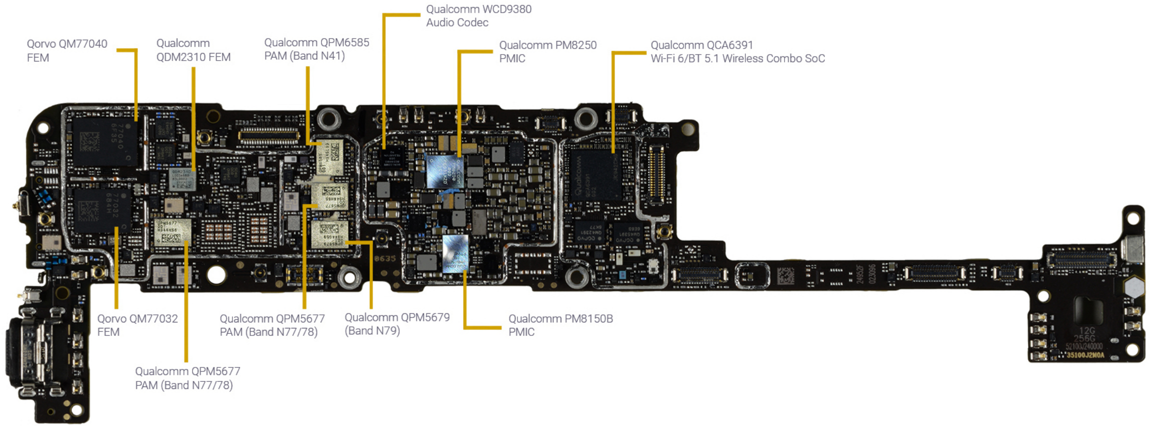

Qorvo QM77040 FEM
Qualcomm QDM2310 FEM
Qualcomm QPM6585 PAM (Band N41)
Qualcomm WCD9380 Audio Codec
Qualcomm PM8250 PMIC
Qualcomm QCA6391 Wi-Fi 6/BT 5.1 Wireless Combo SoC
Qorvo QM77032 FEM
Qualcomm QPM5677 PAM (Band N77/78)
Qualcomm QPM5679 (Band N79)
Qualcomm PM8150B PMIC
Qualcomm QPM5677 PAM (Band N77/78)

AVR timers

Kizito NKURIKIYEYEZU, Ph.D.

What are timers

- Timers are asynchronous counter, i.e., they are separate circuits on an AVR MCU which can run independently from the main program.
- Using timers is independent of the core AVR CPU, thus, reduces overhead and processing load on the MCU
- Timers are used for example to set some time interval like your alarm. This can be very precise to a few microseconds.
- The deterministic clock makes it possible to measure time by counting the elapsed cycles and take the input frequency of the timer into account.

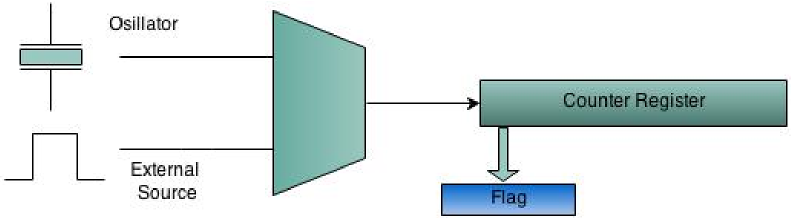


FIG 1. A General View of Counters and Timers in Microcontrollers

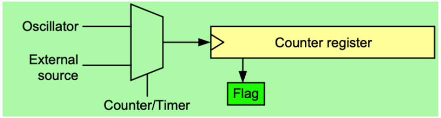
Use of timers

- **Internal Timer**—In this mode it used generate precise delays. Or as precise time counting machine. In this case, the oscillator frequency can be directly feed to the timer or it can be pre-scaled.
- **External Counter** —In this mode the unit is used to count events on a specific external pin on a MCU.
- **Pulse width Modulation(PWM) Generator**: PWM is used in speed control of motors and various other applications.
- The AVR has one to six timers depending on the family member. They are referred to as Timers 0, 1, 2, 3, 4, and 5.
- They can be used as timers to generate a time delay or as counters to count events happening outside the microcontroller.

Basic registers of timers

An AVR MCU has timers with several byte-addressable register:

- TCNTn (Timer/Counter register)



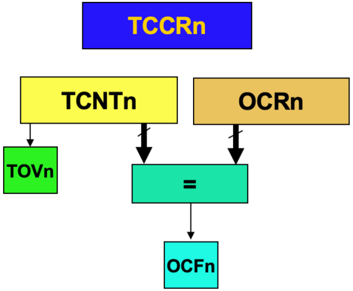
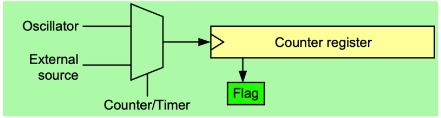


FIG 2. Registers in an AVR timer

Basic registers of timers

An AVR MCU has timers with several byte-addressable register:

- TCNTn (Timer/Counter register)
- TOVn (Timer Overflow flag)



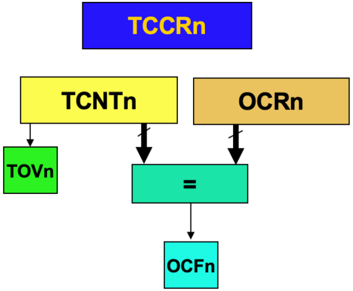
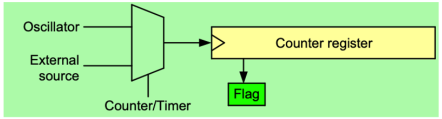


FIG 2. Registers in an AVR timer

Basic registers of timers

An AVR MCU has timers with several byte-addressable register:

- TCNTn (Timer/Counter register)
- TOVn (Timer Overflow flag)
- TCCRn (Timer Counter control register)



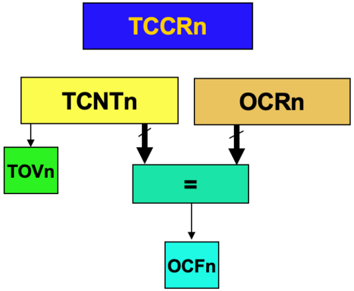
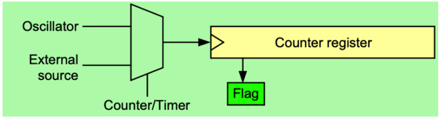


FIG 2. Registers in an AVR timer

Basic registers of timers

An AVR MCU has timers with several byte-addressable register:

- TCNTn (Timer/Counter register)
- TOVn (Timer Overflow flag)
- TCCRn (Timer Counter control register)
- OCRn (output compare register)



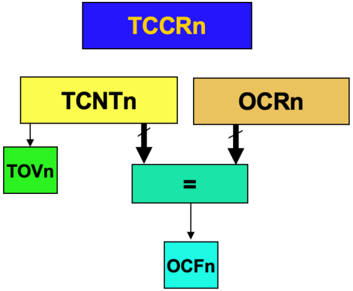
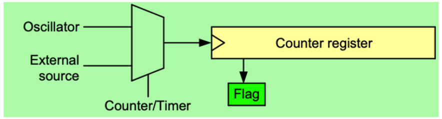


FIG 2. Registers in an AVR timer

Basic registers of timers

An AVR MCU has timers with several byte-addressable register:

- TCNTn (Timer/Counter register)
- TOVn (Timer Overflow flag)
- TCCRn (Timer Counter control register)
- OCRn (output compare register)
- OCFn (output compare match flag)



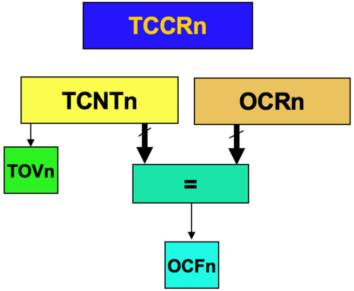


FIG 2. Registers in an AVR timer

Basic registers of timers

- Upon reset, $TCNT_n = 0$. Then, it counts up with each clock pulse
- The content of the timer/counter can be accessed with the $TCNT_n$ register. It is possible to also load a value into the counter.
- The TOV_n flag allows to know when the $TCNT_n$ overflows
- The $TCCR_n$ controls the mode of operation of the timer. For example,it allows to either work as timer or as a counter by loading a proper value into the $TCCR0$
- The OCR_n register is used to compare with the content of the $TCNT_n$. When they are equal the OCF_n flag will be set.

Timers in the ATMega328

Timers in the ATMega328

ATmega328 has three times:

- 8-bit TC0—It is also known as TCNT0 (Timer/Counter 0).
- 16-bit TC1 —It is also known as TCNT1(Timer/Counter 0).
- 8-bit TC2 —It is also known as TCNT2 (Timer/Counter 2).

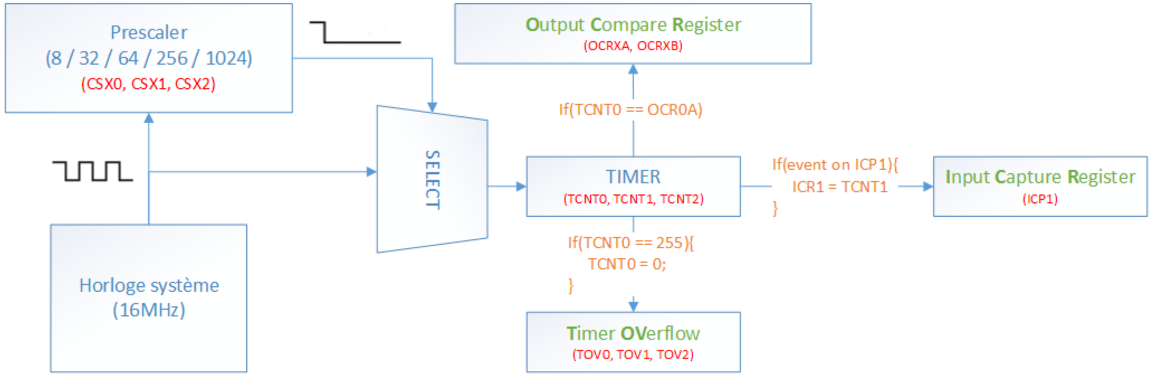
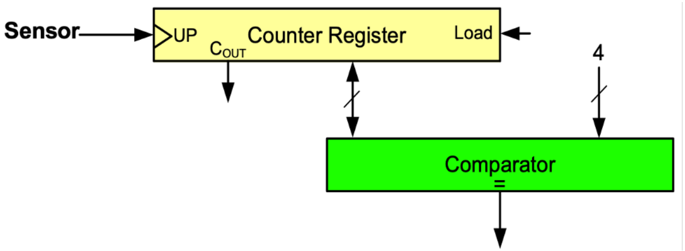


FIG 3. AVR Timer diagram

EXAMPLE: Counting

- The counter register can be used to record the frequency of an event
- The counter register stores the current value of a counter. Any time a certain even occurs, the value of the counter is increased by one.
- For example, this would allow us to know how often a wheel has turned during a given time
- AVR MCU comes with 8-bit and 16-bit timers.
- The 8-bit counters can count to $2^8 - 1 = 255$ whilst the 16- bit counter can count to $2^{16} - 1 = 65,536$.



EXAMPLE: Calculating a car’s speed

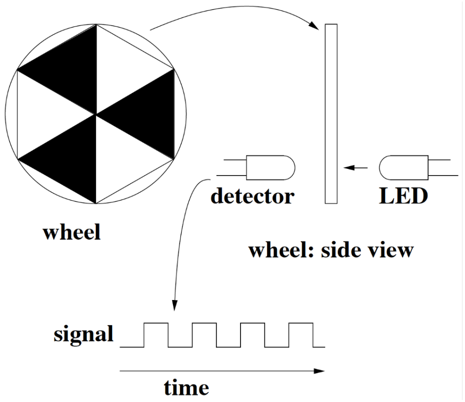


FIG 4. A wheel equipped with a simple encoder. The encoder wheel is partitioned into regions that are alternating opaque and transparent. When the wheel is turning at a constant velocity, the detector observes a digital signal of some frequency and a duty cycle of 50%. A counter can be used to count the number of times the detector transitions from high to low; thus, allow to estimate how many times the wheel has turned

Example: Making an asynchronous delay

- Each clock pulse increments the timer's counter by one, the timer measures intervals in periods of one on the input frequency:

$$\text{Timer Resolution} = \frac{1}{\text{Input Frequency}}$$

- For instance, if we supply a 100Hz signal to a timer, our period becomes:

$$\begin{aligned}\text{Timer Resolution} &= \frac{1}{\text{Input Frequency}} \\ &= \frac{1}{100Hz} \\ &= .01s\end{aligned}$$

- In this case, the period is $T = .01s$, thus our timer will be able to measure times that are a multiple of this duration.

Example: Making an asynchronous delay

- Objective: flash an LED at 10Hz on an AVR MCU running at 1MHz clock.
- At 1MHz, one tick is 1μs, so flashing an LED at 10Hz requires 50ms of delay
- Algorithm 1 shows a pseudocode required to drive the LED:

Algorithm 1: Flashing an LED at 10Hz with a timer

```
Result: Initialize the LED DDR to outpu
Initialize the timer hardware;
while While forever do
    if COUNTER_VALUE > 0.05 then
        Reset counter;
        Turn on LED;
    end
end
```

Example: Making an asynchronous delay

- But what is the counter values of 50ms?
- Reminder: 8-bit counter store up to 255 and 16-bit counter up to 65535
- For 1MHz clock, each tick corresponds to 1μs.
- Thus, for 50ms (i.e., 1/20seconds)

$$\begin{aligned}
\text{Target Timer Count} &= \frac{1}{\text{Target Frequency}} \Big/ \frac{1}{\text{Timer Clock Frequency}} - 1 \\
&= \frac{1}{20} \Big/ \frac{1}{1000000} - 1 \\
&= \frac{.05}{0.000001} - 1 \\
&= 50000 - 1 \\
&= 49999
\end{aligned}$$

- We need a counter that can count up to 49999.
- Only a 16-bit counter is able to do this.

```c
#include <avr/io.h>
int main (void){
  DDRB |= (1 << 0);
  // Set up timer
  TCCR1B |= (1 << CS10);
  while(1){
    //Check if the counter if full
    if (TCNT1 >= 49999){
      PORTB ^= (1 << PB0);
      // Reset timer value
      TCNT1 = 0;
    }
  }
}
```

LISTING 1: Flashing an LED at 10Hz with a timer