



INTERFACING

TIMER :INPUT CAPTURE UNIT

AMIT

Input Capture Unit:

Definition:

- Input capture is a method of dealing with input Pulse-Width-Modulation signals in an embedded system.
- Embedded systems using input capture will record a timestamp in memory when an input signal is received. It will also set a flag indicating that an input has been captured. This allows the system to continue executing without interruption while an input is being received while still having the capability to trigger events based on the exact time when the input was received.

Input Capture Unit:

Definition:

- The corresponding capability to trigger an output at a specified time, based on a timestamp in memory, is called output compare.
- There are many programmable interrupt controllers that provide dedicated input capture pins and a programmable counter along with it. These pins generate interrupts to the controller, which then executes an interrupt service routine. The interrupts can be programmed to occur at the rising or falling edge of the input signal, depending on requirements.

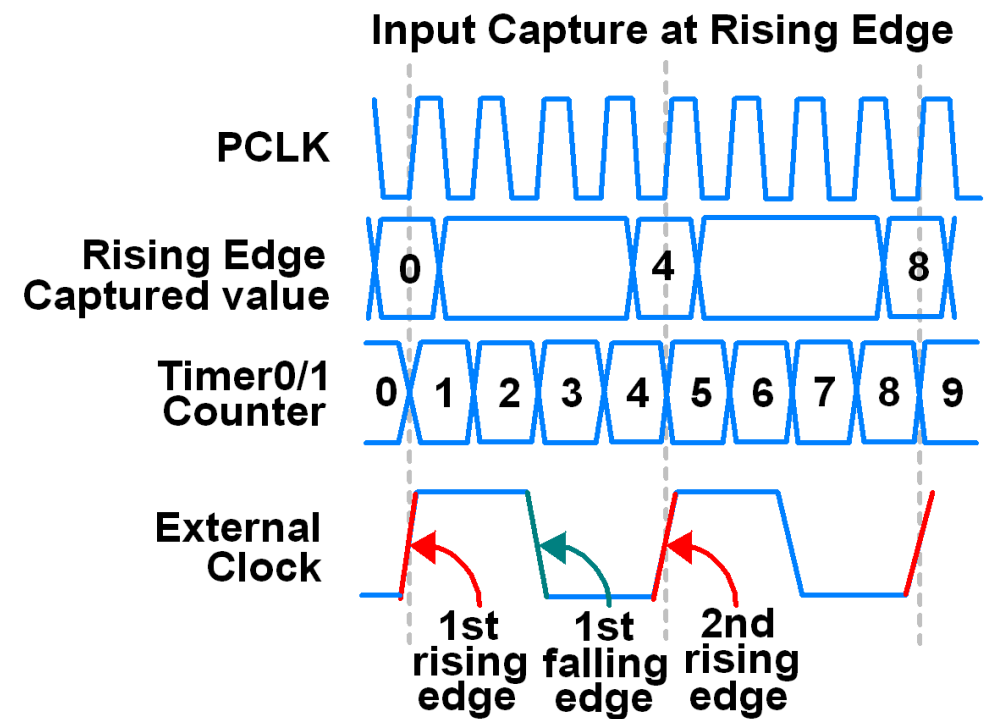
Input Capture Unit:

Definition:

- Input capture is a method to measure the most important main parameters of any Pulse-Width Modulation that are:
 - Duty Cycle.
 - Frequency.
- Those parameters directly depend on the time when pulse was being high, and the full time of period or cycle.
- NOTE:
if your microcontroller has the main peripheral “DIO, Timer” and the additional peripherals like “ADC, EXTI”, you can create any peripheral function by software.

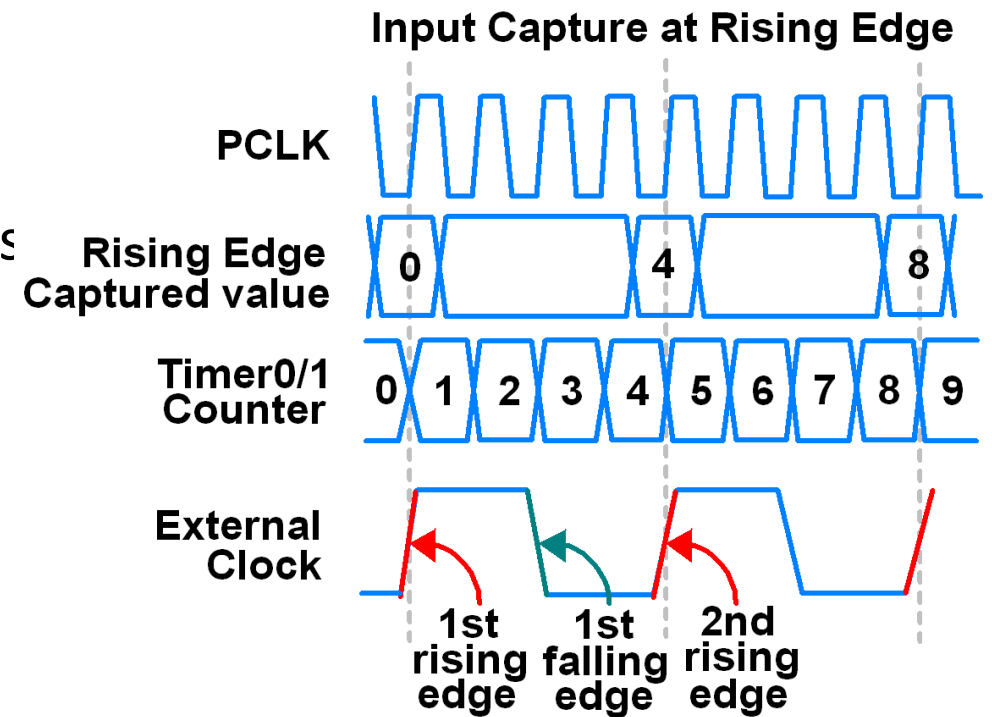
Input Capture Unit: Implementation:

- Any PWM contains four-main levels “rising edge, high level, falling edge, low level”.
- The period time can be calculated by calculating the time between any two points that have the same phase, so the period can be calculated by counting timer between two sequence rising edges or two sequence falling edges.
- The high-level period “**ON Time**” can be calculated by counting timer between two sequence rising/falling edges, and vice versa.



Input Capture Unit: Implementation:

- At first rising-edge detecting, the Timer must be powered on or zeroized if it was powered on before, then change the detection to falling.
- At first falling edge, Timer must be read, this value indicates to the high-level period “**ON Time**”, then change the detection to rising.
- At second rising-edge, Timer must be read, this value indicates to the full period, then change the detection to rising to read more cycles or turn off timer and detection system.



Input Capture Unit:

**Guess,
how can we
Implement
ICU?**



AMIT

Input Capture Unit: **Implementation:**

- As we mentioned before, any peripheral can be implemented by software if we have DIO and Timer or External interrupt.
- We will take advantage the External Interrupt for edge detection, and the Timer for counting the time.
- We will follow the sequence that we mentioned before, but assume that: the time of high level or the full period is higher than our timer overflow counts?, this will violate the results because we did not count if any overflow was happened or not.

Input Capture Unit:

Software Implementation:

- In this method, we will use EXTIO, and TIMER0:
 - EXTIO: for edge detection.
 - Timer0: for timing calculation.
- Reading the time depends on the Interrupt Service Routine of External Interrupt 0.
- Reading the Interrupt Service Routine of Timer 0 for reading if the Timer was overflowed or not.
- If Timer is overflowed, it must to read how many times that the timer was overflowed.

Input Capture Unit:

Sudo Code of Software Implementation:

```
int TimeON, TimeTotal, OnCount, TotalCount, State, Counter;
```

```
int main(void)
```

```
{
```

```
    //set Timer on Overflow Mode;
```

```
    //set EXTI at Rising edge;
```

```
    Enable INT of Timer0;
```

```
    Enable INT of EXTI0;
```

```
    while (1)
```

```
    {
```

```
        if (state == 3)
```

```
        {
```

Input Capture Unit:

Sudo Code of Software Implementation:

```
duty = (TimeON + (256 x OnCount) ) / (TimeTotal + (256 x TotalCount) );
```

```
freq= F_CPU / ( (TimeTotal + (256 x TotalCount) x Prescaler );
```

```
state++;
```

```
}
```

```
}
```

```
}
```

```
ISR(Timer0)
```

```
{
```

```
    counter++;
```

```
}
```

Input Capture Unit:

Sudo Code of Software Implementation:

```
ISR(EXTIO)
{
    if (state == 0)
    {
        //zeroize TCNT0;
        //zeroize counter;
        //set EXTIO on falling;
        state++;
    }
    else if (state == 1)
    {
        TimeON = TCNT0;
        OnCount = counter;
```

Input Capture Unit:

Sudo Code of Software Implementation:

```
        //set EXTI0 on Rising edge;
        state ++;
    }
    else if (state == 2)
    {
        TimeTotal = TCNT0;
        TotalCount= counter;
        //turn off EXTI;
        //turn off Timer;
        state ++;
    }
}
```

Input Capture Unit:

Hardware Implementation:

- In this method, we will use TIMER1, it has a 16-bit register that can capture the reading of TCNT1 if the edge has been detected.
- ICR1 is immediately updated by the value of TCNT1, if the edge has been detected.
- The edge detector can be configured to detect any edge “Falling or Rising” by configuration “**ICES1**” bit into “**TCCR1B**” register:
 - “**0**”: to detect falling edge.
 - “**1**”: to detect rising edge.
- When the ICR1 is used as TOP value, the ICP1 is disconnected and consequently the input capture function is disabled.

Input Capture Unit:

Hardware Implementation:

- “**ICNC1**” bit into “**TCCR1B**” register is used for noise cancellation, When the Noise Canceler is activated, the input from the Input Capture Pin “**ICP1**” is filtered. The filter function requires four successive equal valued samples of the “**ICP1**” pin for changing its output.
- The input capture is therefore delayed by four Oscillator cycles when the Noise Canceler is enabled.
- “**ICP1**” pin exists into “**PORTD**” pin 6, it should be configured as input.
- The mode of timer must not be any mode that using “**ICR1**” as a Top, we will choose as example “**Normal**” or overflow mode.

Input Capture Unit:

Sudo Code of Hardware Implementation:

```
int TimeON, TimeTotal, OnCount, TotalCount, State, Counter;
```

```
int main(void)
```

```
{
```

```
    //set Timer1 on Overflow Mode;
```

```
    //set ICP1 pin at Rising edge;
```

```
    //Enable INT of Timer1 overflow;
```

```
    //Enable INT of ICU;
```

```
    while (1)
```

```
    {
```

```
        if (state == 3)
```

```
        {
```


Input Capture Unit:

Sudo Code of Software Implementation:

```
        duty = (TimeON + (65536ul x OnCount) ) / (TimeTotal + (65536ul x
TotalCount) );
        freq= F_CPU / ( (TimeTotal + (65536ul x TotalCount) x  Prescaler );
        state++;
    }
}
}

ISR(Timer1_OVF)
{
    counter++;
}
```

Input Capture Unit:

Sudo Code of Software Implementation:

```
ISR(ICU)
{
    if (state == 0)
    {
        //zeroize TCNT1;
        //zeroize counter;
        //set ICP1 on falling;
        state++;
    }
    else if (state == 1)
    {
        TimeON = TCNT1;
        OnCount = counter;
```

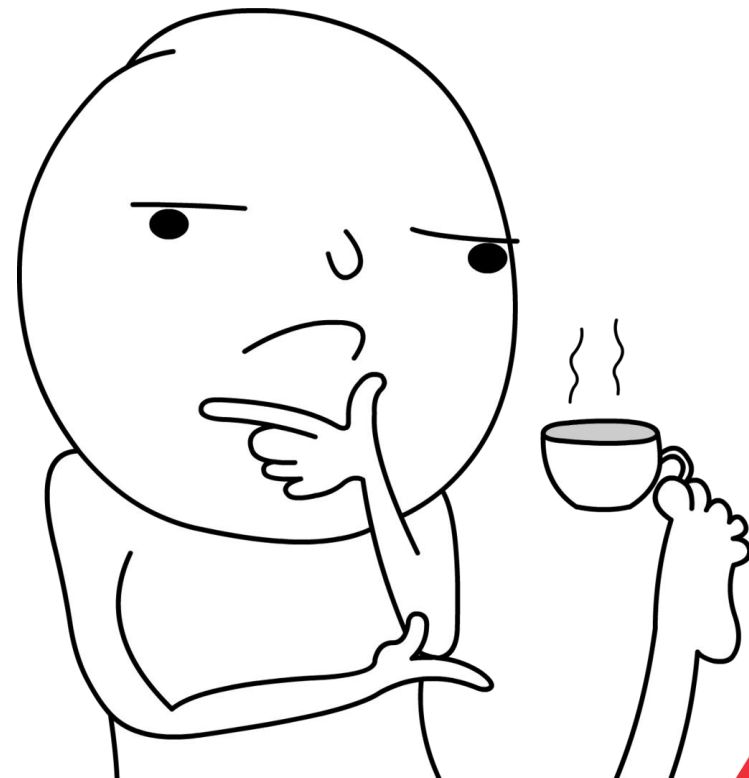
Input Capture Unit:

Sudo Code of Software Implementation:

```
        //set ICP1 on Rising edge;
        state ++;
    }
    else if (state == 2)
    {
        TimeTotal = TCNT1;
        TotalCount= counter;
        //turn off Timer1;
        state ++;
    }
}
```

ICU Driver:

Time To
Code



AMIT

THANK YOU!

AMIT