INTERFACING

TIMERS 1[NORMAL & CTC MODES]

AMIT

Counting time Concept:

Definition:

- In middle eras, there was a Muslim scientist called Ibn Al-Jazari who invented a mechanical clock depends on weight and water, it sounds and throw a metal ball into a container every half an hour, so if we counts the balls we can calculate the passed time: like if we find 7 balls,
  that means about three and half hours have been passed.
- Depending the last concept, our micro controller works depending on a Clock cycles, this cycle has a known rate, like our MC is 16 MHz, so if we managed to count these cycles, and knowing the period of every cycle, we can calculate the time.
- So, if we have a counter register that is increased by one every clock-cycle, if it is read, its value expresses how many clock-cycles are come, and we can count the time if we has the period of one cycle by multiplying the value by the period of one cycle.

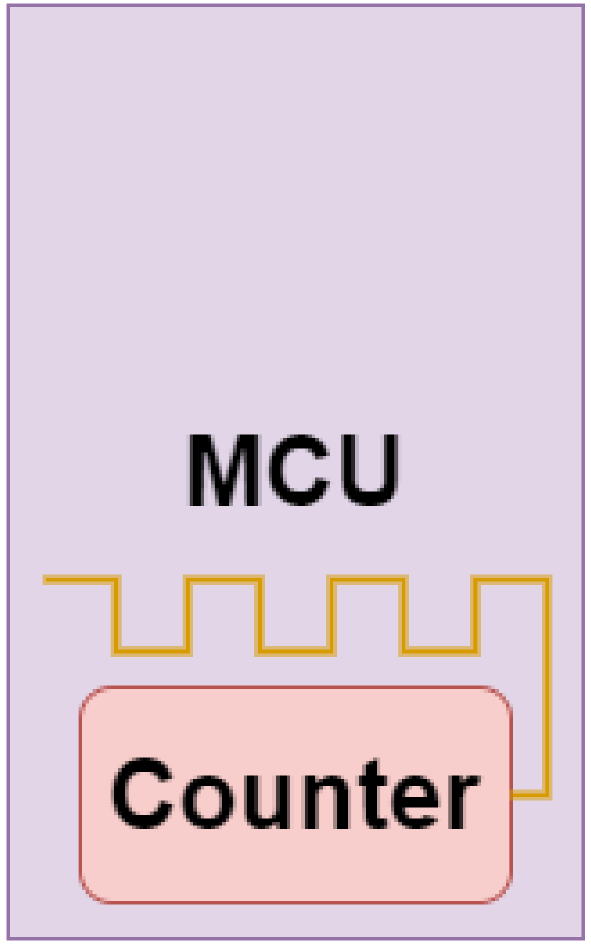
Timer Hardware:

Definition:

- According to the previous theory, any timer hardware depends on a counter register that is increased by one every clock cycle, it depends on the system's clock.

- It means that any timer is a counter "it will be declared later" , but triggering event of this counter is a clock.

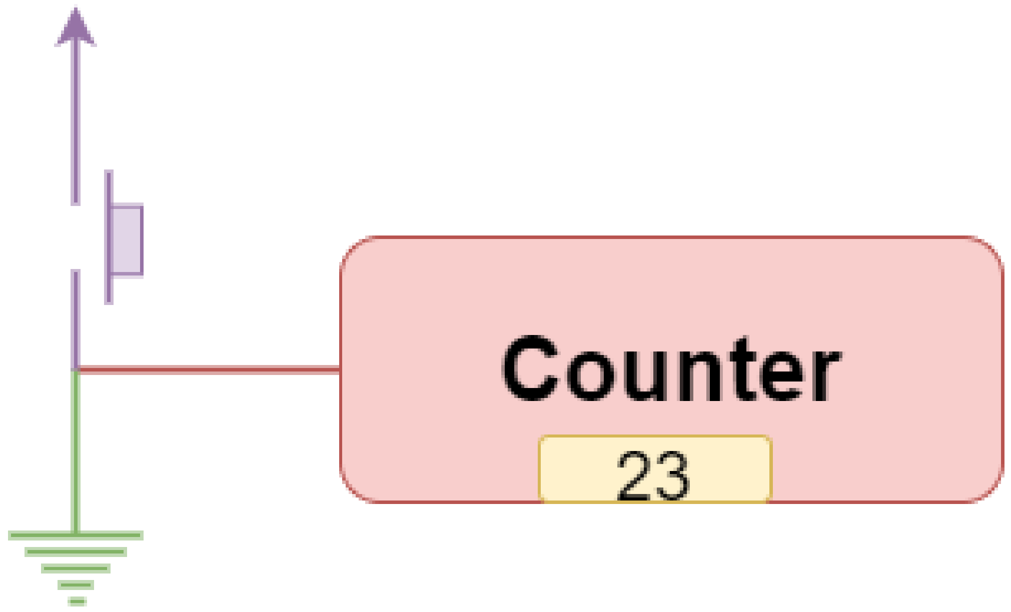
- Timer can not be transferred to a counter because it internally depends on the system's clock, so no way to be a counter without an external pin.



Counter Hardware:

Definition:

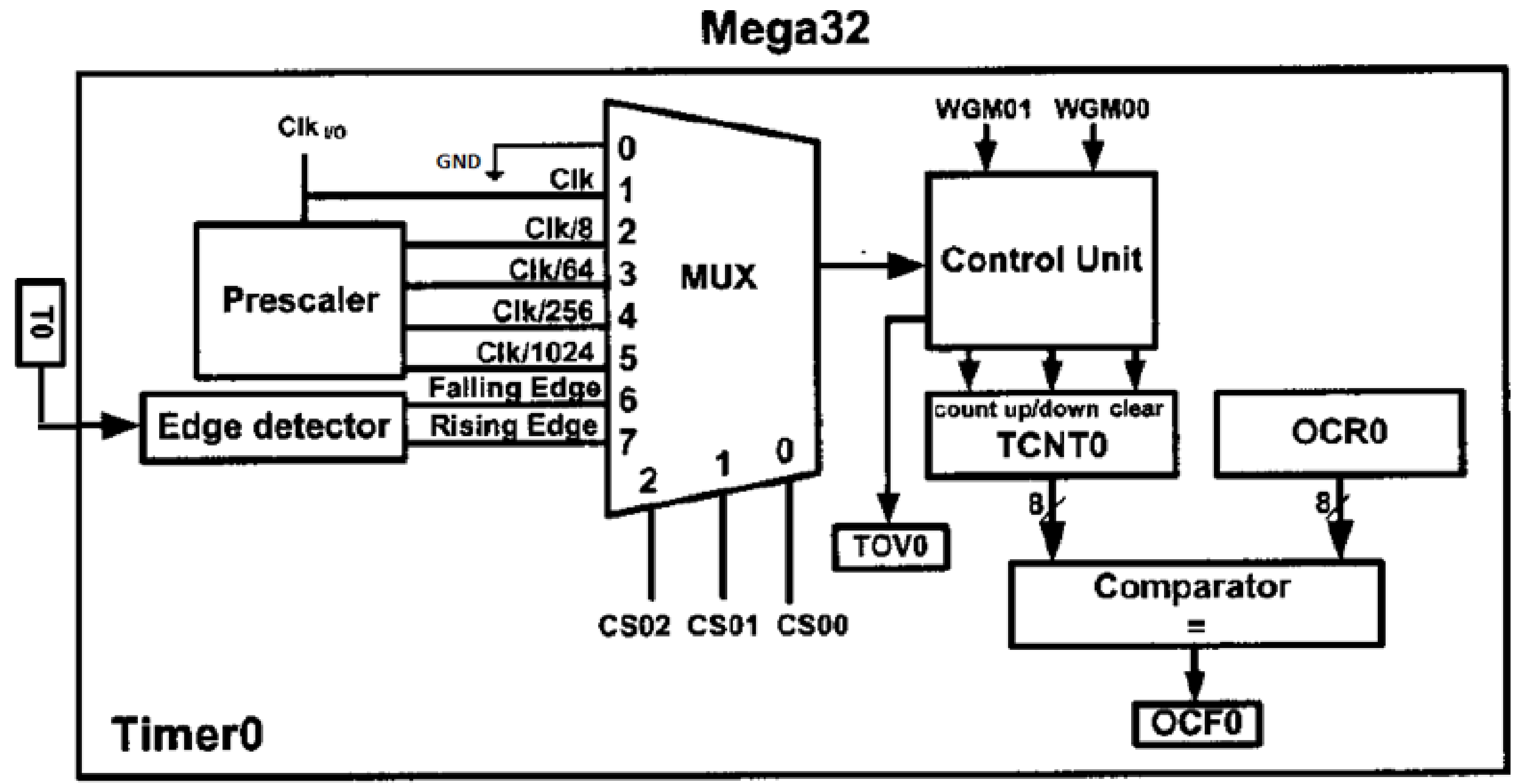
- It is a hardware circuit depends on increasing one into the counter register every a triggering event occurs.
- It can be triggered every rising edge of falling edge.
- It can be used to count any external event like counting how many presses on the switch like the following figure:
- Any counter can be transferred to Timer by triggering it by a clock.



AMIT

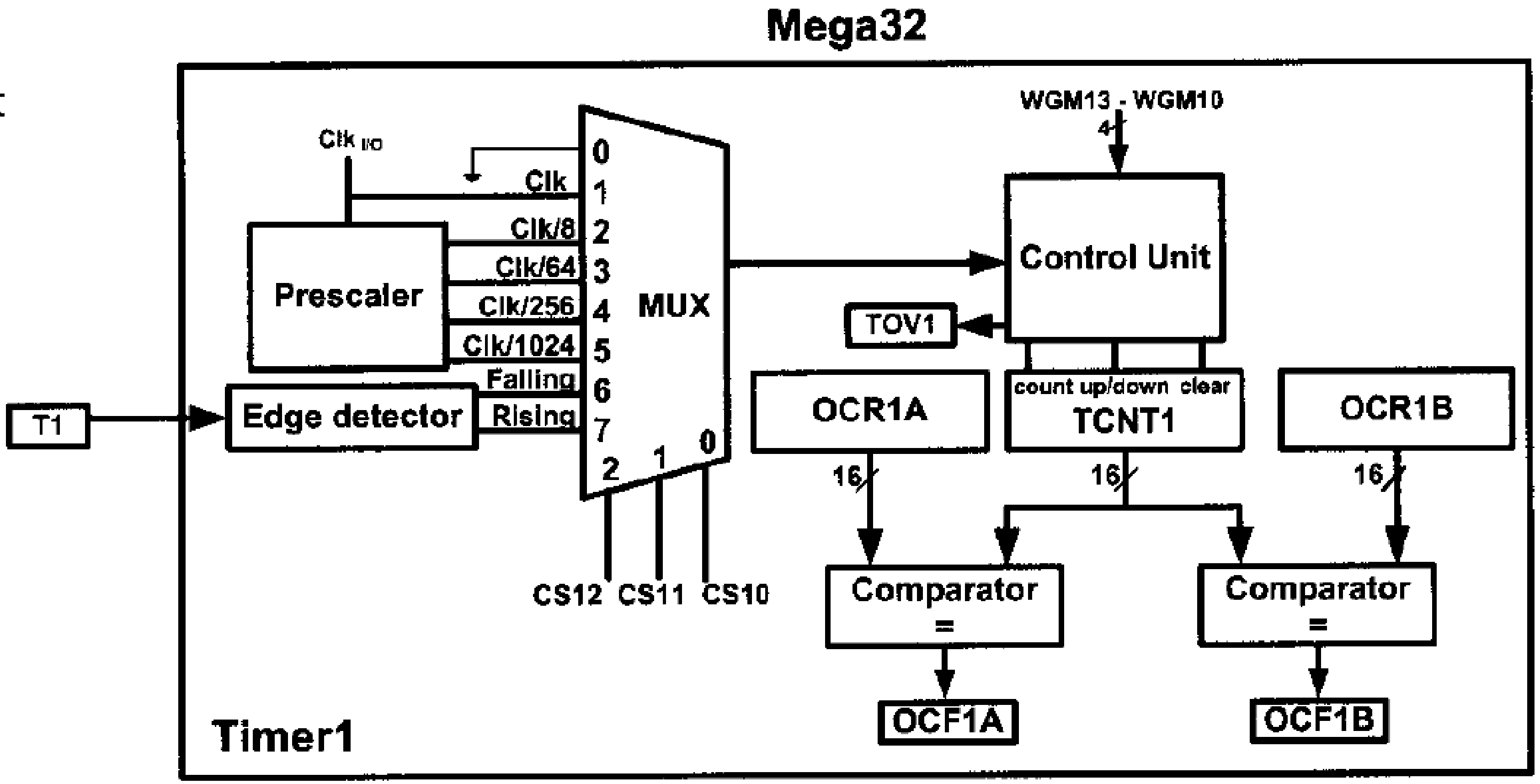
Timer Hardware in ATMEGA32:

- We have three-peripheral timers into our microcontroller:
  - TIMER0,
    8-bit resolution.



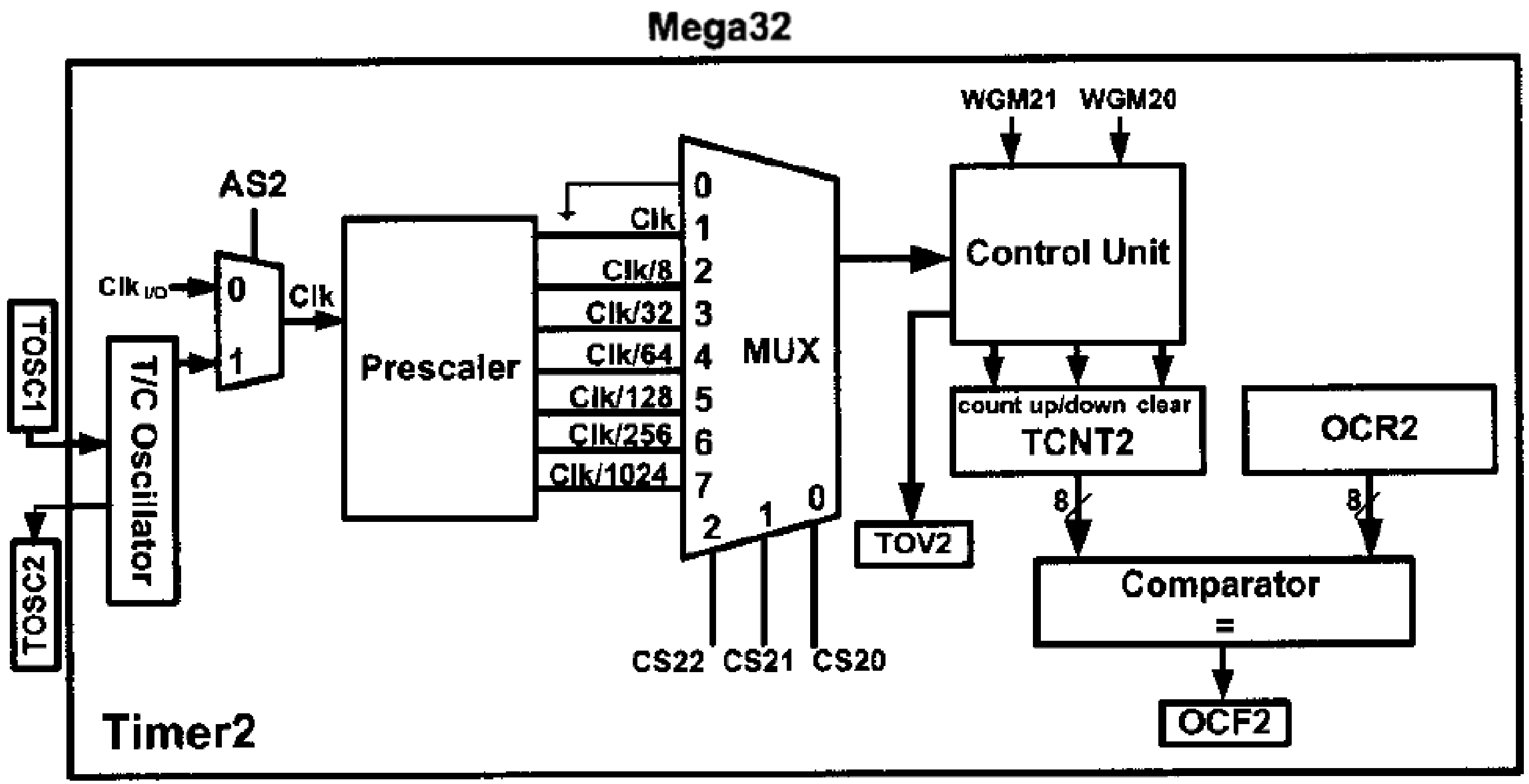
Timer Hardware in ATMEGA32:

- We have three-peripheral timers into our microcontroller:
    - TIMER1, 16-bit resolution.



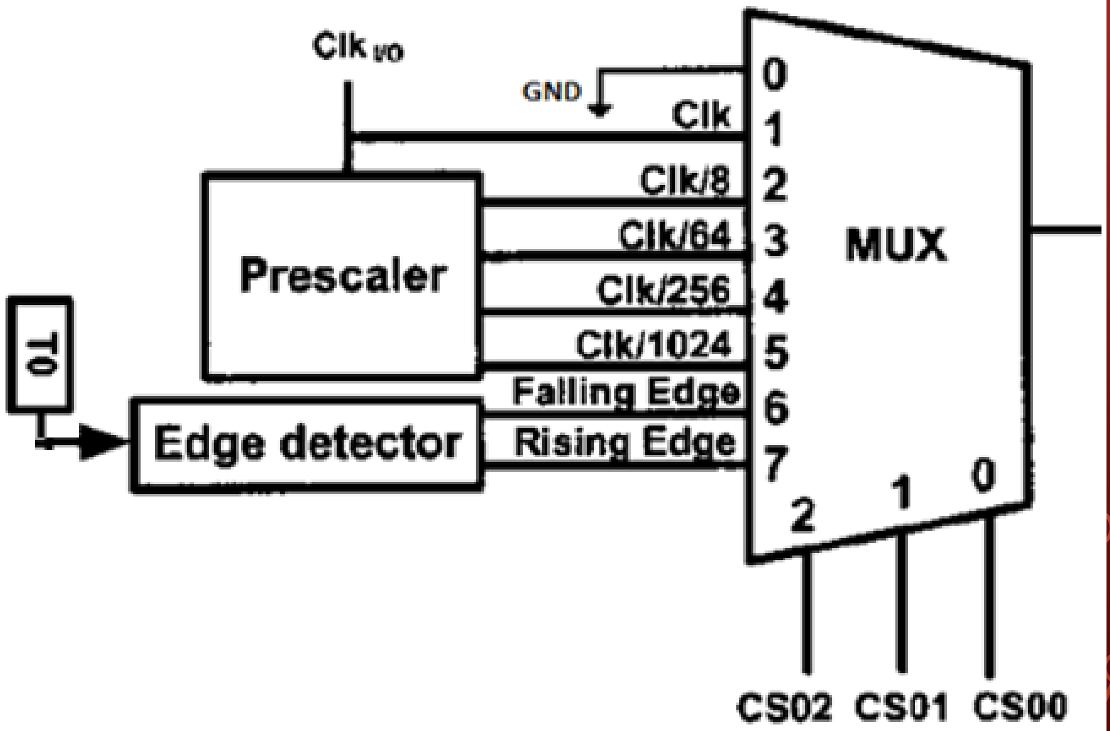
Timer Hardware in ATMEGA32:

- We have three-peripheral timers into our microcontroller:
  - TIMER2,
    8-bit resolution.



Timer0:

- As we mentioned before, timer peripheral depends on the system clock, and no way to be a counter without an external pin, but our timer0 peripheral is a Timer and Counter, it can be configured to be a timer or counter.
- Timer hardware has a multiplexer that determine clock source will be selected by setting "**CS00, CS01, CS02**":
    - **000** : there is no clock source.
    - **001** : system clock is divided by 1.
    - **010** : system clock is divided by 8.
    - **011** : system clock is divided by 64.
    - **100** : system clock is divided by 256.
    - **101** : system clock is divided by 1024.
    - **110** : every falling edge on "**T0**" pin.
    - **111** : every rising edge on "**T0**" pin.



AMIT

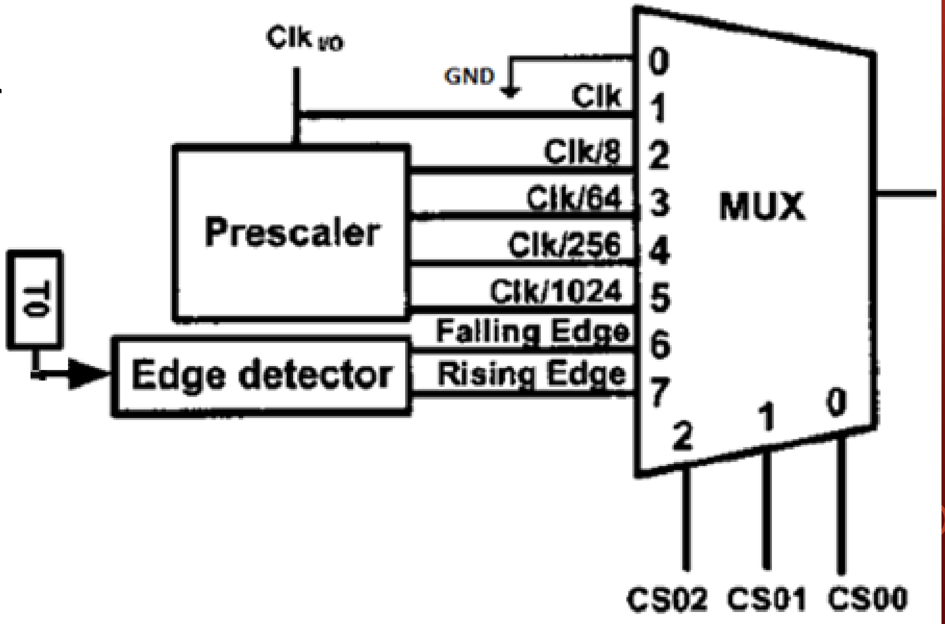
Timer0:

- If "**CSxx**" bits are selected to "**000**", this means the timer hardware control will be connected to ground, so the timer will be disabled.

- If "**CSxx**" bits are selected to "**001**", this means the timer hardware control will be directly connected to system clock, so Timer speed will be the same Processor speed.

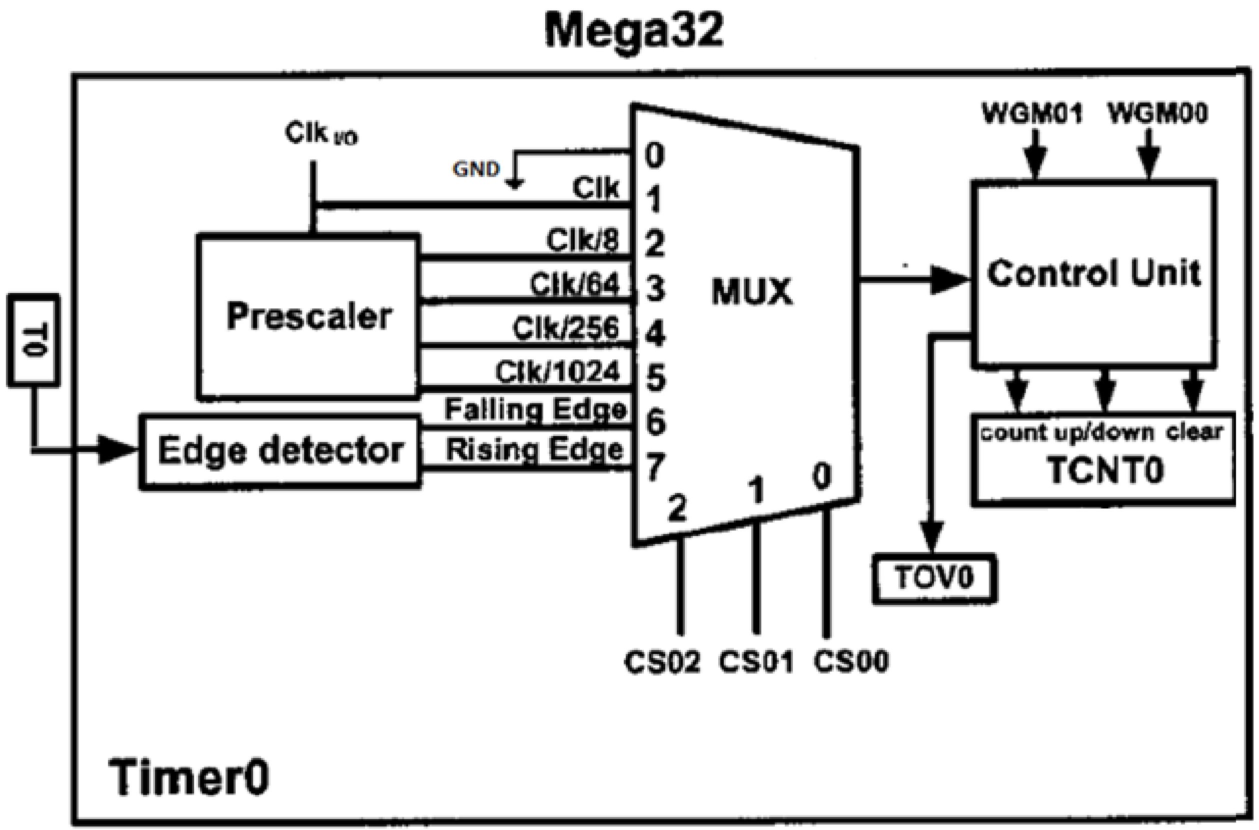
- If "**CSxx**" bits are selected from "**010**" to "**101**", this means the timer hardware control will be connected to system clock, but clock is divided by a specific factor.

- If "**CSxx**" bits are selected to "**11x**", this means the timer hardware control will be Counter, which it will be triggered by an external event through "**T0**" pin.



Timer0:
How it works:

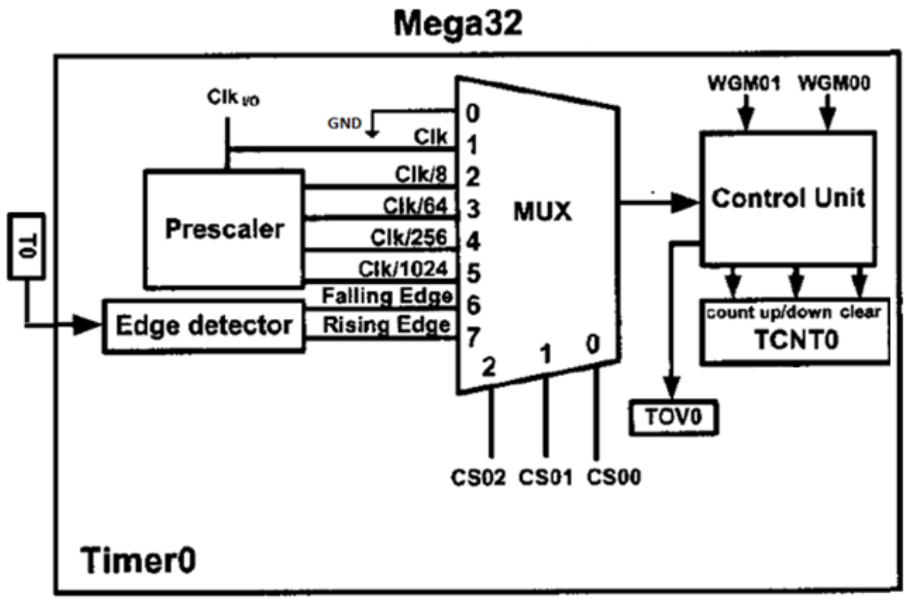
- Timer0 has different modes:
  - Normal "Overflow mode".
  - CTC "Compare Match".
  - Fast PWM.
  - Phase-correct PWM.
- Overflow mode in Timer0:
  - In this mode, the counter register "**TCNT0**" is increased by one every clock cycle or triggered event on "**T0**" until it reaches to its top, then it overflows.
  - After overflow, "**Tov**" flag will be raised, and it may cause an interrupt service routine if "**Timer PIE**" and "**GIE**".



Timer0:

How it works:

- After clock-source or edge-type selection, the multiplexer will through out the clock or the event triggered to the control unit of timer.
- After that, the control unit will take an action according to its configuration bits "**WGM00, WGM01**".
- "**WGM00, WGM01**" is configured as an overflow mode, so the control unit will increase "**TCNT0**" by one.
- It will repeated until the register overflow and "**Tov**" flag is raised.
- So, this flag can be used into a blocking system "**Polling**" or into a foreground-background system "**Interrupt**".



Timer0:

How to Calculate the Time?:

- As we mentioned before, "**TCNT0**" register is increased by one every clock cycle, so by knowing the value of register and the period of the cycle:
  - $Time\ of\ One\ Cycle = \frac{1}{Timer\ frequency} = \frac{prescaler}{system\ frequency}$
  - $Time\ of\ counts = all\ counts\ of\ timer\ \times\ time\ of\ one\ cycle$

    $= all\ counts\ of\ timer\ \times \frac{prescaler}{system\ frequency}$
- So, the time of Overflow of Timer0 will be calculated by:
  - $Time\ of\ counts = 256\ \times \frac{prescaler}{system\ frequency}$
- Now, what will happen if the desired time is higher or lower than the overflow time?

Timer0:

Calculation of desired time overflow counts:

- Assume that the desired time is "**1s**" and the chosen Prescaler is "**256**" and our system frequency is "**16 MHz**":
  - One second is higher than of overflow time, so we need more than one overflow to get "**1s**":
    - $Num\ of\ Ovf = \frac{Desired\ Time}{Time\ of\ overflow} = \frac{1}{256 \times \frac{256}{16\ x\ 10^6}} = 244.140625$
    - The number of Ovfs is an Irrational number, so the number must be approximated to the next integer number "Not an algebraic approximation" even if the number is a rational, so, it will equal "245".
    - The fraction expresses a part of ovf, so we can calculate the preload of "**TCNT0**" by:
      - $preload = \max counts\ - fraction\ counts$
        $= \max counts\ -\ (\max count - fraction \times \max counts)$
        $= maxcounts\ \times\ (1\ - fraction) = 220$
      - *So, TCNT0 must be set by 220 every 245 ovfs.*

Timer0:

Calculation of desired time overflow counts:

- Assume that the desired time is “**1s**” and the chosen Prescaler is “**256**” and our system frequency is “**16 MHz**”:
  - One second is higher than of overflow time, so we need more than one overflow to get “**1s**”:
    - $Num\ of\ Ovf = \frac{Desired\ Time}{Time\ of\ overflow} = \frac{1}{256 \times \frac{256}{16\ x\ 10^6}} = 244.140625$
    - The number of Ovfs is an Irrational number, so the number must be approximated to the next integer number “Not an algebraic approximation” even if the number is a rational because fraction means that there is a new needed overflow event but not a complete ovf, part of it , so, it will equal “**245**”.
    - The fraction expresses a part of ovf, so we can calculate the preload of “**TCNT0**” by:
      - $preload = \max counts - fraction\ counts$
        $= \max counts - (\max count - fraction \times \max counts)$
        $= maxcounts \times (1 - fraction) = 220$
      - So, TCNT0 must be set by **220** every 245 ovfs.

Timer0:

Calculation of desired time overflow counts:

- Sudo code of the previous example:

```
u16 counter=0;
int main(void){
   set timer0;
   TCNT0=220;
   sei;
   while  (1)      {
          if (counter == 245){
                 TCNT0=220;
                 counter=0;
                 // do what you want;
          }
} }
ISR (TIMER0_OVF){
   counter ++;
}
```

Timer0:

Calculation of desired time overflow counts:

- Assume that the desired time is "**1ms**" and the chosen Prescaler is "**256**" and our system frequency is "**16 MHz**":
  - One second is higher than of overflow time, so we need more than one overflow to get "**1ms**":
    - $Num\ of\ Ovf = \frac{Desired\ Time}{Time\ of\ overflow} = \frac{0.001}{256 \times \frac{256}{16\ x\ 10^6}} = 0.244140625$
    - The number of Ovfs is an Irrational number, so the number must be approximated to the next integer number "Not an algebraic approximation" even if the number is a rational because fraction means that there is a new needed overflow event but not a complete ovf, part of it , so, it will equal "**1**".
    - The fraction expresses a part of ovf, so we can calculate the preload of "**TCNT0**" by:
      - $preload = maxcounts \times (1 - fraction) = 193.5$
      - *So, TCNT0 must be set by 193 every ovf event.*

Timer0:

Calculation of desired time overflow counts:

- Sudo code of the previous example:

```
u8 flag=0;
int main(void){
  set timer0;
  TCNT0=193;
  sei;
  while  (1)      {
         if (flag == 1){
                TCNT0 = 193;
                flag =0;
                // do what you want;
         }
} }
ISR (TIMER0_OVF){
   flag = 1;
}
```

Timer0:

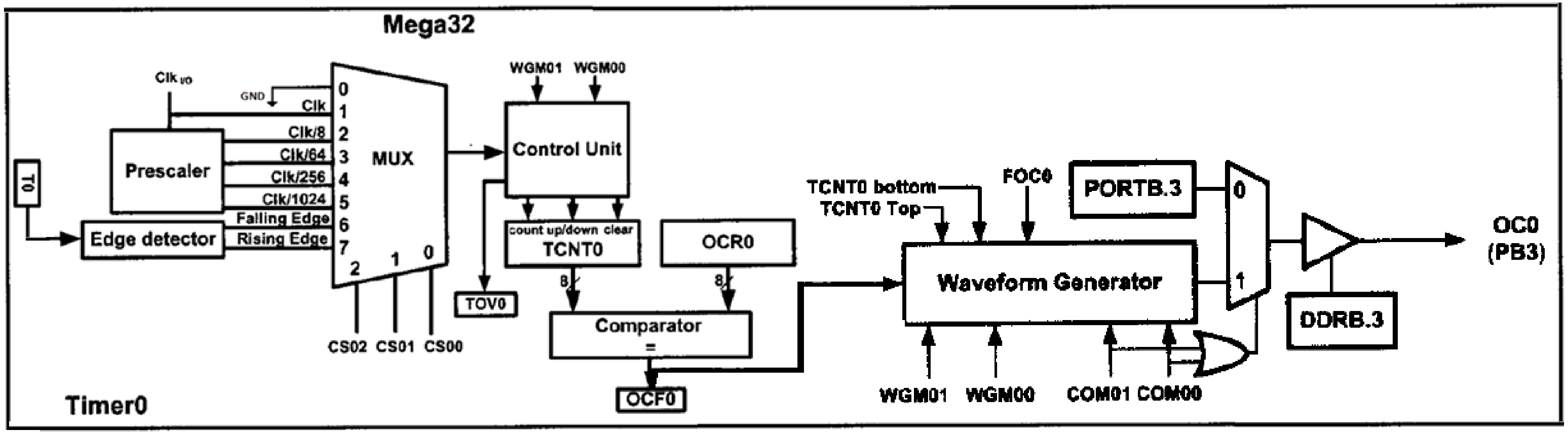
Calculation of desired time overflow counts:

- At the previous example, the user only needs about 63 counts into TCNT0, so every overflow event it is preloaded by 193.
- This solution is not accurate at all because of setting the TCNT0 every event.
- So, according to the last case, overflow or normal mode is not a perfect chosen mode.
- To run the hardware timer to only counts a specific number like the previous example "**63**", the compare match mode is a perfect chosen mode in that case.
- But, how can the timer hardware force the "TCNT0" to zeroize and start from beginning?
- In this case, we need another register to save the value what we need and configure the timer control to compare TCNT0 register with it every clock cycle, Now, let's recognize on this mode in details.

Timer0:

Compare Match “CTC”:

- At this mode, Control Unit of Timer Must be configured to CTC mode.
- In this mode, Output Compare Register “**OCR0**” must be used, it is used to save the number what I want, and the comparator compares “**TCNT0**” with “**OCR0**” every clock cycle, if the value of bit order x in “**TCNT0**” equals with the value of same bit order in “**OCR0**”, the output of comparator will raise “**OCF0**” flag and trigger the wave generator.



Timer0:

Compare Match “CTC”:

- At this mode, Control Unit of Timer Must be configured to CTC mode.

- After raising flag, Counter Register “**TCNT0**” is forced to zeroize by Control unit, and the wave generator will trigger the “**OC0**” pin as “**COM00 , COM01**” are configured:
  - “**00**” Disconnected.
  - “**01**” Toggle “**OC0**” pin on Compare match.
  - “**10**” Clear “**OC0**” pin on Compare match.
  - “**11**” Set “**OC0**” pin on Compare match.

- Flag raising will cause an interrupt event if timer in compare match interrupt is enabled “**PIE**” and global interrupt bit is set “**GIE**”.

Timer0:

Calculation of Compare counts:

- If you have a desired time and you need the number of counts, use:
  - $Counts\ that\ OCR\ will\ be\ set = Time \times \frac{system\ frequency}{prescaler}$
- If you have number of counts and you need the desired time , use:
  - $Time = Counts\ that\ OCR\ will\ be\ set \times \frac{system\ frequency}{prescaler}$
- Also, it can be used to ignore preload term, like specify OCR0 to fire an interrupt after a fixed intervals which it will be a Greatest Common Factor, like:
  - Assume that we have "**16MHz**" system, **"256**" Prescaler and the desired time is "**1s**":
    - The desired time is **1000 ms** and ovf time is 4.096 ms.
    - The greatest common factor is "4" to "1000".
    - So, we will calculate the counts of "**4 ms**".

Timer0:
Calculation of Compare counts:

- The counts of 4 ms:
  - $Counts\ that\ OCR\ will\ be\ set = 0.004 \times \frac{16{,}000{,}000}{256} = 250$
  - So, we need “250” counts to generate a 4ms delay.
  - Now, calculate the CTC Interrupt Counts:
    - $CTC_{INT_{Counts}} = \frac{Desired\ Time}{CTC\ Time} = \frac{1000\ ms}{4\ ms} = 250\ CTC\_INT$
- With this function, we can ignore the preload term.

Timer0:

Laws Summary:

- $Time\ of\ counts = all\ counts\ of\ timer \times \frac{prescaler}{system\ frequency}$

- $Num\ of\ Ovf = \frac{Desired\ Time}{Time\ of\ overflow}$

- $Preload = maxcounts \times (1 - fraction)$

- $Counts\ that\ OCR\ will\ be\ set = Time \times \frac{system\ frequency}{prescaler}$

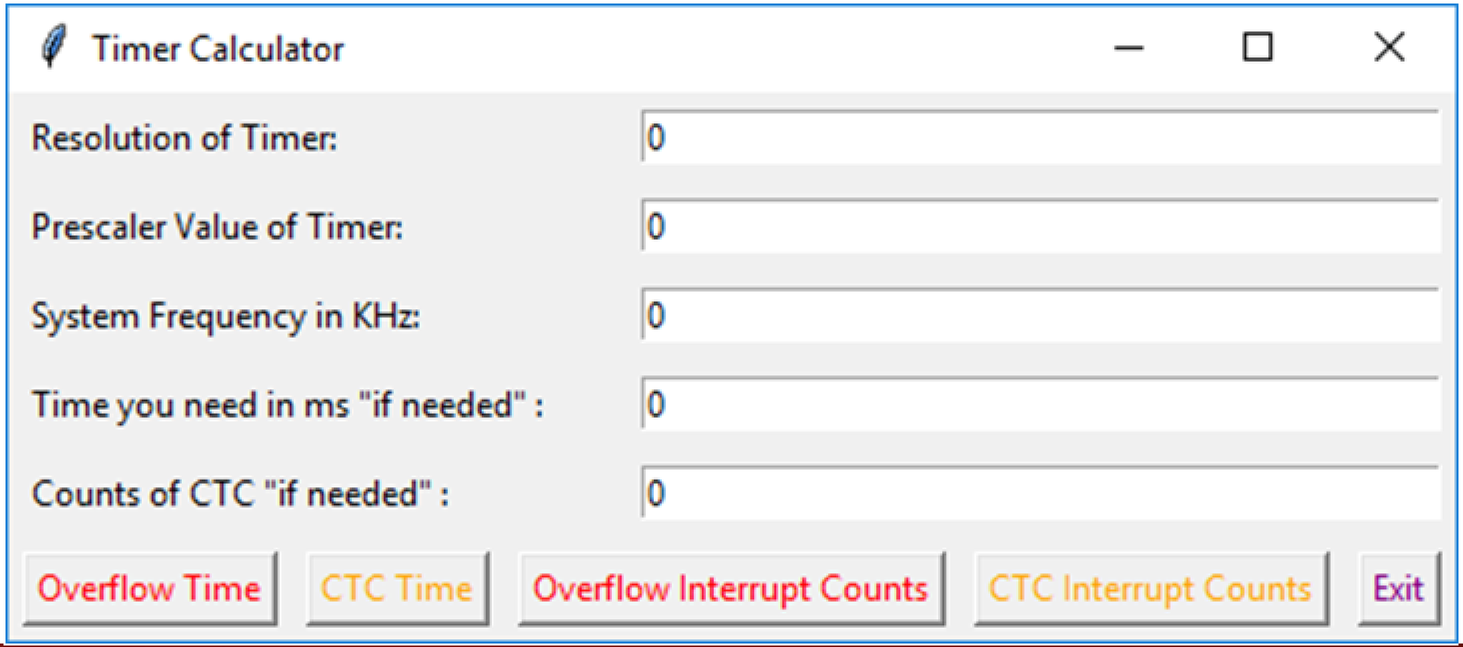
- $CTC_{INT_{Counts}} = \frac{Desired\ Time}{CTC\ Time}$

Timer0:

Laws Summary:

- Now, after well understanding the laws,
  you can use the following Timer Calculator, it is very easy to use:

https://drive.google.com/file/d/1boSS_BvpcsCZAQL6E3M3YKsAl0OuEnmA/view


Timer Calculator

Resolution of Timer:	0
Prescaler Value of Timer:	0
System Frequency in KHz:	0
Time you need in ms "if needed" :	0
Counts of CTC "if needed" :	0

Timer0:

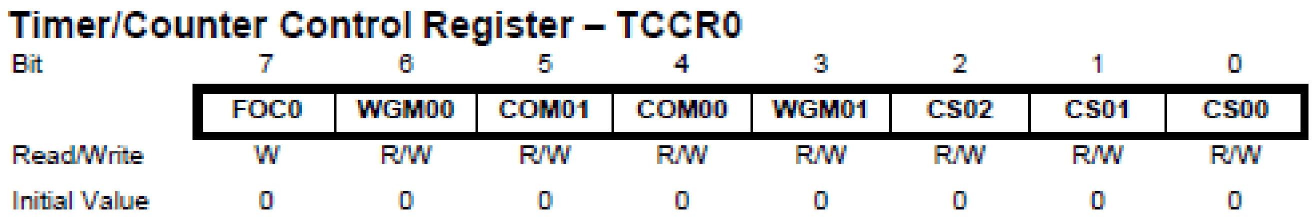
Register Description:

- Timer0/Counter Control Register:
  - Bit 7 – FOC0 Force Output Compare:

    When writing a logical one to the FOC0 bit, an immediate compare match is forced on the Waveform Generation unit. The OC0 output is changed according to its COM01:0 bits setting.

    A FOC0 strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR0 as TOP.

    The FOC0 bit is always read as zero.



Timer/Counter Control Register – TCCR0

Bit	7	6	5	4	3	2	1	0
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Timer0:

Register Description:

- Bit 6, 3 – WGM01:0: Waveform Generation Mode:
  these bits are use to configure the Timer control unit how it works like:
  - “**00**” to configure the timer to work at overflow or normal mode,
    Top means the maximum value can counter register reaches, So in normal mode Top equal 0xFF. Updating of OCR0 means when exactly the new value of OCR0 will be written into the OCR0 register, in overflow the updating of OCR0 will immediately occur.
  - “**10**” to configure the timer to work at CTC mode,
    Top at this mode is at the value of OCR0.
    Updating of OCR0 will immediately occur.

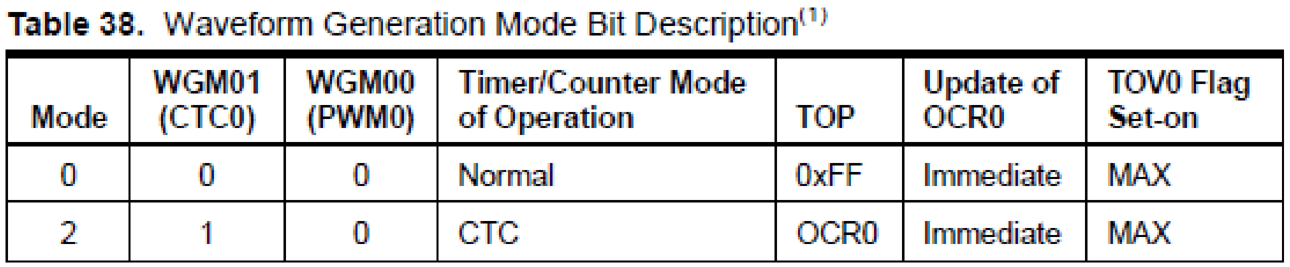


Table 38. Waveform Generation Mode Bit Description[1]

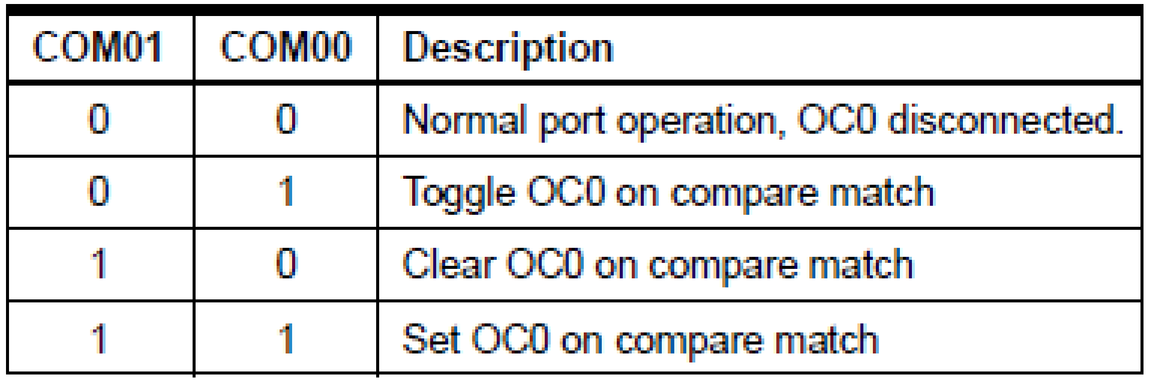
Mode	WGM01 (CTC0)	WGM00 (PWM0)	Timer/Counter Mode of Operation	TOP	Update of OCR0	TOV0 Flag Set-on
0	0	0	Normal	0xFF	Immediate	MAX
2	1	0	CTC	OCR0	Immediate	MAX

Timer0:

Register Description:

- <u>Bit 5:4 – COM01:0: Compare Match Output Mode</u>:
  These bits control the Output Compare pin (OC0) behavior, When OC0 is connected to the pin, the function of the COM01:0 bits depends on the WGM01:0 bit setting.
  Table 39 shows the COM01:0 bit functionality when the WGM01:0 bits are set to a normal or CTC mode (non-PWM):
  - “**00**” OC0 pin will be disconnected, because COM00, COM01 bits are connected to OR-Gate, so the pin will be connected to PORTB not wave generator.
  - “**01**” the pin will be connected to wave generator, and the generator will toggle the OC0 when the value of TCNT0 equals with OCR0 value even if timer mode is overflow.
  - “**10**”, “**11**” the generator will Clear/Set the OC0 when the value of TCNT0 equals with OCR0 value even if timer mode is overflow.

Table 39. Compare Output Mode, non-PWM Mode



COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Toggle OC0 on compare match
1	0	Clear OC0 on compare match
1	1	Set OC0 on compare match

Timer0:
Register Description:

- Bit 2:0 – CS02:0: Clock Select/ event trigger:
  The three Clock Select bits select the clock source/external event trigger to be used by the Timer/Counter.:
  - “**000**” there is no clock source because this channel is connected to ground, so it is also used to disable the timer.
  - “**001 : 101**” different choices for Prescaler of the system frequency, the control unit acts as a timer in these bits settings.
  - “**110 : 111**” to choose the type of edge will be detected on “**T0**” pin, the control unit acts as a counter in these bits settings.

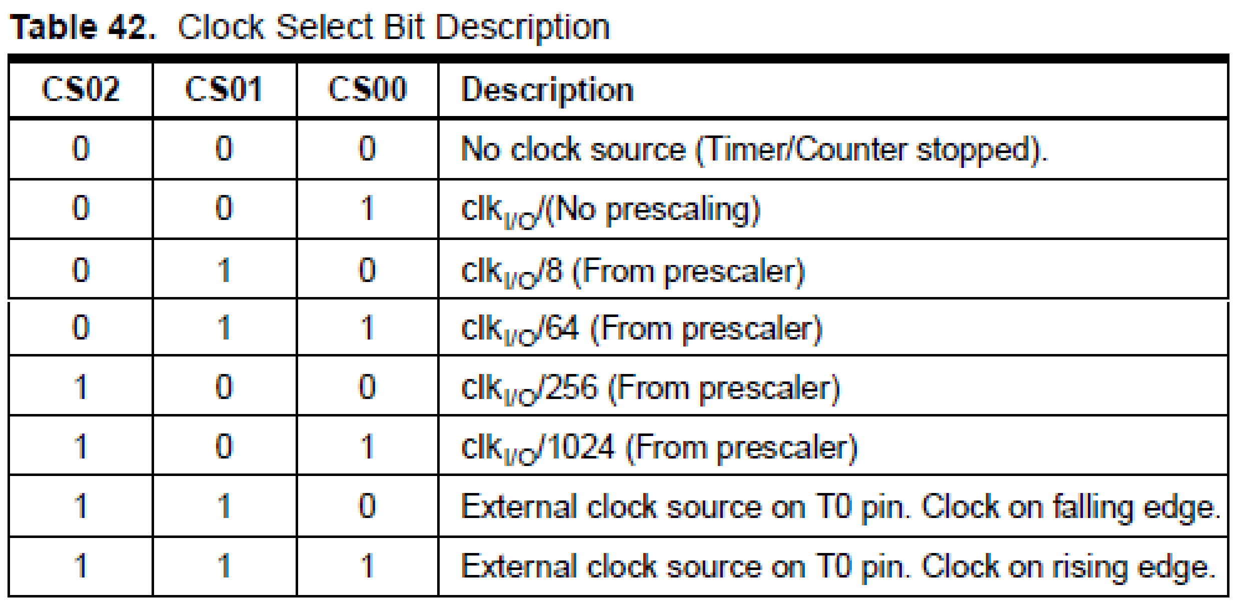


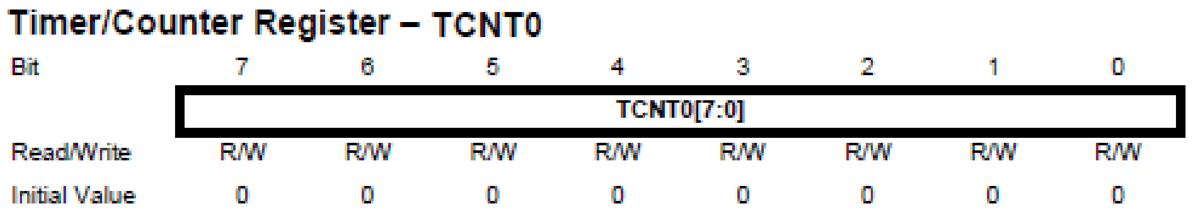
Table 42. Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

Timer0:

Register Description:

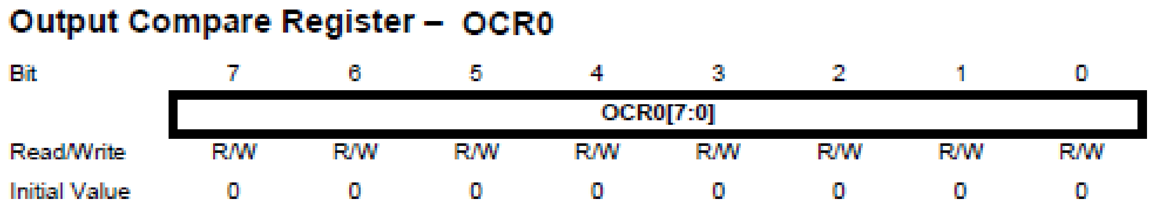
- Timer/Counter Register –TCNT0:
  - The Timer/Counter Register gives direct access, both for read and write operations, to the Timer/Counter unit 8-bit counter. Writing to the TCNT0 Register blocks (removes) the compare match on the following timer clock. Modifying the counter (TCNT0) while the counter is running, introduces a risk of missing a compare match between TCNT0 and the OCR0 Register.



Timer/Counter Register – TCNT0

Bit	7	6	5	4	3	2	1	0
	TCNT0[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- Timer/Counter Register –TCNT0:
  - The Output Compare Register contains an 8-bit value that is continuously compared with the counter value (TCNT0). A match can be used to generate an output compare interrupt, or to generate a waveform output on the OC0 pin.

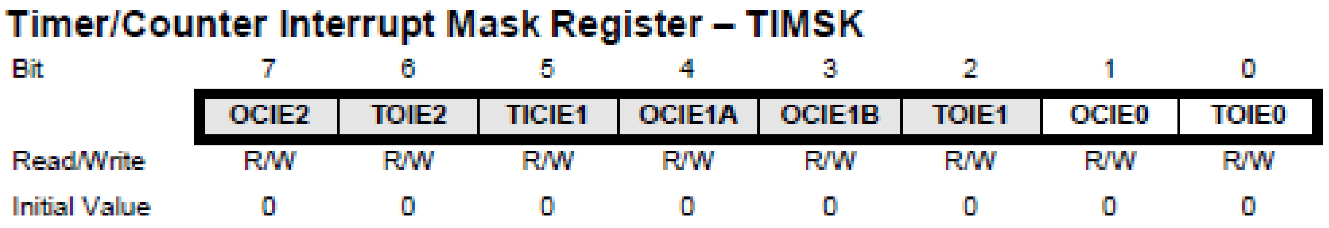


Output Compare Register – OCR0

Bit	7	6	5	4	3	2	1	0
	OCR0[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Timer0:
Register Description:

- Timer/Counter Interrupt Mask Register – TIMSK:
  - Bit 1 – OCIE0: Timer/Counter0 Output Compare Match Interrupt Enable:
    When the OCIE0 bit is written to one, and the I-bit in the Status Register is set (one), the Timer/Counter0 Compare Match interrupt is enabled.



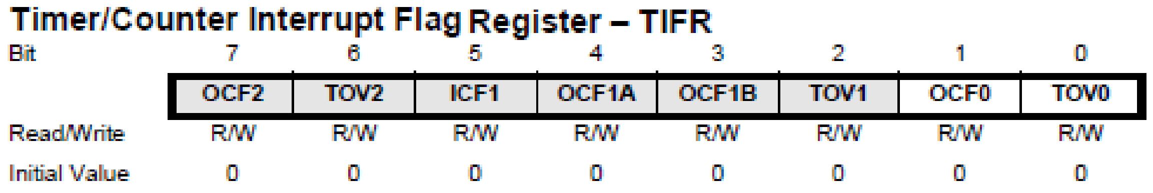
Timer/Counter Interrupt Mask Register – TIMSK

Bit	7	6	5	4	3	2	1	0
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

  - Bit 1 – OCIE0: Timer/Counter0 Output Compare Match Interrupt Enable:
    When the TOIE0 bit is written to one, and the I-bit in the Status Register is set (one), the Timer/Counter0 Overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter0 occurs, i.e., when the TOV0 bit is set in the Timer/Counter Interrupt Flag Register – TIFR.

Timer0:
Register Description:

- Timer/Counter Interrupt Flag Register – TIFR:
  - Bit 1 – OCF0: Output Compare Flag 0:
    The OCF0 bit is set (one) when a compare match occurs between the Timer/Counter0 and the data in OCR0 – Output Compare Register0. OCF0 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0 is cleared by writing a logic one to the flag.



Timer/Counter Interrupt Flag Register – TIFR

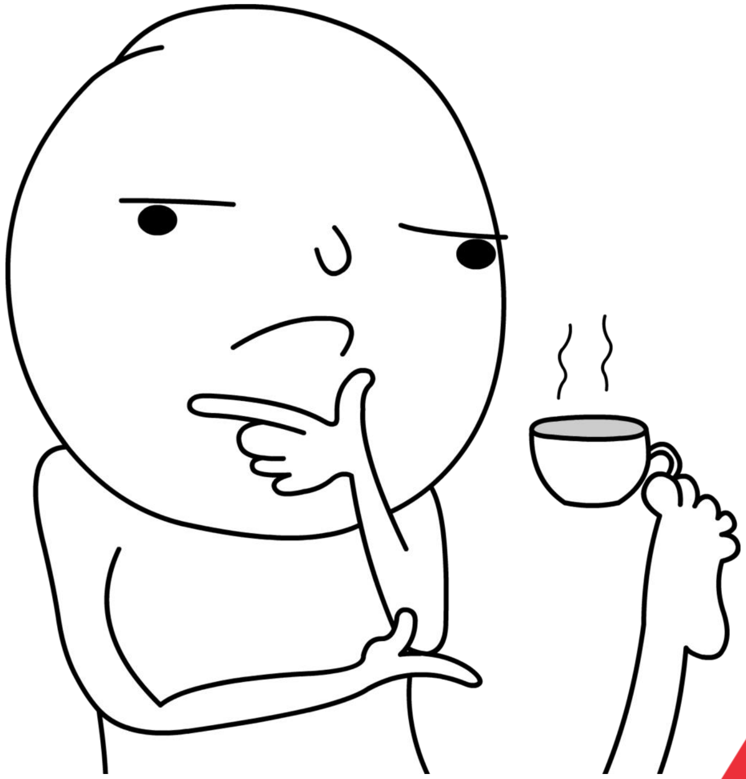
Bit	7	6	5	4	3	2	1	0
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

  - Bit 0 – TOV0: Timer/Counter0 Overflow Flag:
    The bit TOV0 is set (one) when an overflow occurs in Timer/Counter0. TOV0 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV0 is cleared by writing a logic one to the flag. When the SREG I-bit, TOIE0 (Timer/Counter0 Overflow Interrupt Enable), and TOV0 are set (one), the Timer/Counter0 Overflow interrupt is executed. In phase correct PWM mode, this bit is set when Timer/Counter0 changes counting direction at $00..

TIMER0 Driver:

Time To

Code



THANK YOU!