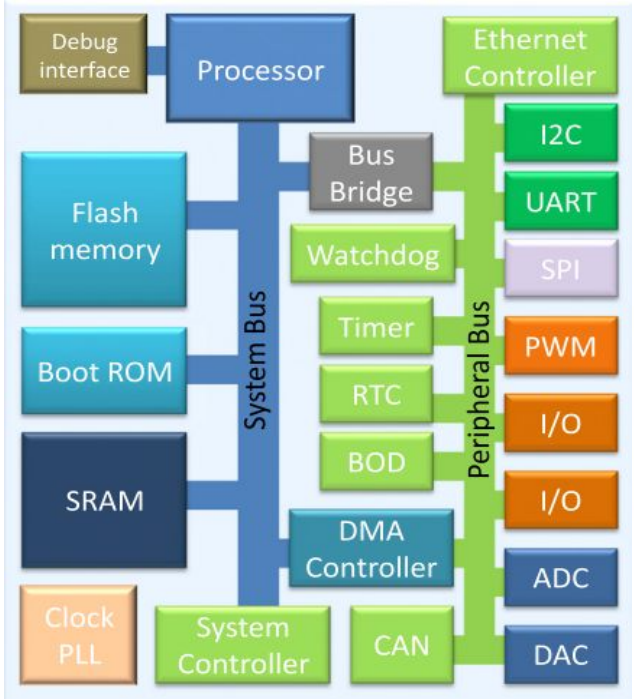
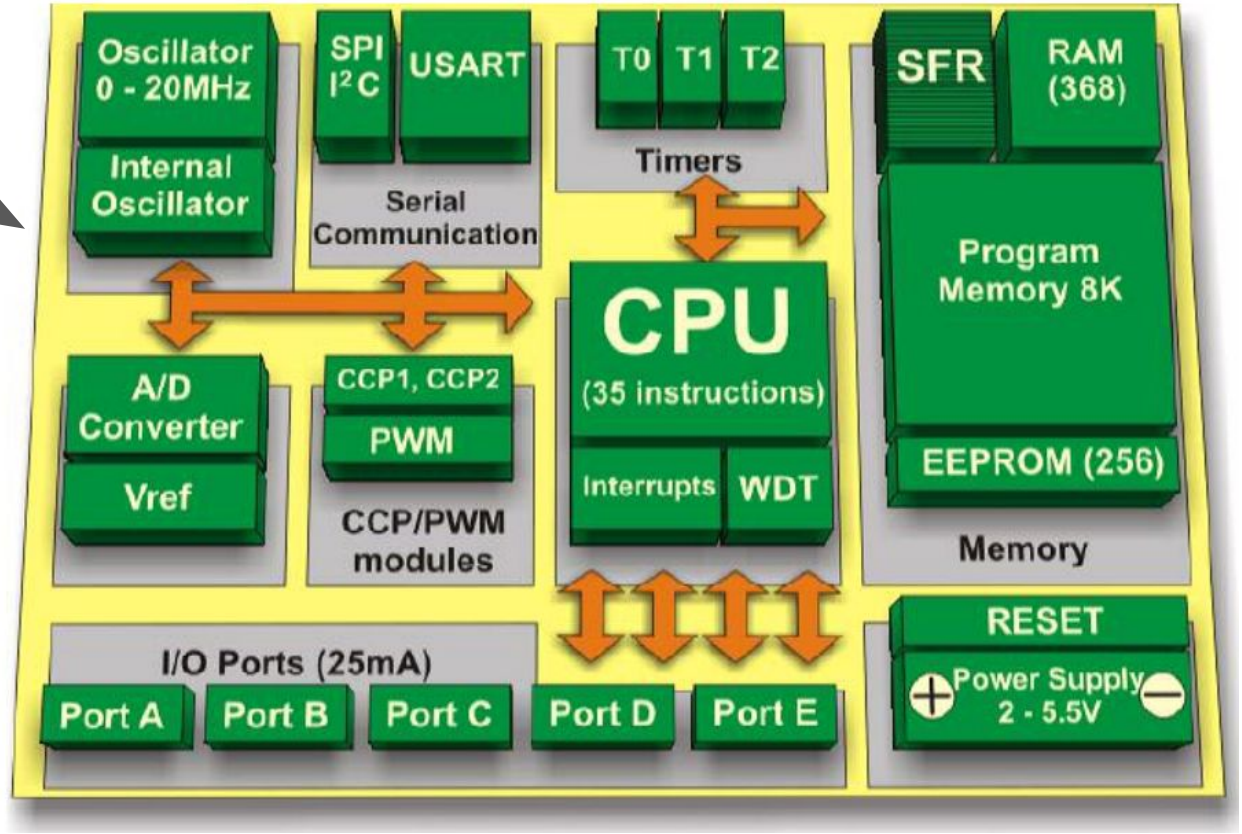
Microcontroller

~Tapadyuti Baral

Topics :

Clock & Timer

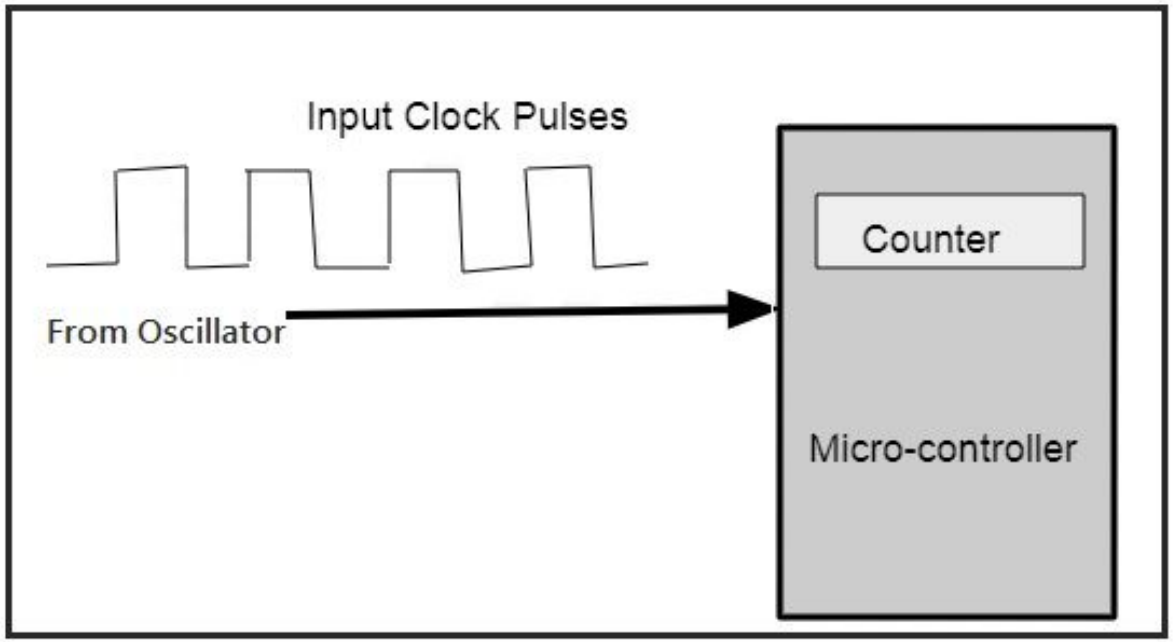


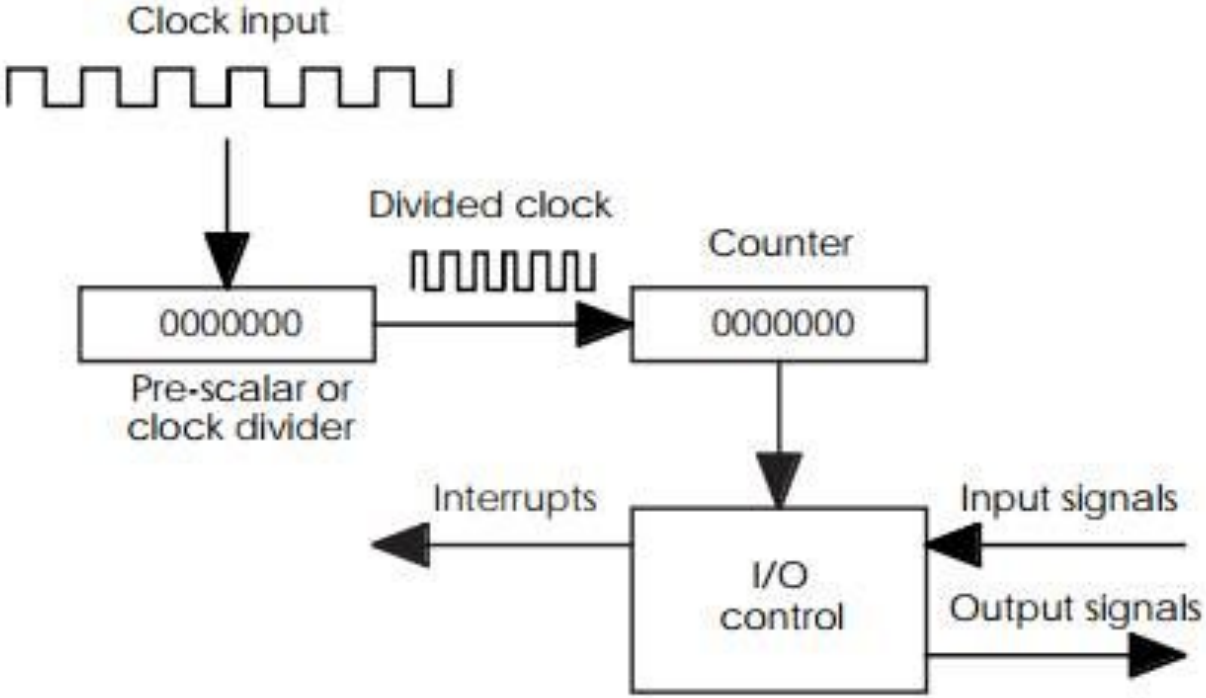


Ref Diagram - PIC

What is Clock

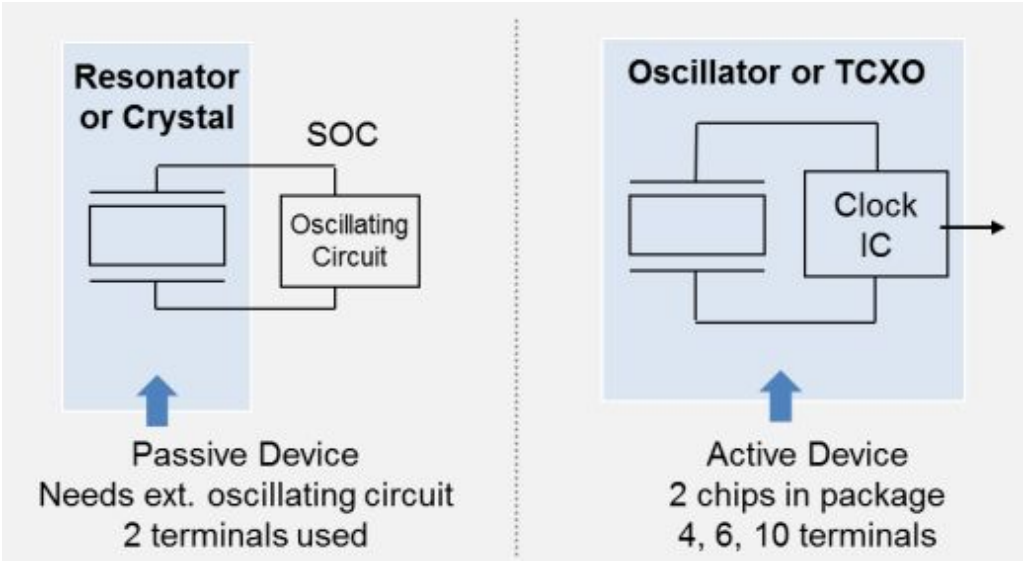
- Track time, date, month & Year
- Maintain the System Clock

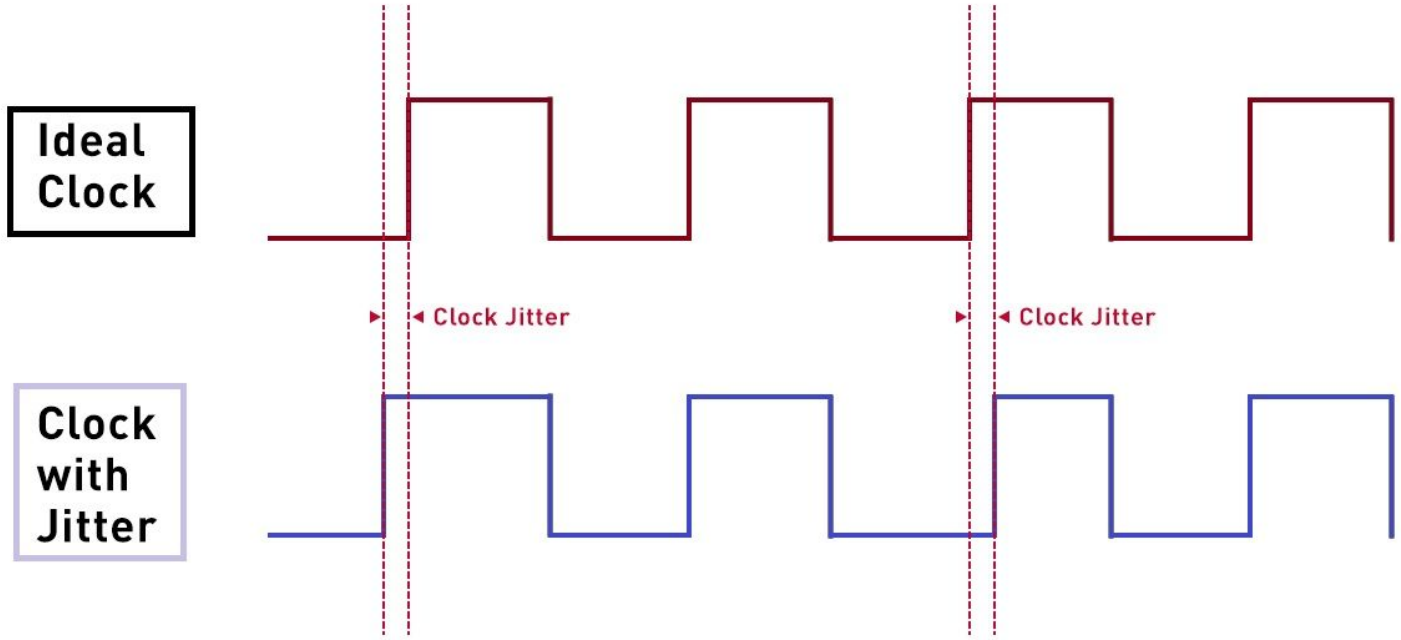


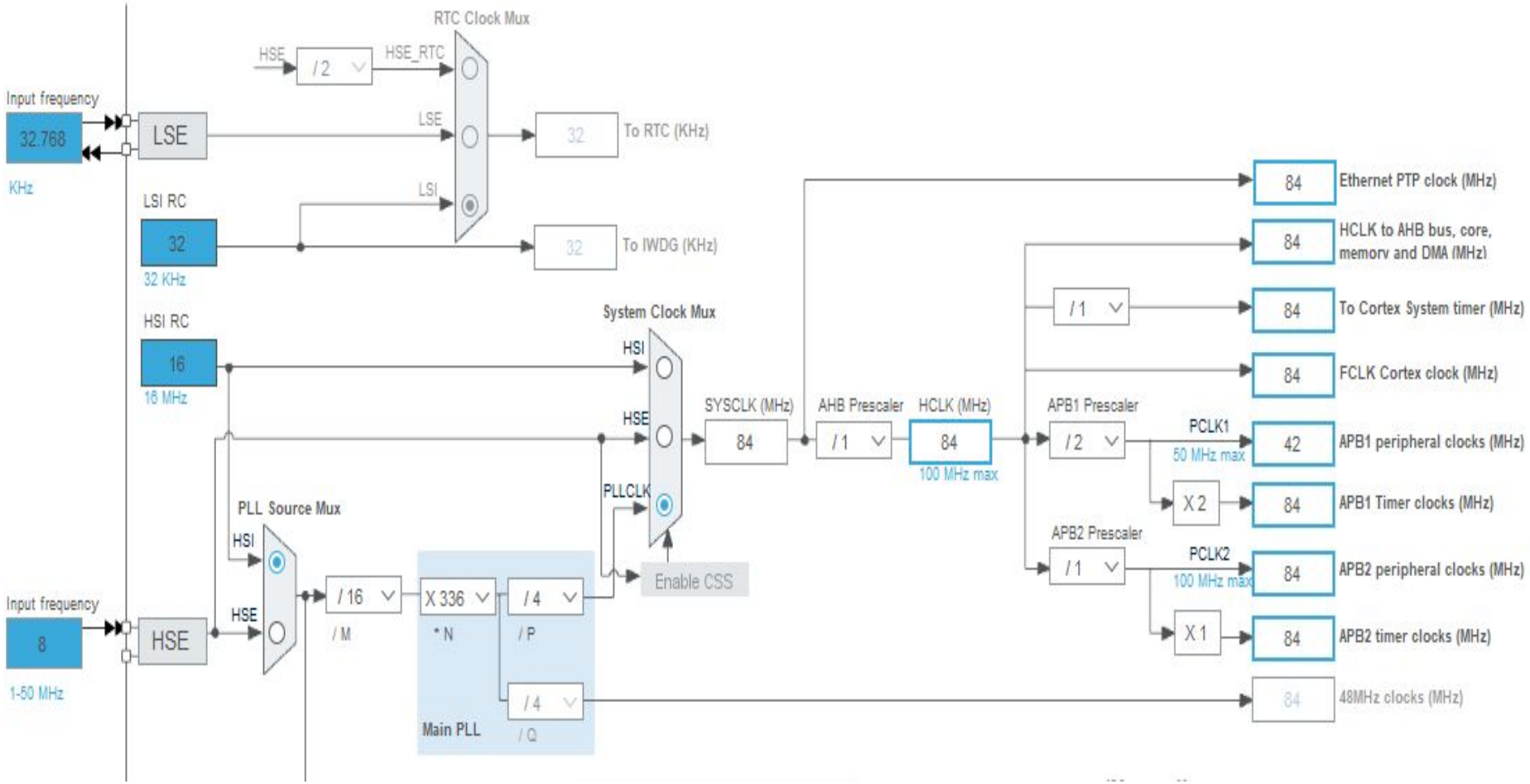


Generic timer/counter

- Data Sampling
- Synchronous & Asynchronous
- Internal vs External oscillators
- Ceramic Resonator vs Quartz Crystal




Ideal Clock
Clock Jitter
Clock Jitter
Clock with Jitter


RTC Clock Mux
HSE
/ 2
HSE_RTC
Input frequency
32.768
LSE
LSE
32
To RTC (KHz)
KHz
LSI RC
32
LSI
32
To IWDG (KHz)
32 KHz
HSI RC
16
16 MHz
System Clock Mux
HSI
HSE
PLLCLK
SYSCLK (MHz)
84
AHB Prescaler
/ 1
HCLK (MHz)
84
100 MHz max
Enable CSS
PLL Source Mux
HSI
HSE
Input frequency
8
1-50 MHz
HSE
/ 16
/ M
X 336
* N
/ 4
/ P
/ 4
/ Q
Main PLL
84
Ethernet PTP clock (MHz)
84
HCLK to AHB bus, core, memory and DMA (MHz)
/ 1
84
To Cortex System timer (MHz)
84
FCLK Cortex clock (MHz)
APB1 Prescaler
/ 2
PCLK1
50 MHz max
42
APB1 peripheral clocks (MHz)
X 2
84
APB1 Timer clocks (MHz)
APB2 Prescaler
/ 1
PCLK2
100 MHz max
84
APB2 peripheral clocks (MHz)
X 1
84
APB2 timer clocks (MHz)
84
48MHz clocks (MHz)

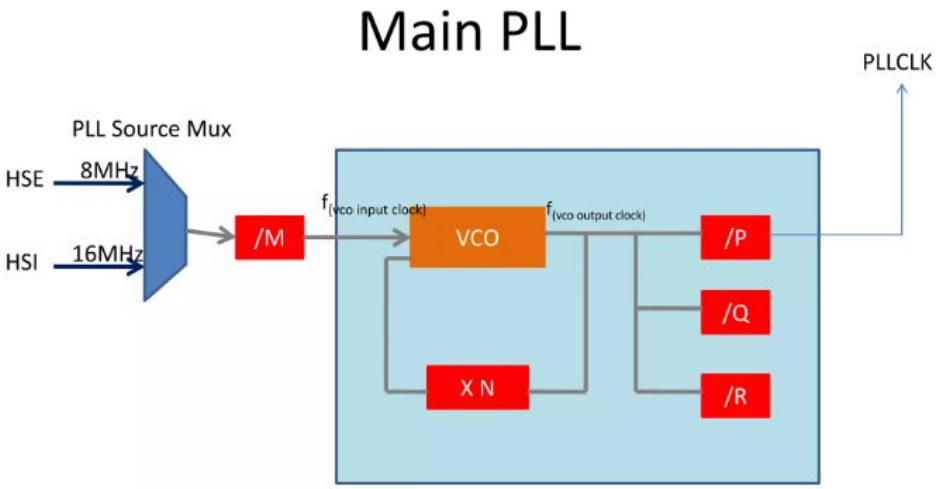
System Clock(SYSCLK)

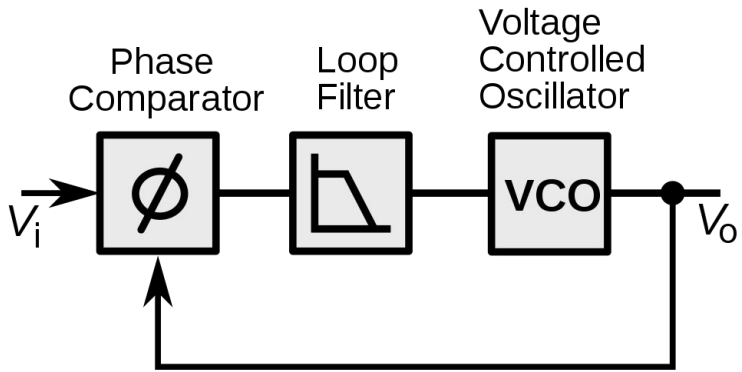
- Three different clock sources can be used to drive the system clock (SYSCLK):
  - HSI oscillator clock
  - HSE oscillator clock
  - Two main PLL (PLL) clocks
- The devices have the two following secondary clock sources
  - 32 kHz low-speed internal RC (LSI RC) which drives the independent watchdog and, optionally, the RTC used for Auto-wakeup from the Stop/Standby mode.
  - 32.768 kHz low-speed external crystal (LSE crystal) which optionally drives the RTCclock (RTCCLK)

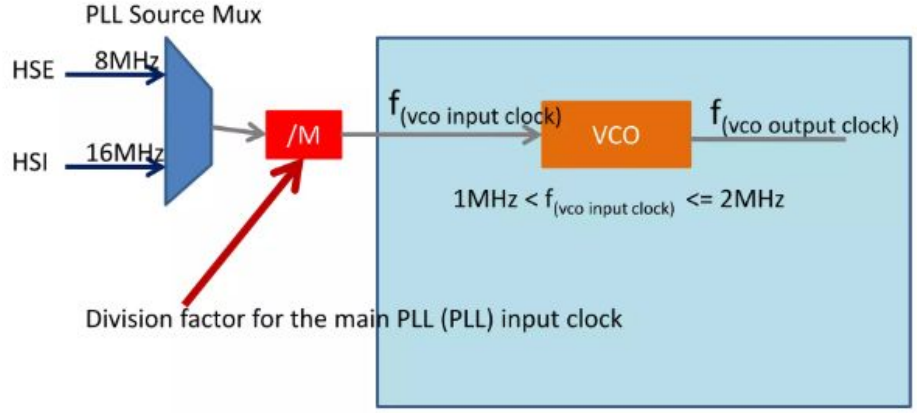
- HSI → 16MHz (Internal to MCU)
- HSE→ 8MHz (External to MCU)
- PLL can generate clock up to 180MHz (Internal to MCU)
- LSI→32kHz (Internal to MCU)
- LSE→32.768kHz (External to MCU)

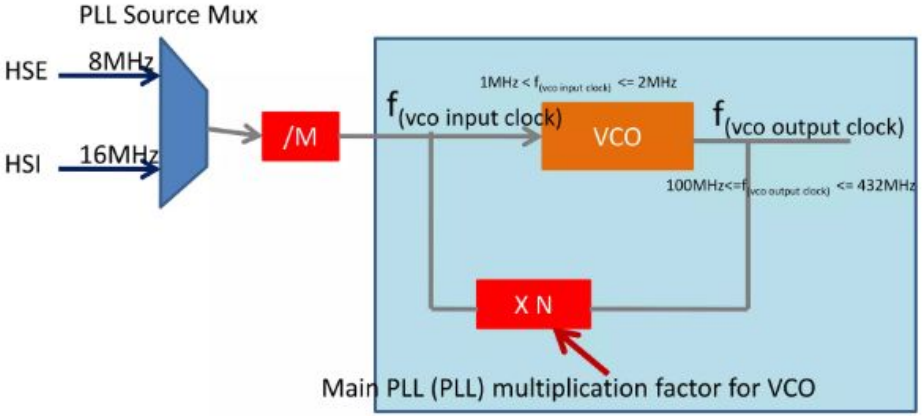
Phase Lock Loop

The oscillator's frequency and phase are controlled proportionally by an applied voltage, hence the term voltage-controlled oscillator (VCO). The oscillator generates a periodic signal of a specific frequency, and the phase detector compares the phase of that signal with the phase of the input periodic signal, to adjust the oscillator to keep the phases matched.









PLL Formulas

$$f_{(\text{vco output clock})} = \left[\frac{f_{(\text{vco input clock})}}{\text{PLLM}}\right] \times \text{PLLN}$$

$$f_{\substack{(\text{PLL general clock output}) \\ \textbf{PLLCLK}}} = \frac{f_{(\text{vco output clock})}}{\text{PLLP}}$$

Default Clock State

After reset of the MCU,

HSI is ON, HSE is OFF , PLL is OFF, LSE is OFF,

LSI is OFF

So, SYSCLK is sourced by HSI .

I.e : SYSCLK = 16MHz

Each clock source can be switched on or off independently when it is not used, to optimize power consumption.

HSI

✓The HSI is used (enabled by hardware) as system clock source after startup from Reset, wake-up from STOP and STANDBY mode, or in case of failure of the HSE used directly or indirectly as system clock

✓The HSI RC oscillator has the advantage of providing a clock source at low cost (no external components).

✓It also has a faster startup time than the HSE crystal oscillator.

✓However, even with calibration the frequency is less accurate than an external crystal oscillator or ceramic resonator

✓The HSI signal can also be used as a backup source (auxiliary clock) if the HSE crystal oscillator fails.

Whenever an interrupt happens, the processor stops the current code, and handle the interrupt by running an *Interrupt Service Routines (ISR)* which is located in a predefined table called *Vector Interrupt Table (VIC)*.

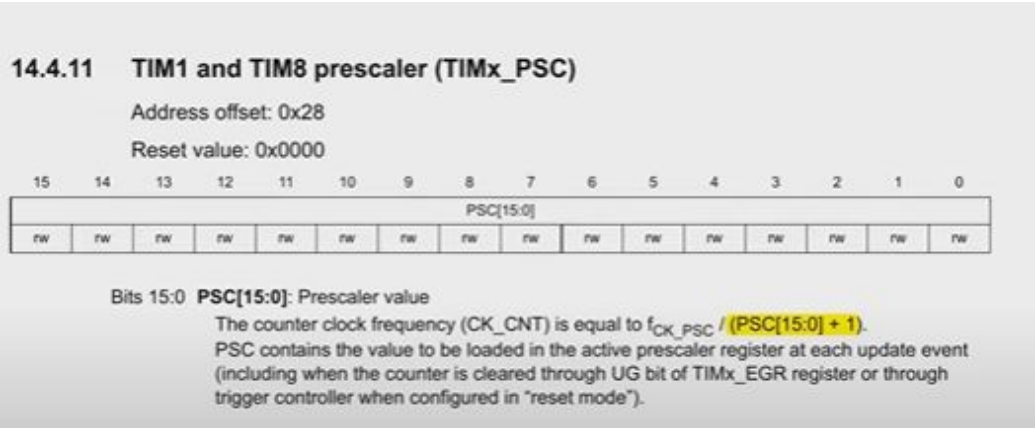
Rule of order of execution:

- Higher priority (as the same as lower number) runs first
- If the same priority in pending, the lowest exception number takes precedence when the processor is executing an exception handler, the exception handler is preempted if a higher priority exception occurs.
- If an exception occurs with the same priority as the exception being handled, the handler is not preempted, irrespective of the exception number. However, the status of the new interrupt changes to pending.

Prescaler is an electronic circuit used to reduce a high frequency electrical signal to a lower frequency. The prescaler takes the basic timer clock frequency (which may be the CPU clock frequency or may be some higher or lower frequency) and divides it by some value before feeding it to the timer, according to how the **prescaler register(s)** are configured.

- The purpose of the prescaler is to allow the timer to be clocked at the rate a user desires. For shorter (8 and 16-bit) timers, there will often be a tradeoff between resolution (high resolution requires a high clock rate) and range (high clock rates cause the timer to overflow more quickly).
- Timer Resolution
- Timer Range

Period



14.4.11 TIM1 and TIM8 prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (CK_CNT) is equal to f_{CK_PSC} / (PSC[15:0] + 1).

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in "reset mode").

The zero value of a digital timer is 0.00 seconds. What is the resolution of the timer? (A) 1.00 seconds, (B) 0.10 seconds, (C) 0.01 seconds.

$$\text{TIM CLOCK} = \frac{\text{APB TIM CLOCK}}{\text{PRESCALAR}}$$

$$\text{FREQUENCY} = \frac{\text{TIM CLOCK}}{\text{ARR}}$$

$$\text{DUTY \%} = \frac{\text{CCRx}}{\text{ARR}} \times 100$$

Counter is a device which stores (and sometimes displays) the number of times a particular event or process has occurred, often in relationship to a clock. The most common type is a sequential digital logic circuit with an input line called the *clock* and multiple output lines. The values on the output lines represent a number in the binary or BCD number system. Each pulse applied to the clock input increments or decrements the number in the counter.

A timer will tick (increment by one) each time it receives a clock pulse.