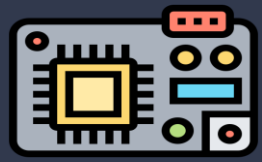
Control Register



- (GPTM) control register can vary depending on the specific microcontroller architecture or family. In ARM Cortex-M microcontrollers, such as those from Texas Instruments or STMicroelectronics, typically have a size of 32 bits.

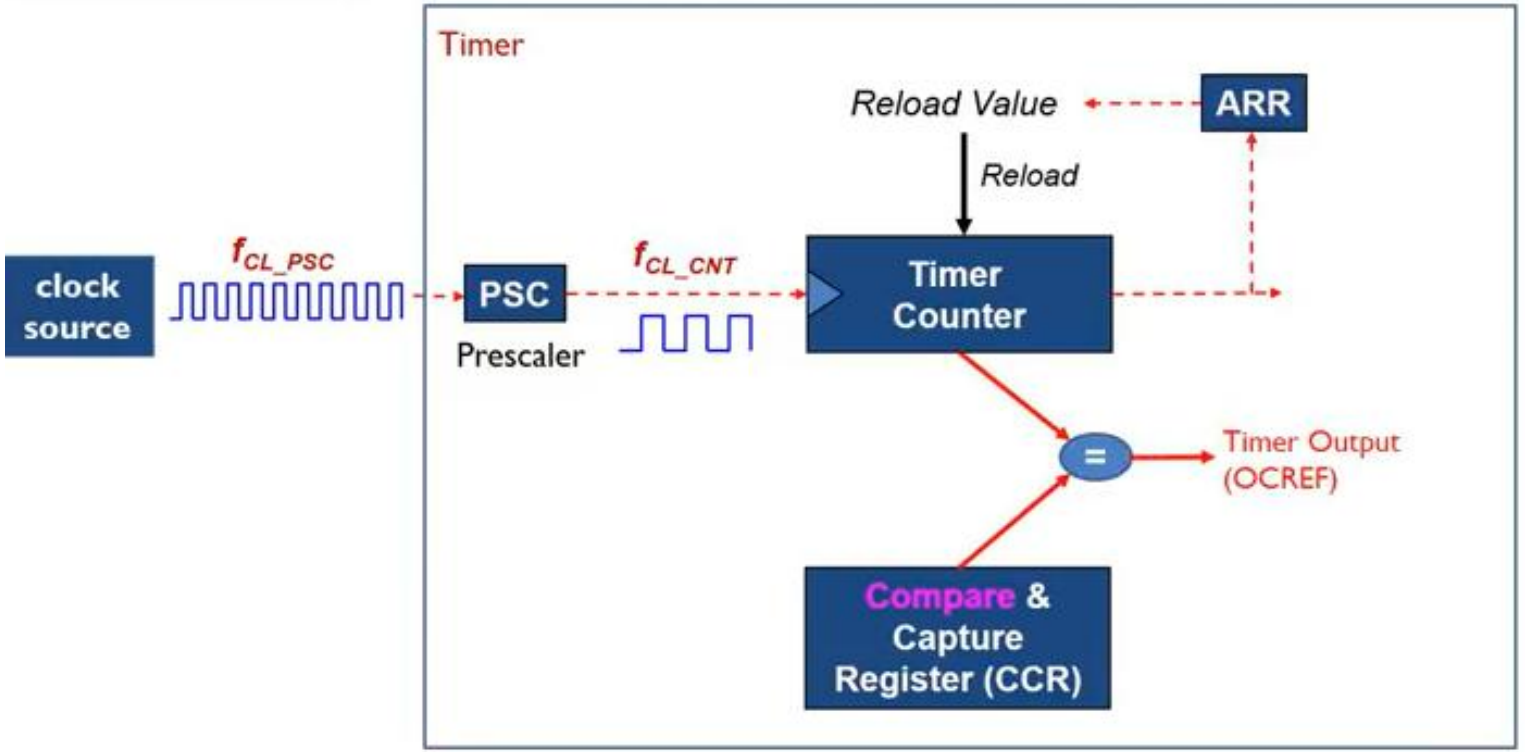
Bit	Name	Description
0	TIMER_EN	Timer Enable
1	TIMER_MODE	Timer Mode (One-shot/Periodic)
2	COUNT_DIR	Counting Direction (Up/Down)
3	INT_ENABLE	Interrupt Enable
4	UPD_MODE	Update Mode
5	PWM_ENABLE	PWM Enable
6-31	Reserved/Other	Additional configuration options

Control Register

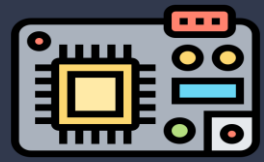


- Timer Enable (TIMER_EN): This bit enables or disables the timer. Setting it to 1 starts the timer, while 0 stops it.
- Timer Mode: (0 = One-shot, 1 = Periodic)
- Counting Direction (0 = Count Up, 1 = Count Down)
- Interrupt Enable (1 = Enable, 0 = Disable timer interrupts)
- Update Mode: Configures whether the timer updates its count on a specific event (0 = like a register read/write, 1 = a specific timer event).
- PWM Enable : 1 = Enable, 0 = Disable
- Capture/Compare Functionality: Additional fields may exist for configuring capture and compare behavior in timer modules designed for PWM or signal capture applications.

Block Diagram

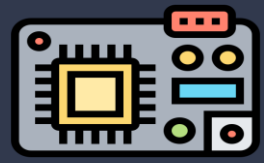


Components



- ❑ **Clock Source:** The clock source for a timer refers to the signal that drives the counting operation of the timer. It is essentially the frequency at which the timer operates and increments or decrements its counter.
- ★ Clock Source Options:
  - System Clock (e.g., HCLK or PCLK)
  - Prescaled Clock (a divided-down version of the system clock using a prescaler)
  - External Pin (input from an external signal)
  - Internal Oscillator (such as LSI or LSE for low-speed timers)

Components

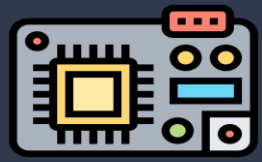


- Timer Counter: It runs freely. If enabled, it automatically counts upwards or downwards, under the driven of a clock source. It's a hardware which, for every rising edge in the clock signal, increments or decrements the counter by **one** automatically. Hence, it's known as *free-running* counter.
- Prescaler (PSC) Register: It holds the frequency prescaler value or the frequency divider. By modifying the PSC we can feed the counter with the clock pulses at the rate we desire. Before the input clock is fed to the counter, the frequency of the clock source is divided by a constant integer which is PSC + 1. It enables a tradeoff between timer resolution and timer range.

$$f_{CK\_CNT} = \frac{f_{CL\_PSC}}{PSC + 1}$$

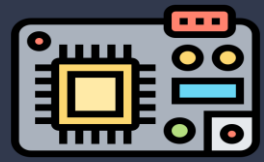
"High timer resolution requires a high clock rate. But it may cause the timer to overflow or underflow more quickly."

Components



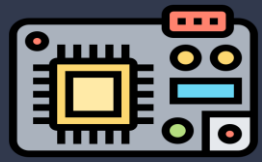
- Compare and Capture Register (CCR): When the timer is configured to generate an input signal, hardware constantly compares the free-running counter with a value stored in a special register called CCR. **The timer output, OCREF can be high or low, depending on the timer settings.**
  - When a timer is used to generate an output, the CCR is used only for compare.
  - When a specific external event occurs, such as the rising edge of an external signal, the CCR can be used for input capture.

Components



- Auto Reload Register (ARR): The Auto-Reload Register (ARR) is a critical register in the GPTM that sets the period of the timer. It defines the value at which the timer's counter will reset or wrap around, depending on the mode of the timer (e.g., up-counting, down-counting, or center-aligned counting).
  - For example, if the counter is in up-counting mode, it counts from zero to the value in ARR. Once the ARR value is reached, the counter resets to zero and starts counting again.

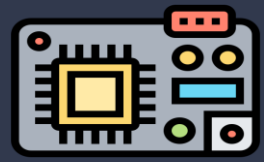
Components



- Repetition Counter Register (RCR): RCR defines how many times the timer counter should overflow before generating an event, like an interrupt or output compare event.
- The timer's event frequency or period can be calculated using the formula:

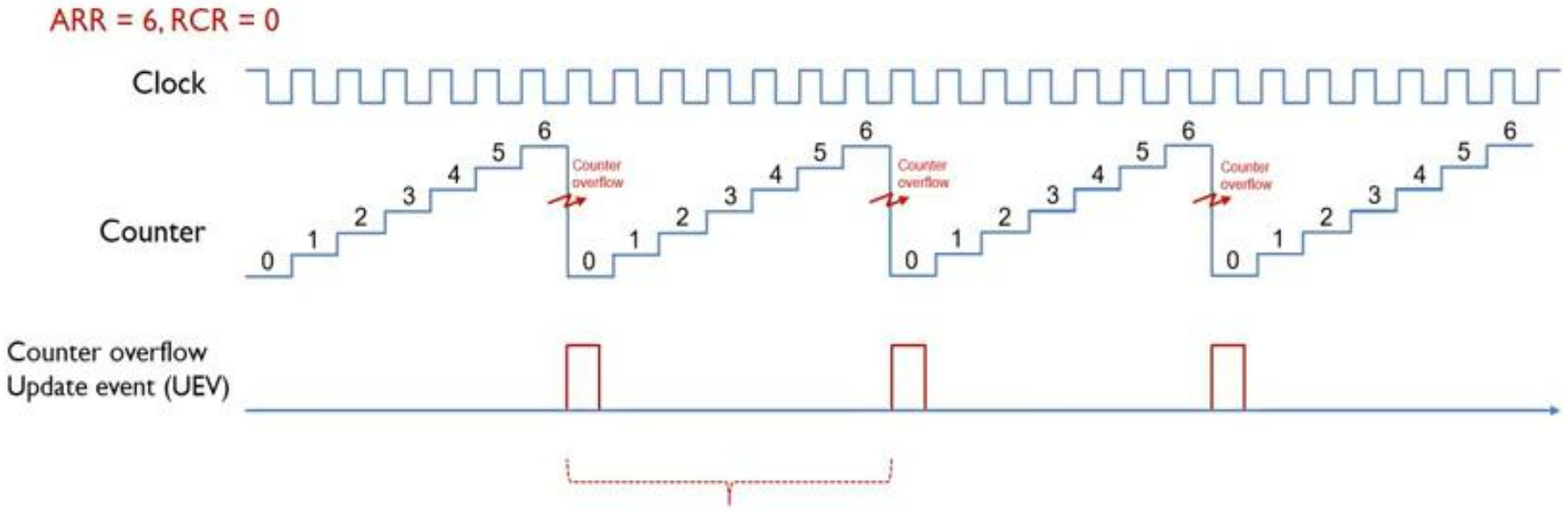
$$\text{Timer Frequency} = \frac{\text{Clock Frequency}}{(\text{Prescaler} + 1) \times (\text{ARR} + 1)}$$

Up Counting Mode



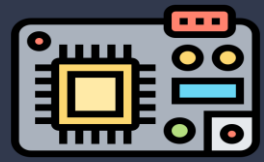
- In **up-counting mode**, the timer starts counting from 0 and increments by 1 on each clock cycle. The counter continues counting upward until it reaches the value in the ARR register. Once it reaches the ARR value, the counter resets to 0 and starts counting again. Say, ARR =6, RCR =0.
  - ARR = 6 means the counter will count from 0 to 6.
  - RCR = 0 means there is no repetition; the timer generates an update event after every complete cycle (from 0 to ARR).

Up Counting Mode



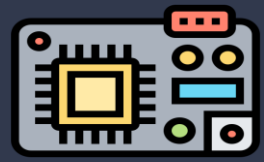

```
Period = (1 + ARR) * Clock Period
       = 7 * Clock Period
```

Steps in Up Counting Mode



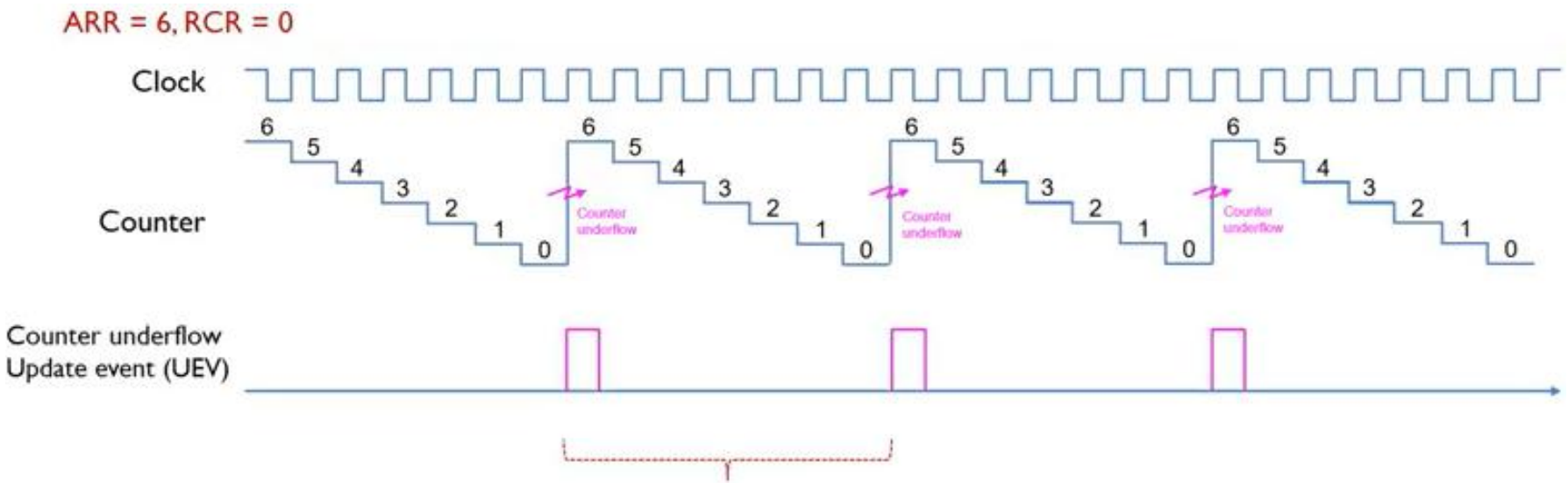
- **Starting at 0**: The counter starts at 0.
- **Incrementing**: The counter increments by 1 on each timer clock pulse: 0, 1, 2, 3, 4, 5, 6.
- **Reaching ARR (6)**: Once the counter reaches 6, it hits the value stored in the ARR register.
- **Overflow (Update Event)**: When the counter hits ARR (6), it triggers an update event (if enabled). Since RCR = 0, the update event is generated immediately. A new period starts. The counting period is determined by ARR value and clock period.
- **Resetting to 0**: After the update event, the counter resets to 0 and starts counting again.

Down Counting Mode



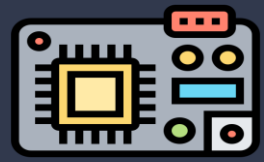
- In **down-counting mode**, the timer starts counting from the value in the ARR (Auto-Reload Register) and decrements down to 0. Once it reaches 0, it can trigger an event, such as an interrupt, and then either restart the count (if configured) or stop, depending on the settings of the timer. Say, ARR =6, RCR =0.
  - ARR = 6 means the counter will count from 6 down to 0.
  - RCR = 0 means there is no repetition; the timer generates an update event after every complete cycle (from 0 to ARR).

Down Counting Mode



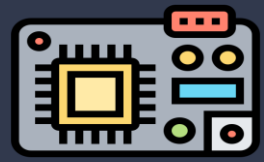
$$\text{Period} = (1 + \text{ARR}) * \text{Clock Period}$$
$$= 7 * \text{Clock Period}$$

Steps in Down Counting Mode



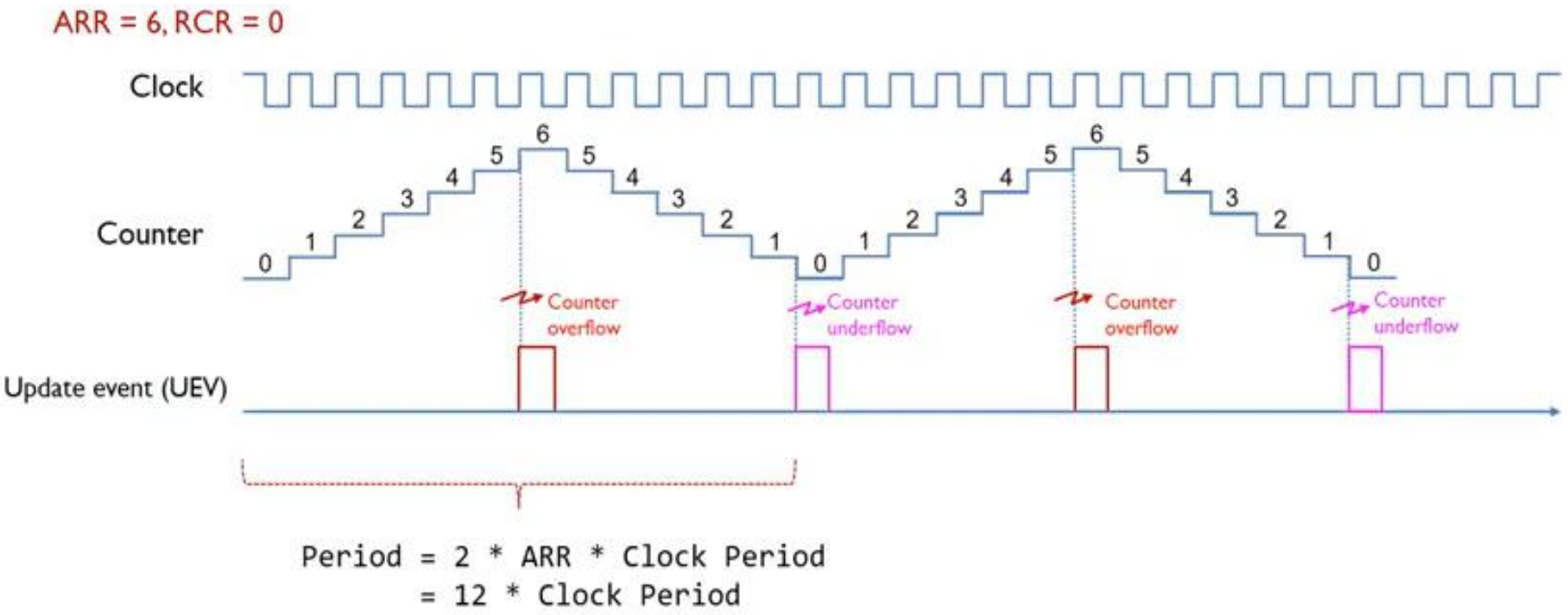
- The timer starts counting from 6 (the value in ARR).
- It decrements down to 0 in discrete steps.
- Once it reaches 0, an update event occurs (and an interrupt can be triggered if enabled).
- Since RCR = 0, the counting process will not repeat. The timer will stop after one down-counting cycle.

Center-aligned Mode



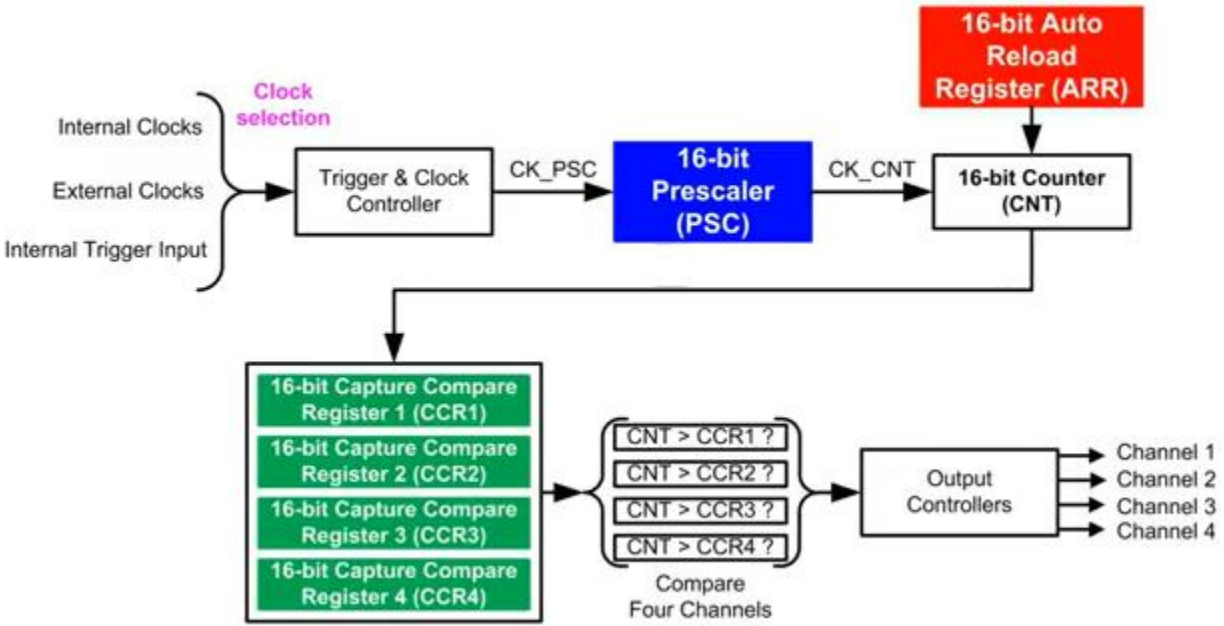
- In **center-aligned mode** (PWM mode 1 or 2), the timer counts up from 0 to ARR, then down back to 0, and this pattern repeats. In this mode, a symmetrical triangular counting behavior can be observed.
  - As ARR = 6, the counter will:
    - Count from **0 → 6 (up)**, then
    - Count from **6 → 0 (down)**, continuously repeating this cycle.
    - **(ARR + ARR) = 6 + 6 = 12** clock ticks for one complete up-down cycle.

Center-aligned Mode



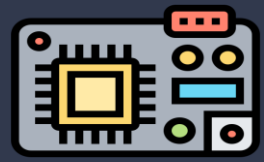
Multi-channel Outputs

- Typically, a timer can generate up to 4 PWM signals with independent duty cycles and identical frequency. Each timer has 4 channels and each channel has its own CCR.



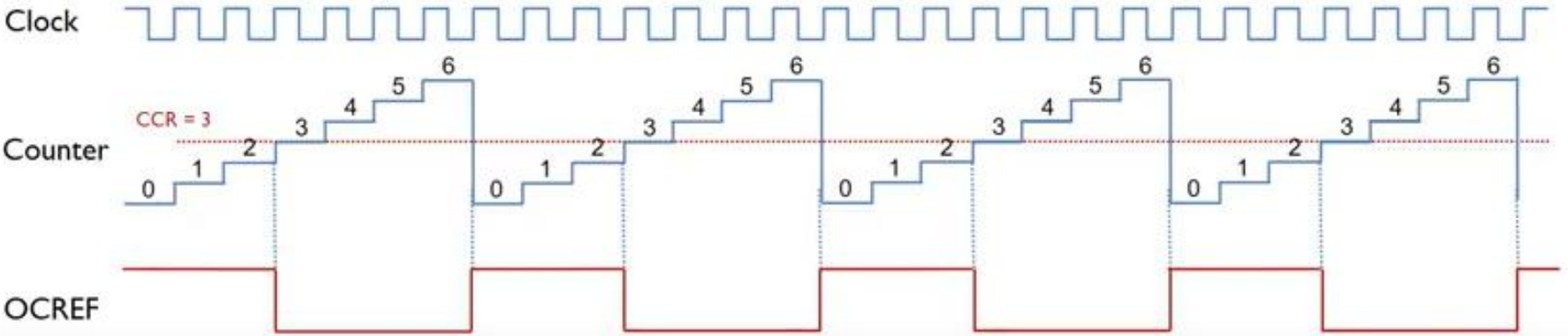
The timer counter (CNT) has 16 bits. The CCR holds the value that is compared with the timer counter. The four output channels share the same free-run timer counter.

PWM Mode 1 (Low-True)



Mode 1
Timer Output =
- High if counter < CCR
- Low if counter ≥ CCR

Upcounting mode, ARR = 6, CCR = 3, RCR = 0



$$\text{Duty Cycle} = \frac{CCR}{ARR + 1} = \frac{3}{7}$$

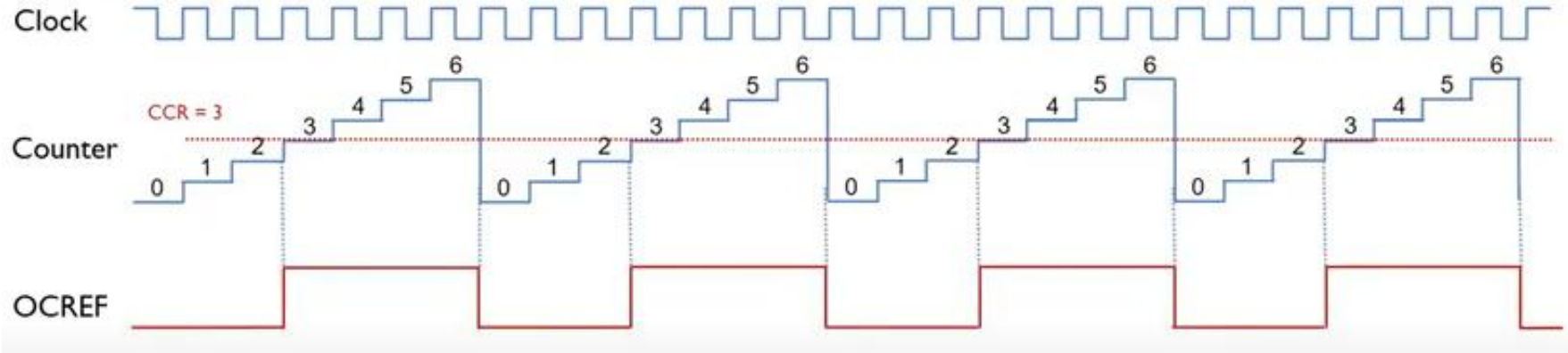
$$\text{Period} = (1 + ARR) * \text{Clock Period} = 7 * \text{Clock Period}$$

PWM Mode 2 (High-True)

Mode 2

Timer Output =
- Low if counter < CCR
- High if counter ≥ CCR

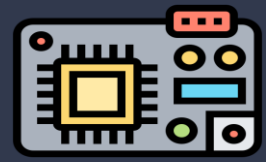
Upcounting mode, ARR = 6, CCR = 3, RCR = 0



$$\text{Duty Cycle} = 1 - \frac{CCR}{ARR + 1} = \frac{4}{7}$$

$$\text{Period} = (1 + ARR) * \text{Clock Period} = 7 * \text{Clock Period}$$

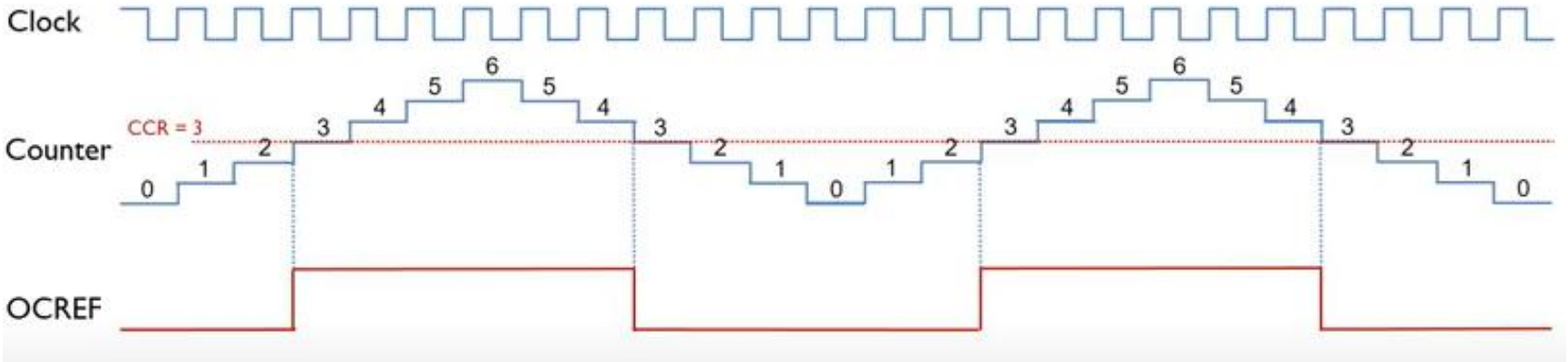
PWM Mode 2 (High-True)



Mode 2

Timer Output =
- Low if counter < CCR
- High if counter ≥ CCR

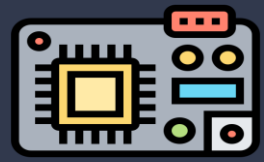
Center-aligned mode, ARR = 6, CCR = 3, RCR = 0



$$\text{Duty Cycle} = 1 - \frac{CCR}{ARR} = \frac{1}{2}$$

$$\text{Period} = 2 * ARR * \text{Clock Period} = 12 * \text{Clock Period}$$

PWM Output Polarity



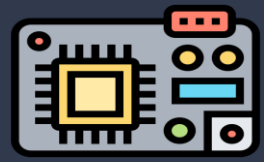
Mode	Counter < CCR	Counter ≥ CCR
PWM mode 1 (Low True)	Active	Inactive
PWM mode 2 (High True)	Inactive	Active

Output Polarity:

- Software can program the CCxP bit in the TIMx_CCER register

	Active	Inactive
Active High	High Voltage	Low Voltage
Active Low	Low Voltage	High Voltage

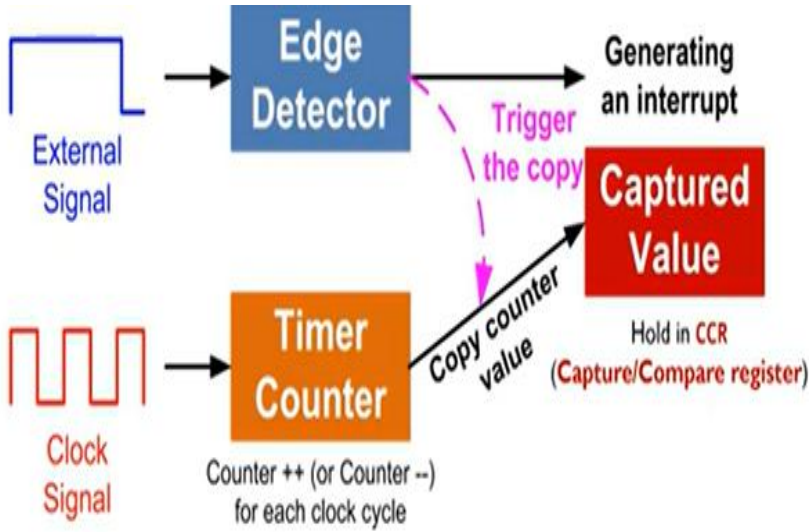
Input Capture



- Input Capture is a feature commonly found in microcontroller timers that allows the timer to record (capture) the value of its counter when a specific external event occurs on a pin.
- Capture the timestamp of the occurrence of an event.
- Event is represented as rising edge or falling edge in a digital signal.

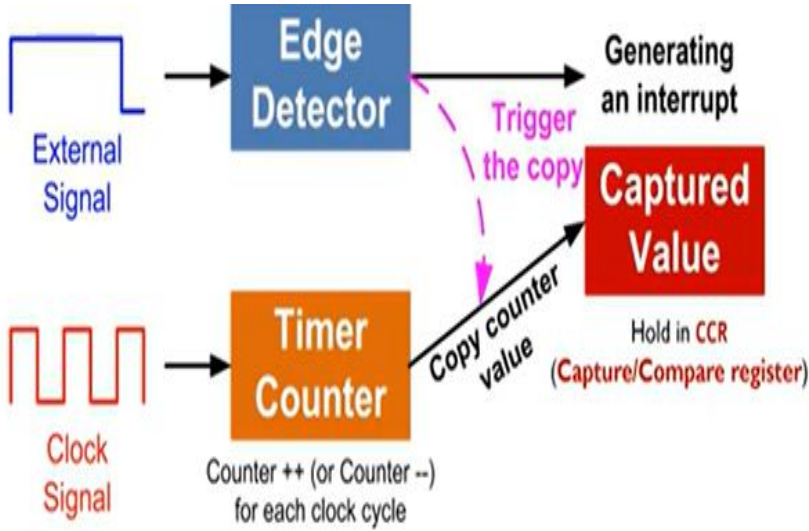
Input Capture Process

- External Signal: This is the input signal (a waveform) whose event timing needs to be captured. The signal can have rising or falling edges.
- Edge Detector: Monitors the input signal and detects edges (rising, falling, or both). When an edge is detected, it triggers a capture event.
- Timer Counter: This is an ongoing counter that increments (or decrements) with each cycle of the clock signal. The counter keeps track of elapsed time.

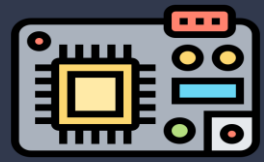


Input Capture Process

- **Captured Value:** When an edge is detected, the edge detector triggers the copying of the current timer counter value into the Capture/ Compare Register (CCR). This register stores the exact time (in terms of clock cycles) when the edge was detected.
- **Interrupt Generation:** After the capture event, an interrupt is generated to notify the processor that a capture has occurred. The system can then read the captured value for further processing.



References



Ref. Book: Embedded Systems with ARM Cortex-M Microcontrollers in Assembly Language and C 3rd Ed - Yifeng Zhu - Eman (2018) **[Chapter 15]**


Thank
you