Embedded Systems

Timers, Capture/Compare/PWM, clock generator, watchdog

Lesson 07

Francesco Menichelli

francesco.menichelli@uniroma1.it

Why timers in MCUs

- Precise time delay in programs
- Periodic interrupt generation
- Time delay measurement
- Period and pulse width measurement
- Frequency measurement
- Event counting
- Arrival time comparison
- Time-of-day tracking
- Waveform generation

Input capture

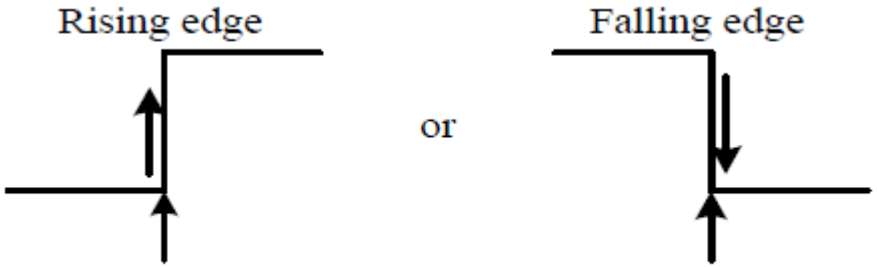
Real Time Clock (RTC)

Output compare/PWM

All these functions are difficult to do using only software operations

Input Capture Functions

- Physical time is often represented by the contents of the main timer.
- The occurrence of an event is represented by a signal edge (rising or falling edge).
- The time when an event occurs is recorded *by latching the count of the main timer* when a signal edge arrives
- Input capture channels share most of the circuit with output compare functions. For this reason, often they cannot be enabled simultaneously.



Events represented by signal edges

Applications of Input Capture Function

- Event arrival time recording
- Period measurement: need to capture the main timer values corresponding to two consecutive rising or falling edges

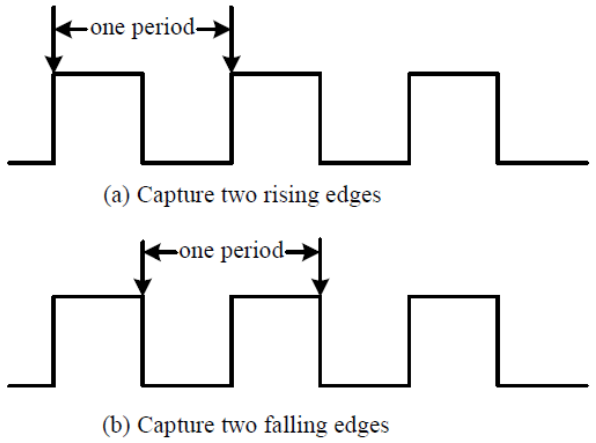


Figure 8.9 Period measurement by capturing two consecutive edges

- Pulse width measurement: need to capture the rising and falling edges

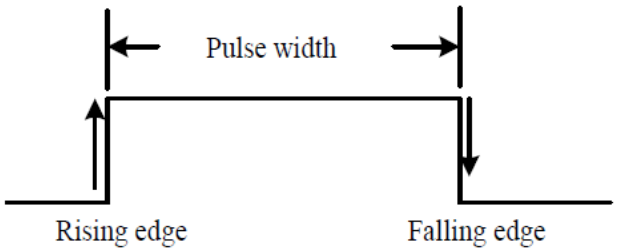


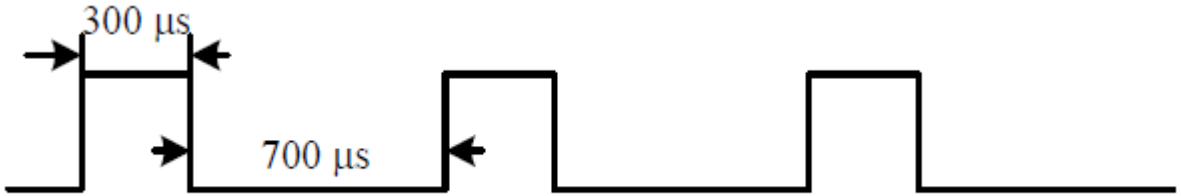
Figure 8.10 Pulse-width measurement using input capture

Output Compare Functions

- Trigger an action (event) at a specific time in the future
- The actions that can be activated on an output compare pin include
  - Set high
  - Set low
  - Toggle
- An interrupt may be optionally requested

Applications of Output Compare Function

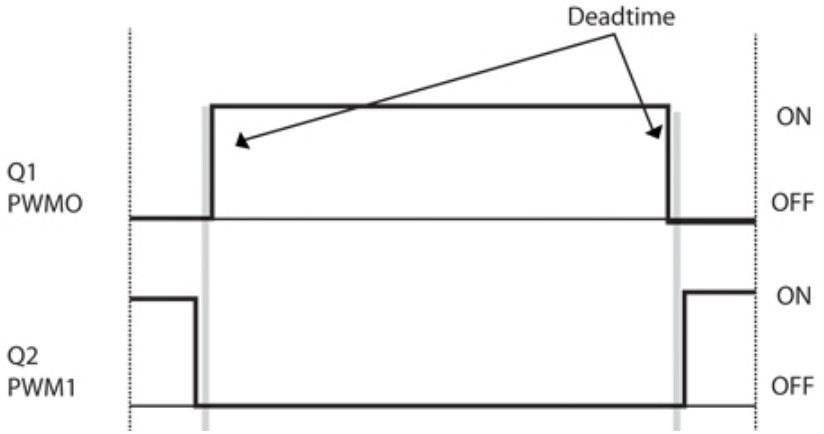
- Generate events (transitions) after a precise delay
- Generate precise pulses or pauses in a waveform
- e.g. Generate an active high 300us pulse and a 700us pause

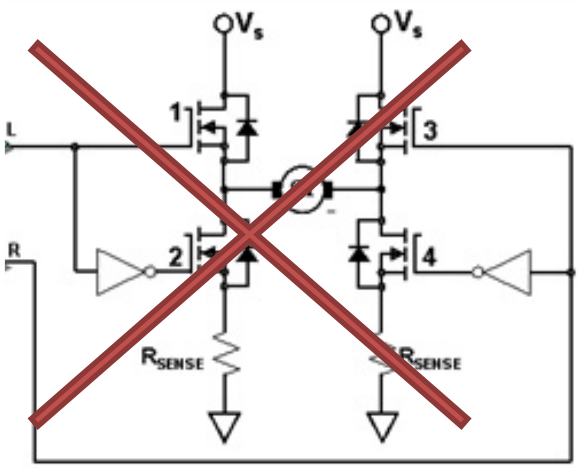


- For periodic waveform requires too much software computation. The preferred way is PWM function
- e.g. Generate an active high 1 KHz digital waveform with 30 percent duty cycle

PWM Functions

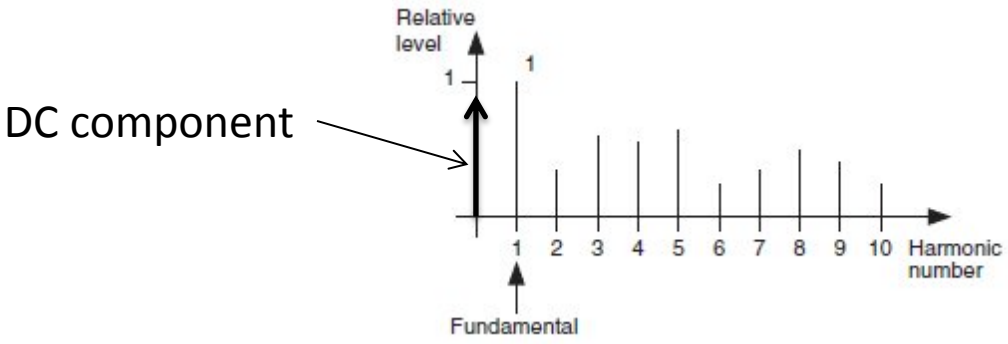
- Generate periodic rectangular waveforms with configurable:
  - Frequency
  - Duty cycle
- Generate synchronized periodic rectangular waveforms including dead time

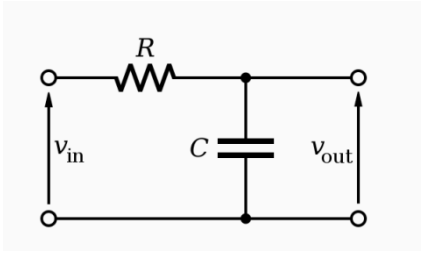


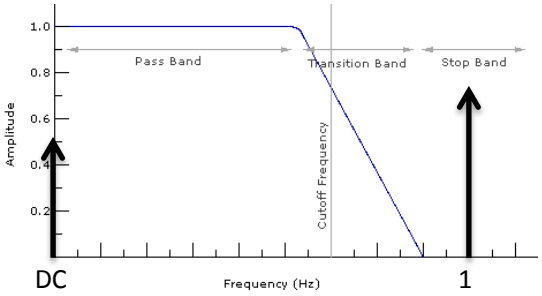


Applications of PWM functions

- PWM singnals are a efficent way of generating analog voltage
  - PWM signal spectrum contain a DC component
    - Amplitude: $V_{dc}=V_{OH}$*duty_cycle
  - A low pass filter can extract the DC component

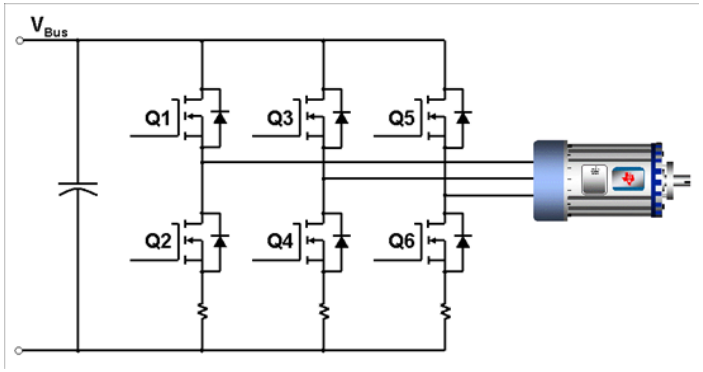


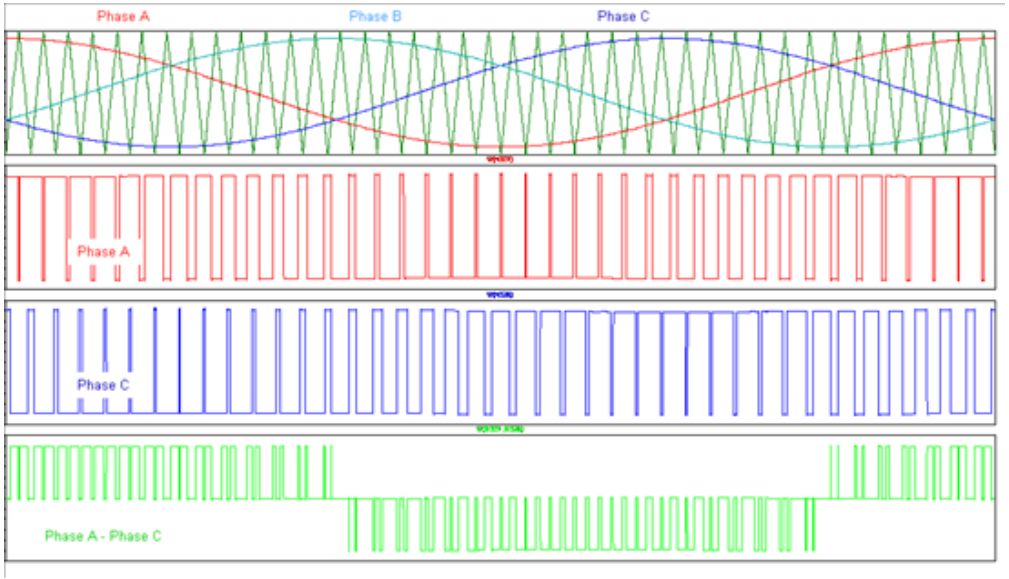




Applications of PWM function

- Direct drive of loads using PWM signals
  - Physical Inertia is the low pass filter (heating resistors, electric motors)
- Some MCU contain a Motor Control PWM device specialized in producing 6 phase PWM signals for driving electric motors (BLDC motors)
- Modulating duty cycle, synthetic sinusoidal waveforms can be reconstructed





RTC Functions

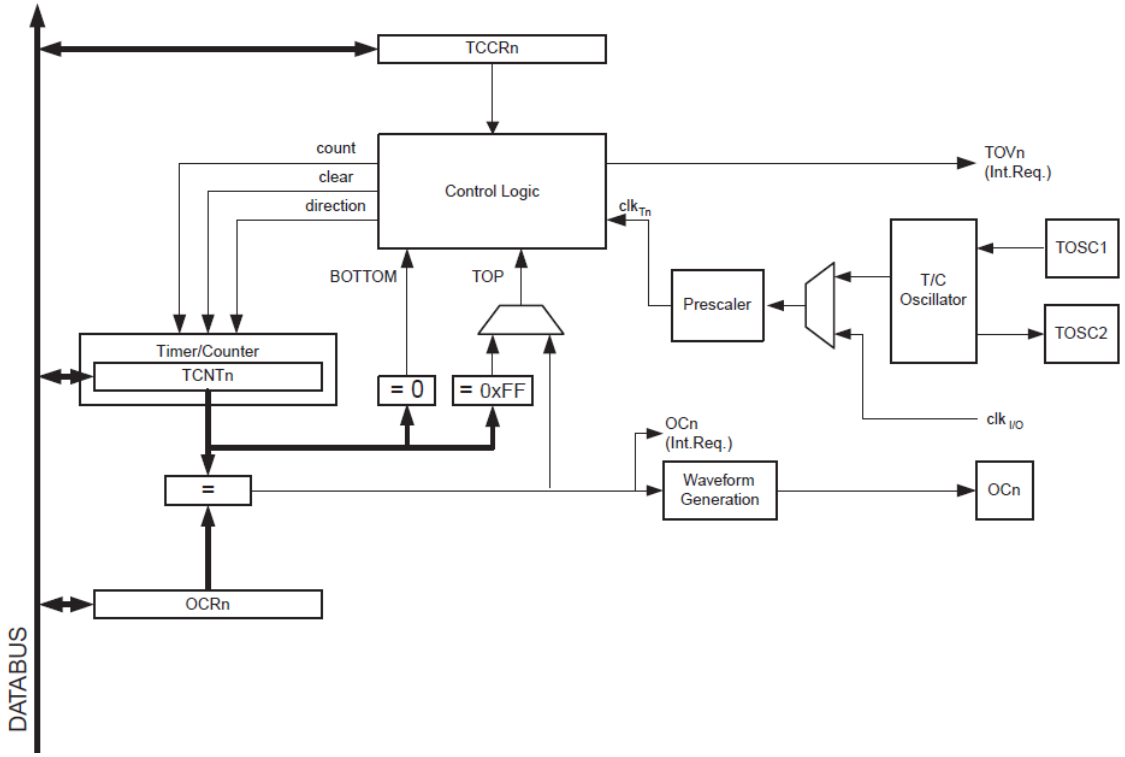
Real Time Clock

- keeps track of the current time in human units (hours, minutes, day, month) and BCD format
- often have an alternate source of power and clock, so they can continue to keep time while the primary source of power is off or unavailable
- Can produce interrupts on events (alarm) and also awake the microcontroller from power saving states

ATmega Timers

- 8 bit Timer/Counter

Figure 34. 8-bit Timer/Counter Block Diagram

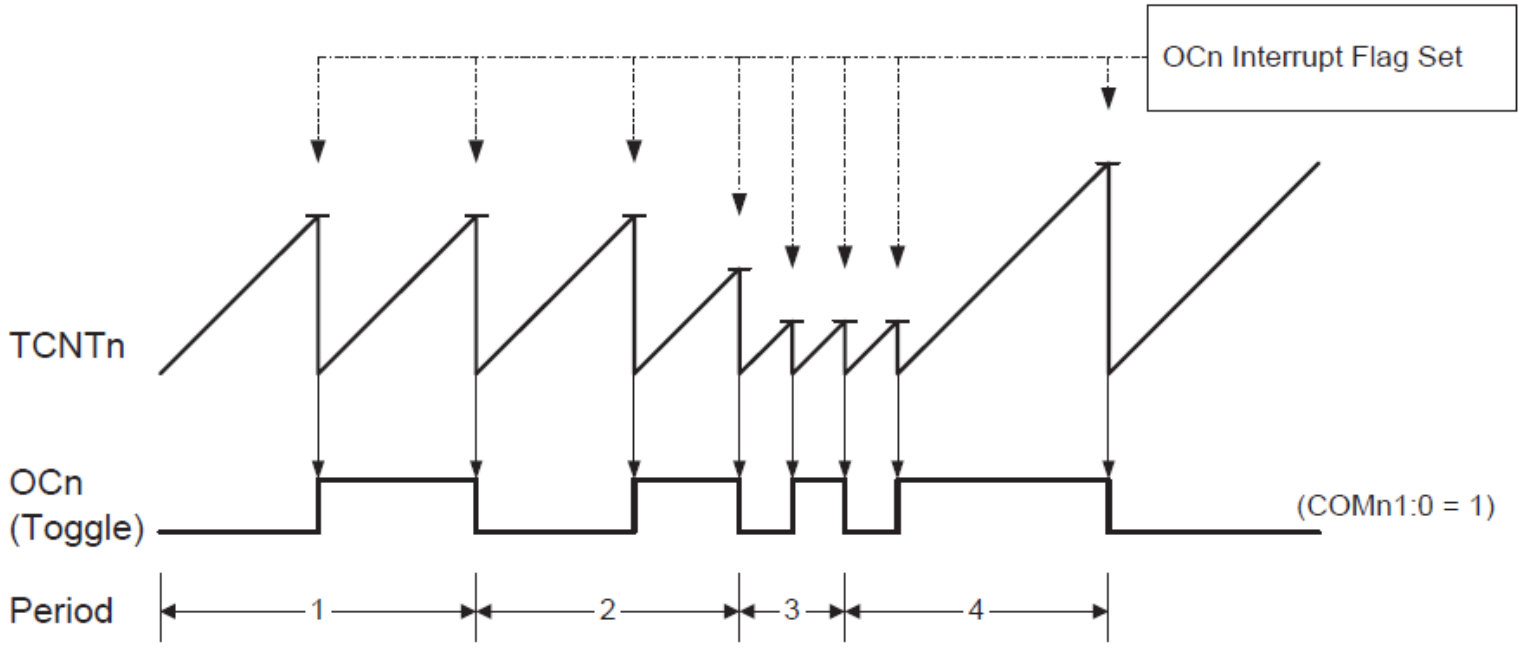


- Single Channel Counter
- Clear Timer on Compare Match (Auto Reload)
- Glitch-free, Phase Correct Pulse Width Modulator (PWM)
- Frequency Generator
- 10-bit Clock Prescaler
- Overflow and Compare Match Interrupt Sources (TOV0 and OCF0)

ATmega Timers

- Clear Timer on Compare Match (CTC) Mode

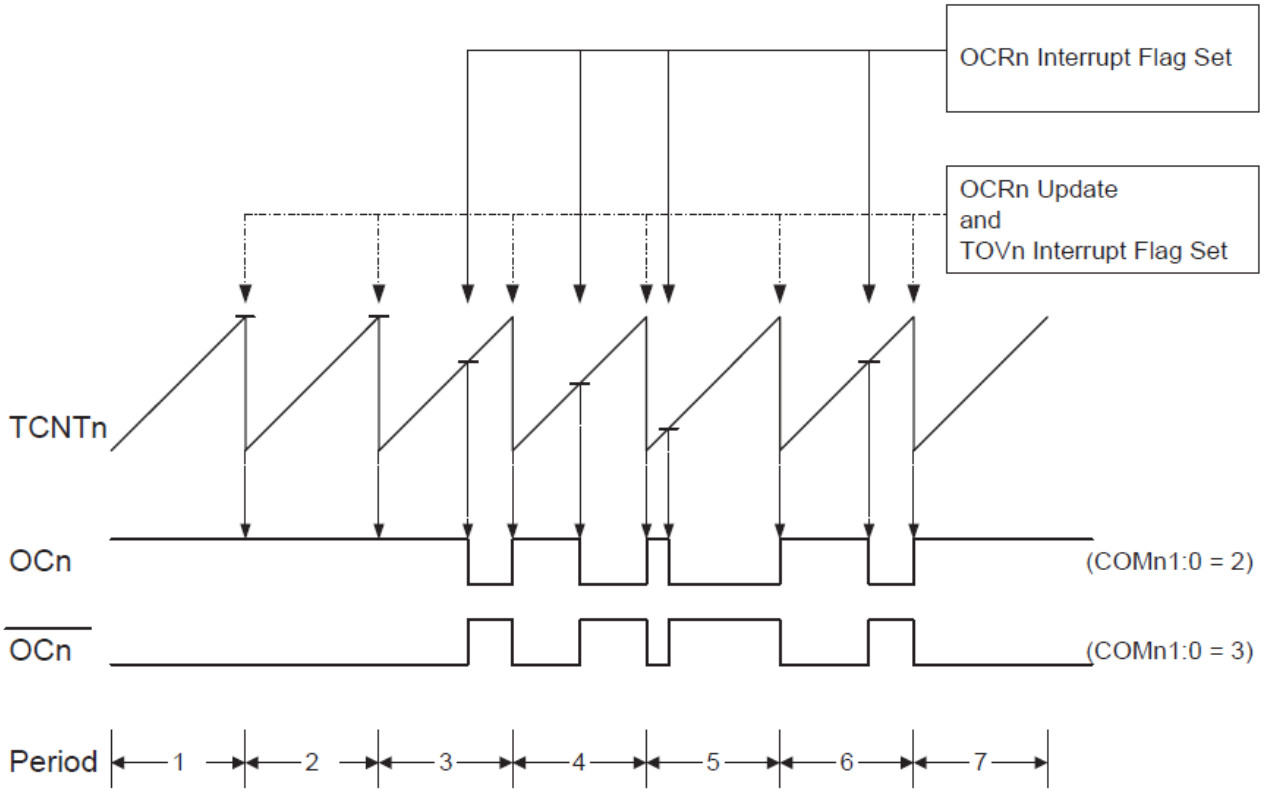
Figure 38. CTC Mode, Timing Diagram



ATmega Timers

- Fast PWM Mode

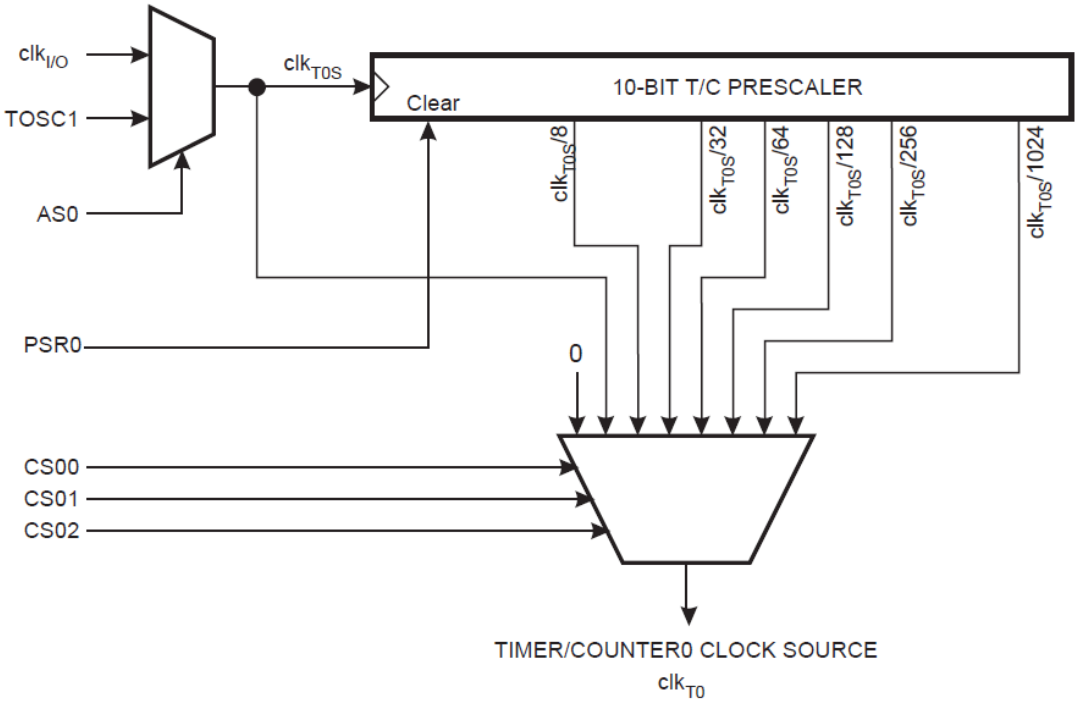
Figure 39. Fast PWM Mode, Timing Diagram



ATmega Timers

- Prescaler for Timer

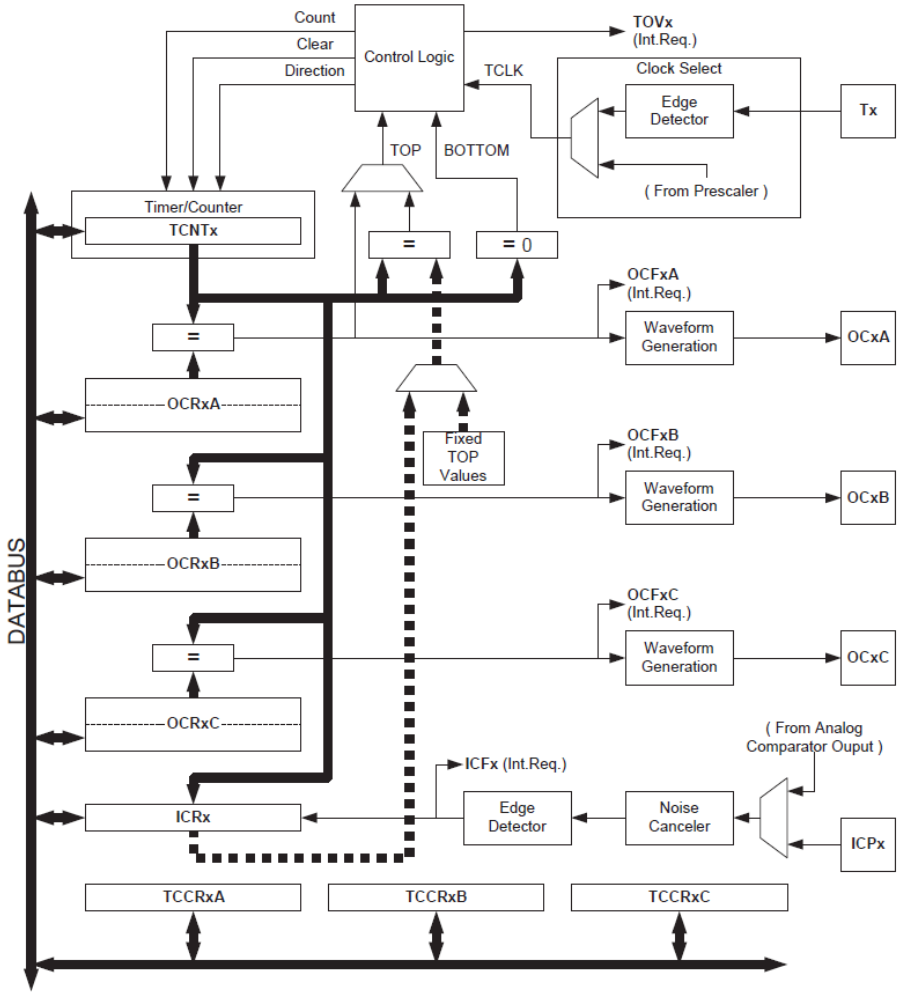
Figure 45. Prescaler for Timer/Counter0



ATmega Timers

- 16 bit Timer/Counter

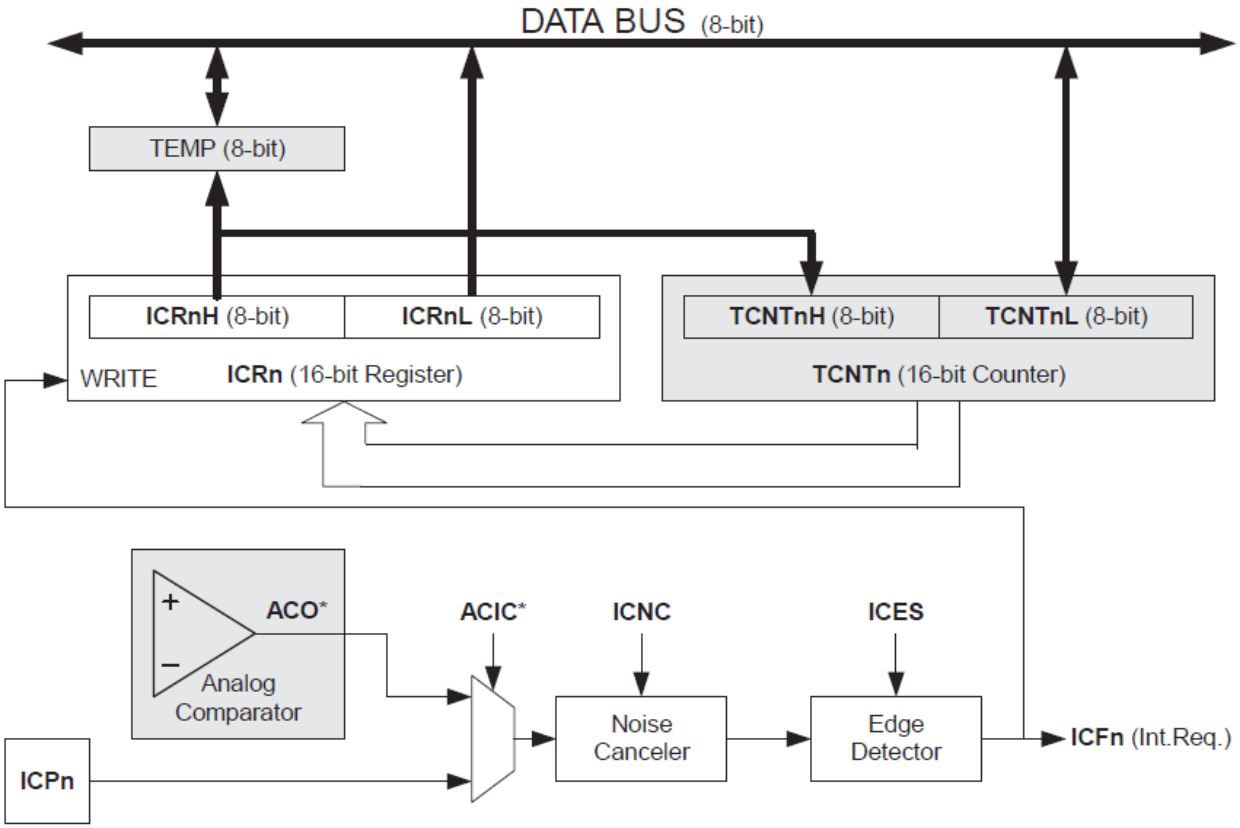
Figure 46. 16-bit Timer/Counter Block Diagram



- True 16-bit Design (i.e. ,Allows 16-bit PWM)
- Three Independent Output Compare Units
- One Input Capture Unit
- Clear Timer on Compare Match (Auto Reload)
- Glitch-free, Phase Correct Pulse width Modulator (PWM)
- Variable PWM Period
- Frequency Generator
- External Event Counter
- Ten Independent Interrupt Sources (TOV1, OCF1A, OCF1B, OCF1C, ICF1, TOV3, OCF3A, OCF3B, OCF3C, and ICF3)

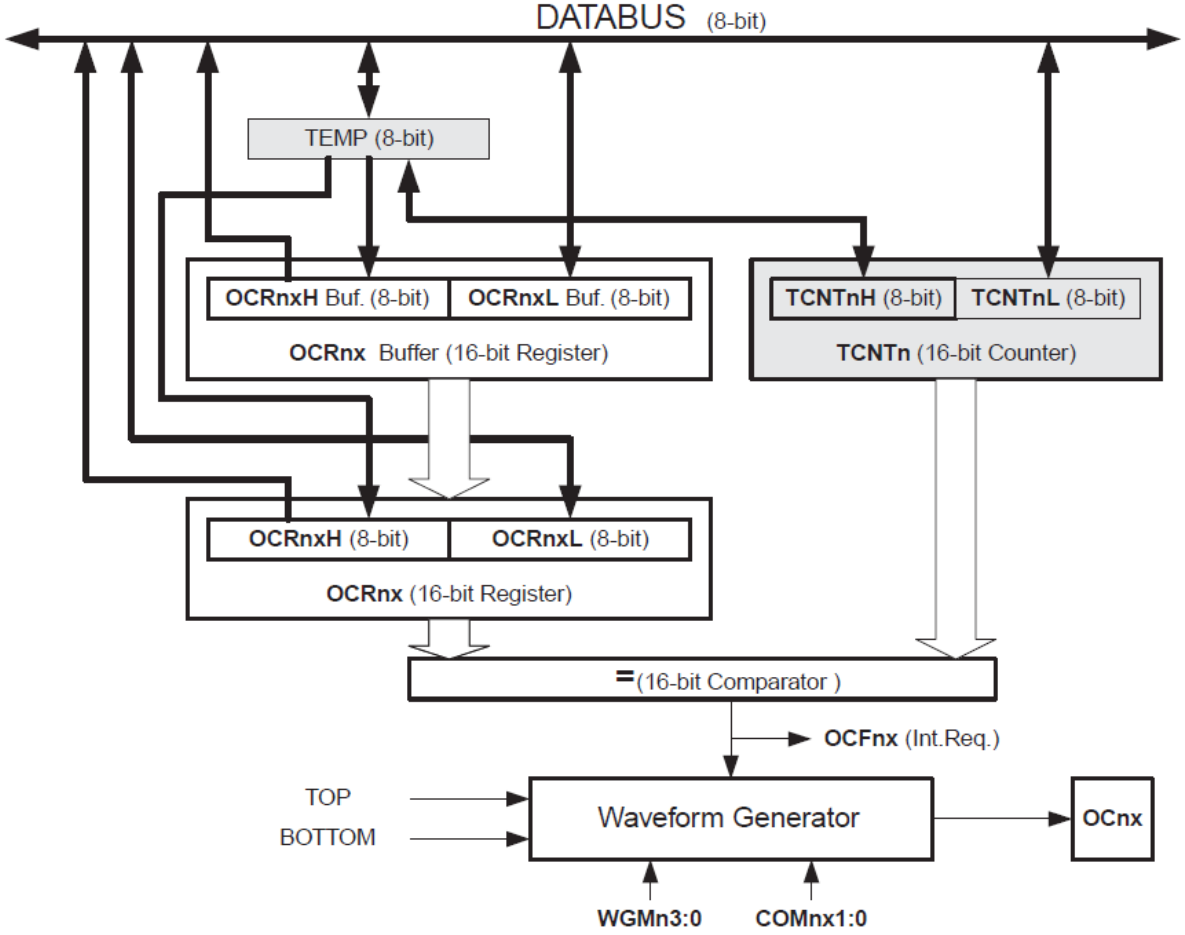
ATmega Timers – Input capture function

Figure 48. Input Capture Unit Block Diagram



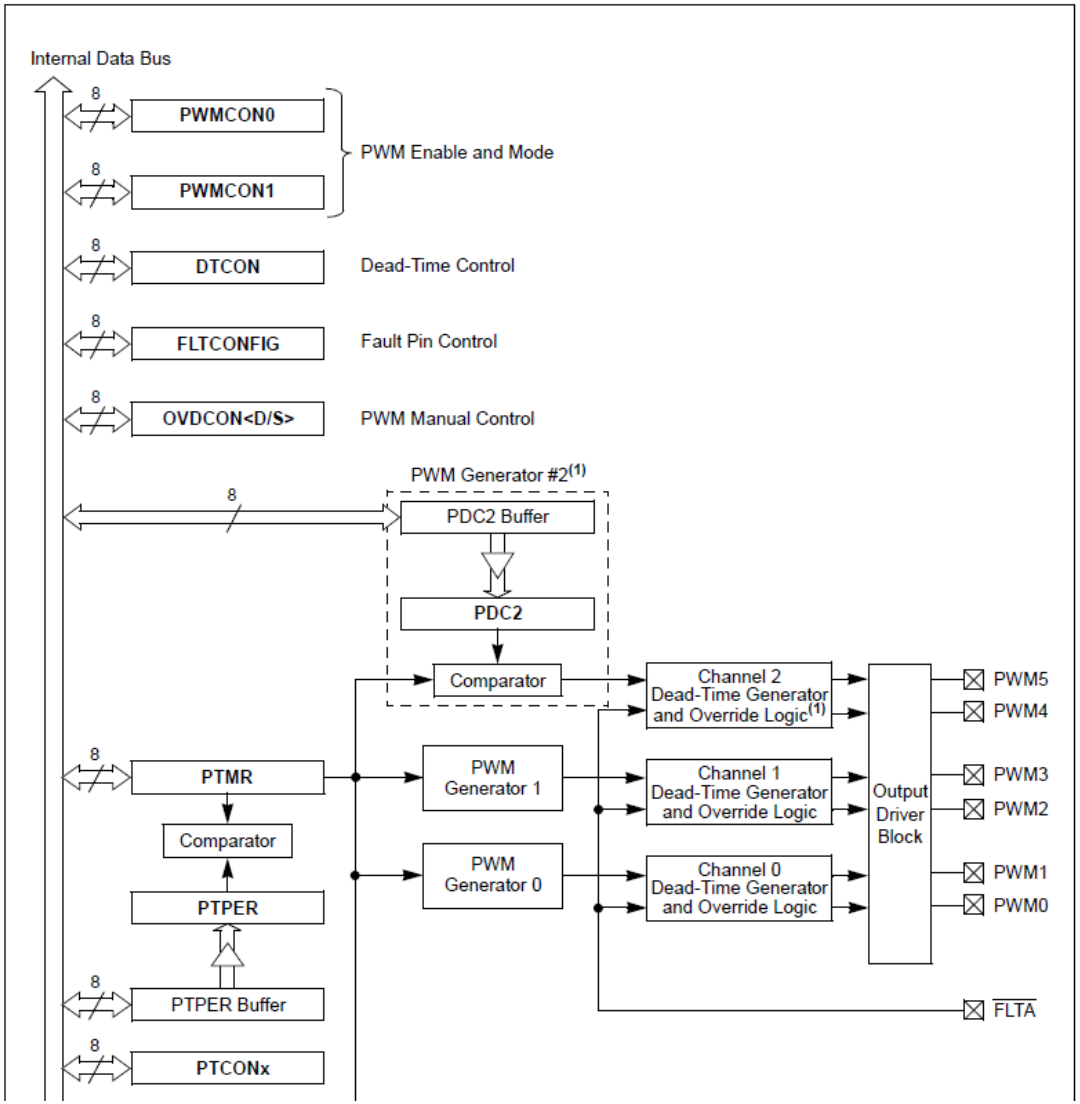
ATmega Timers – Output compare function

Figure 49. Output Compare Unit, Block Diagram



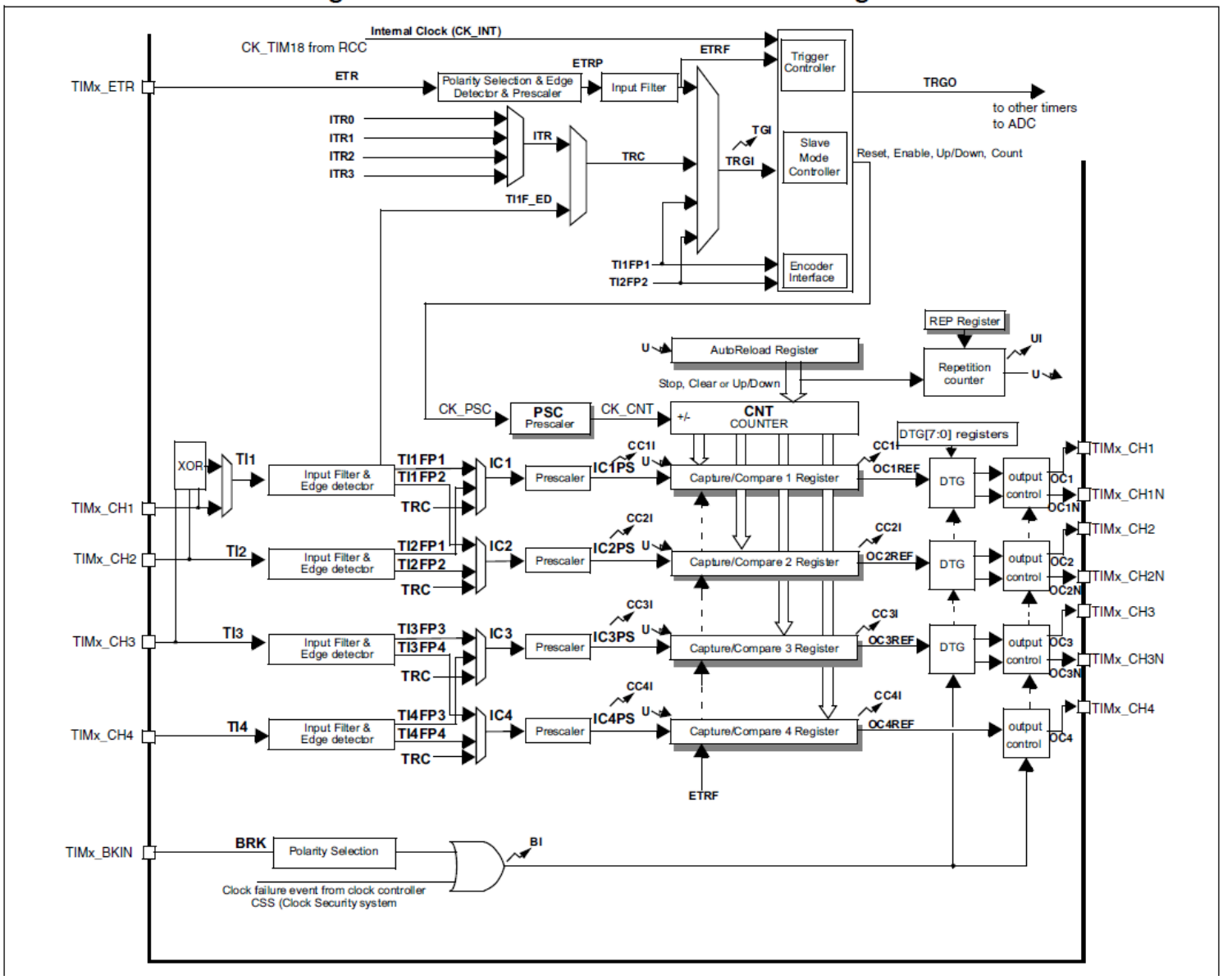
Microchip PIC - Power control PWM

FIGURE 14-1: POWER CONTROL PWM MODULE BLOCK DIAGRAM



STM32 – advanced control timer

Figure 39. Advanced-control timer block diagram



Clock generation in MCU

FIGURE 14-5: RC OSCILLATOR MODE

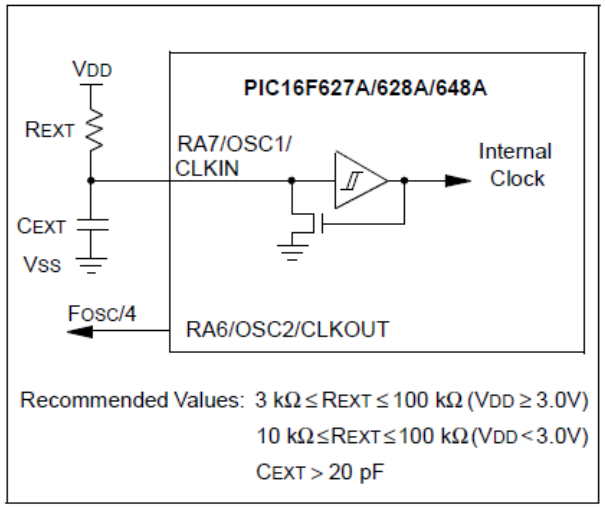
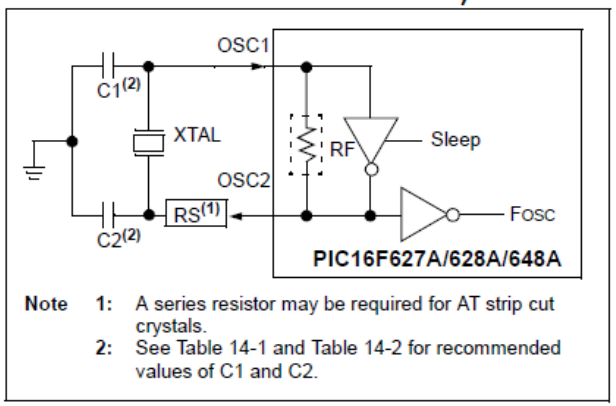


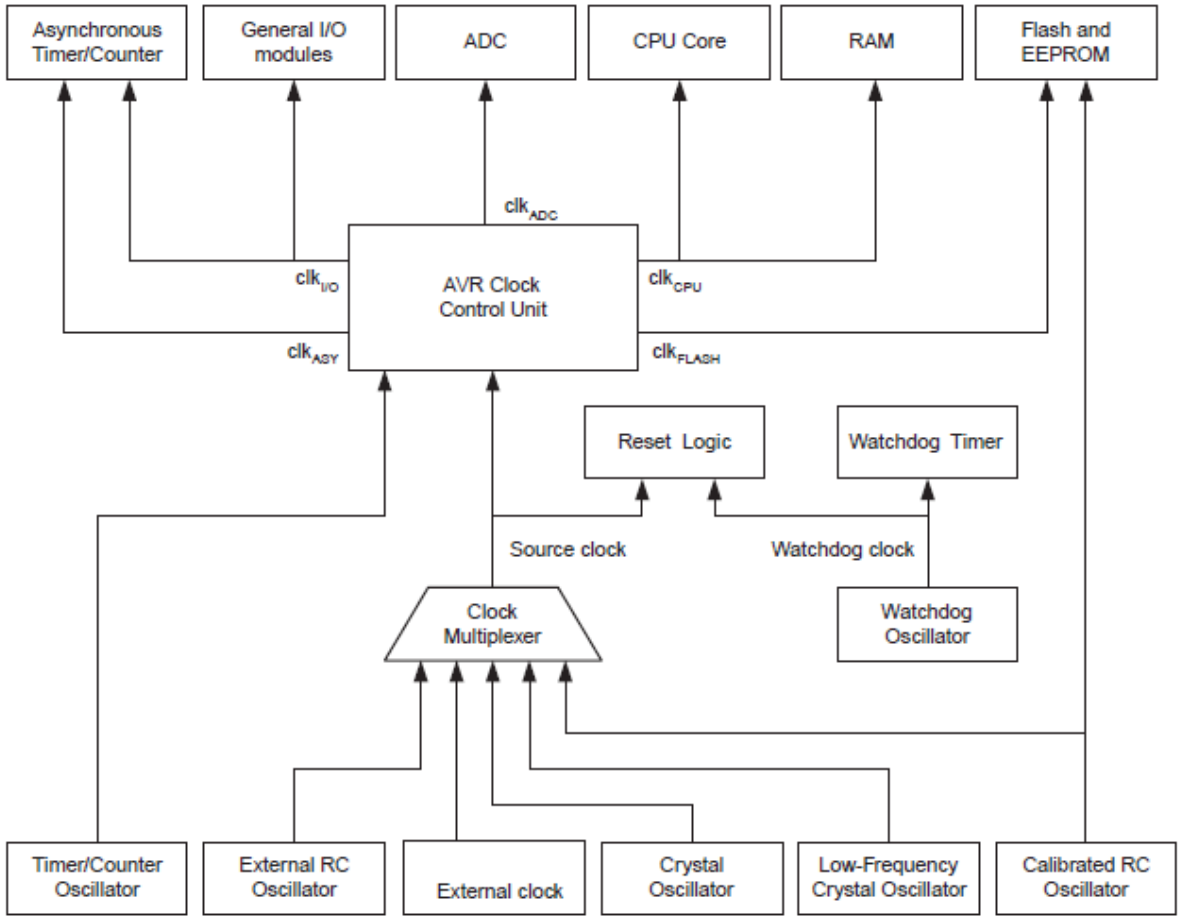
FIGURE 14-1: CRYSTAL OPERATION (OR CERAMIC RESONATOR) (HS, XT OR LP OSC CONFIGURATION)



- Clock generation can be:
  - Internal
  - External (synchronization with external hardware, accuracy, speed)
- Two kinds of oscillators are generally present
  - RC Oscillator (low cost, low accuracy)
  - Crystal oscillator
    - accurate to 100ppm
    - 10ppm with temperature compensation
    - 1ppm with temperature control

Clock tree - ATmega

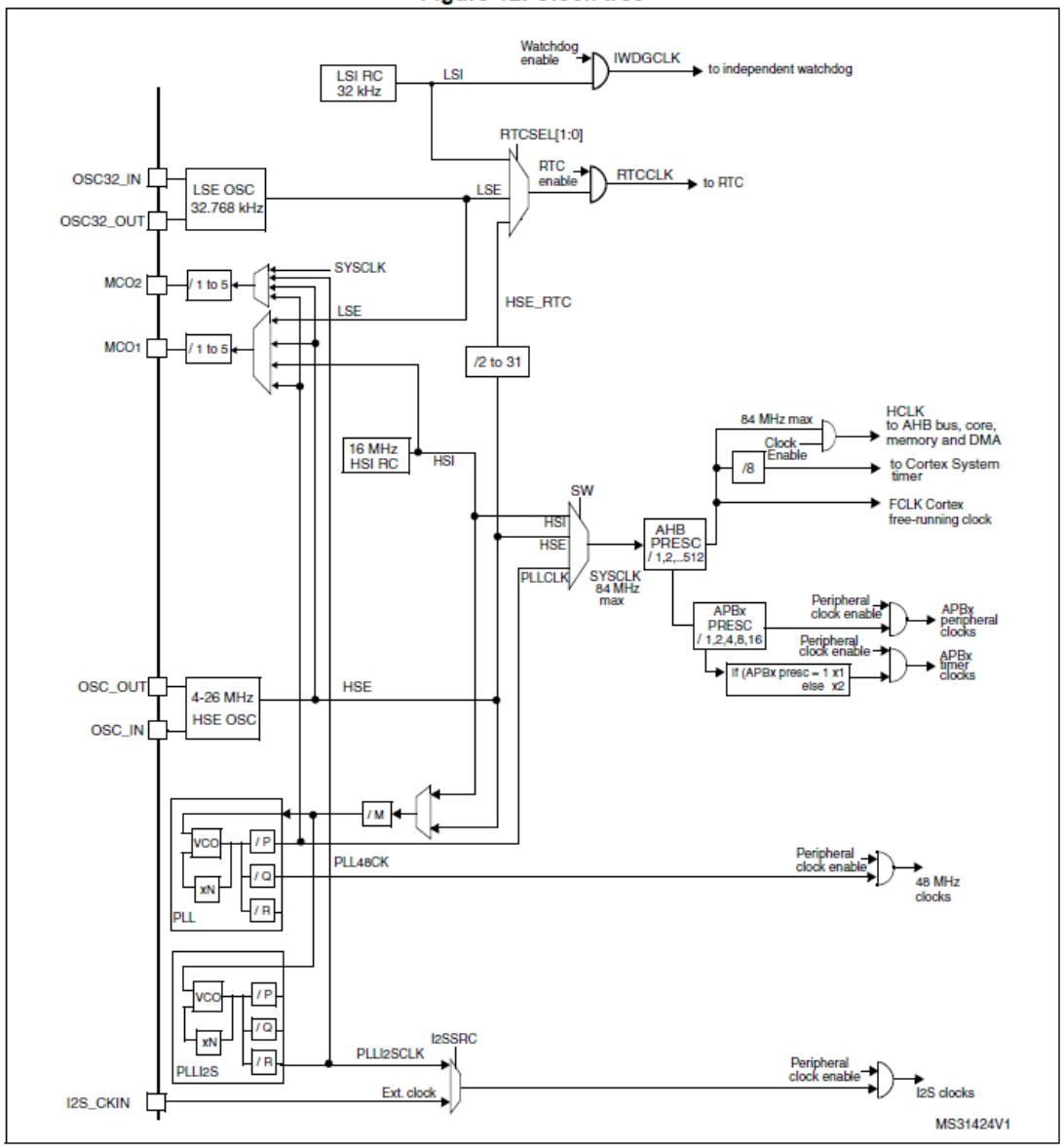
Figure 18. Clock Distribution



- A multiplexer selects clock source
- A control unit generates different clocks by dividing/multiplying the base clock

Clock tree – STM32

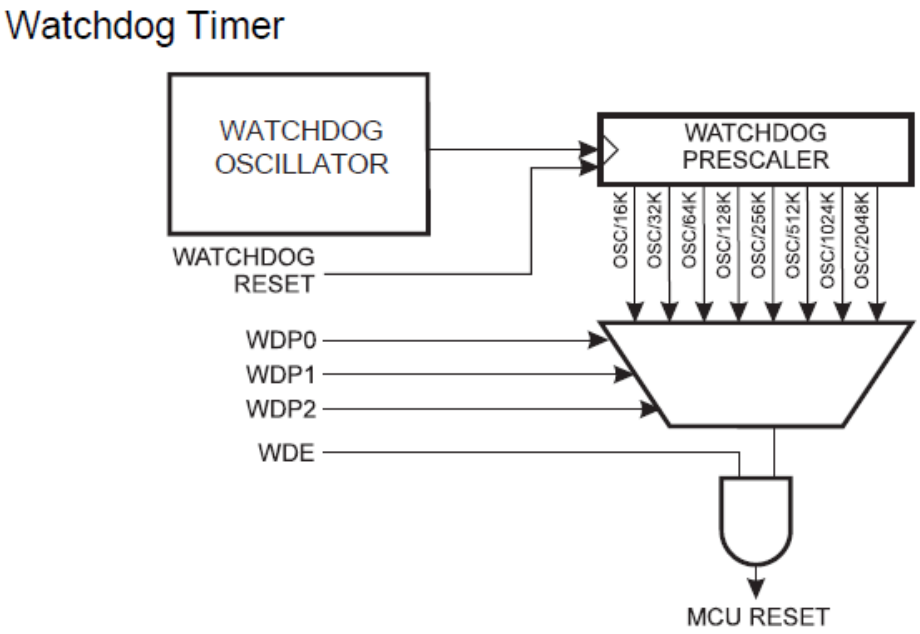
Figure 12. Clock tree



- Clock is generated by
  - HSI (high speed internal)
  - HSE (high speed external)
- Frequency can be multiplied by PLL (PLLCLK) by a fraction N/P
- Fixed frequency can be generated for peripherals (48MHz USB)

Watchdog Timer

- Clocked from separate on-chip oscillator
- Three operating modes
  - Interrupt
  - System reset
  - Interrupt and system reset
- Selectable time-out period from 16ms to 8s
- Possible hardware fuse watchdog always on (WDTON) for fail-safe mode



Watchdog Timer

- The WDT gives an interrupt or a system reset when the counter reaches a given time-out value. In normal operation mode, it is required that the system restart the counter before the time-out value is reached. If the system doesn't restart the counter, an interrupt or system reset will be issued.
- In **Interrupt mode**, the WDT gives an interrupt when the timer expires. This interrupt can be used to wake the device from sleep-modes, and also as a general system timer. One example is to limit the maximum time allowed for certain operations, giving an interrupt when the operation has run longer than expected.
- In **System Reset mode**, the WDT gives a reset when the timer expires. This is typically used to prevent system hang-up in case of runaway code.
- **In Interrupt and System Reset mode**, combines the other two modes by first giving an interrupt and then switch to System Reset mode. This mode will for instance allow a safe shutdown by saving critical parameters before a system reset.