INTERFACING

TIMERS 3 [ICU & WDT]

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.

AMIT

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.

AMIT

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.

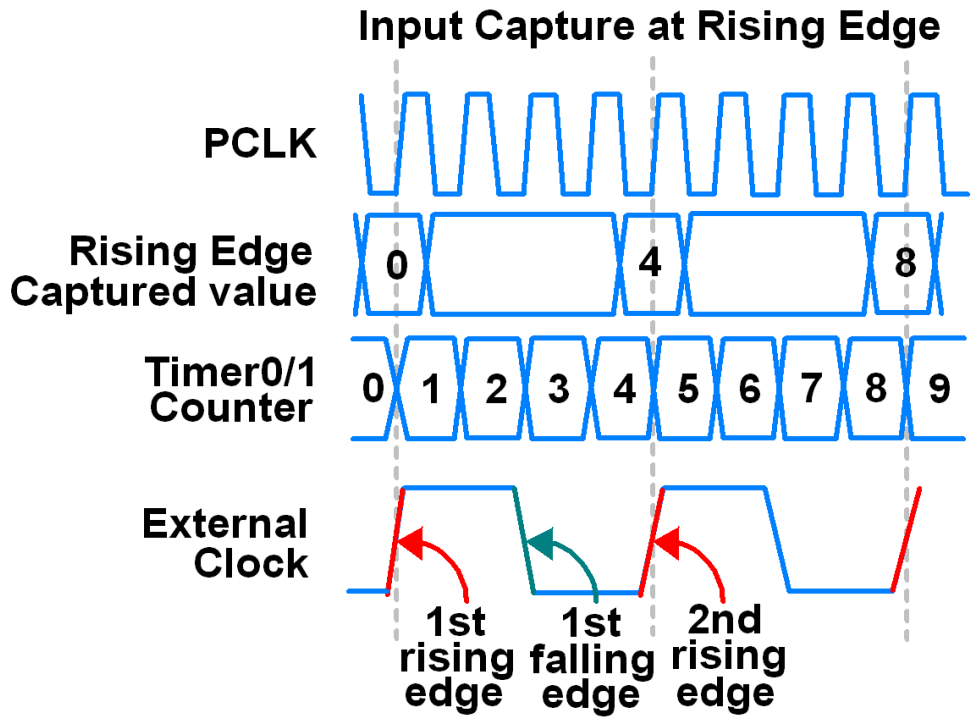
AMIT

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.

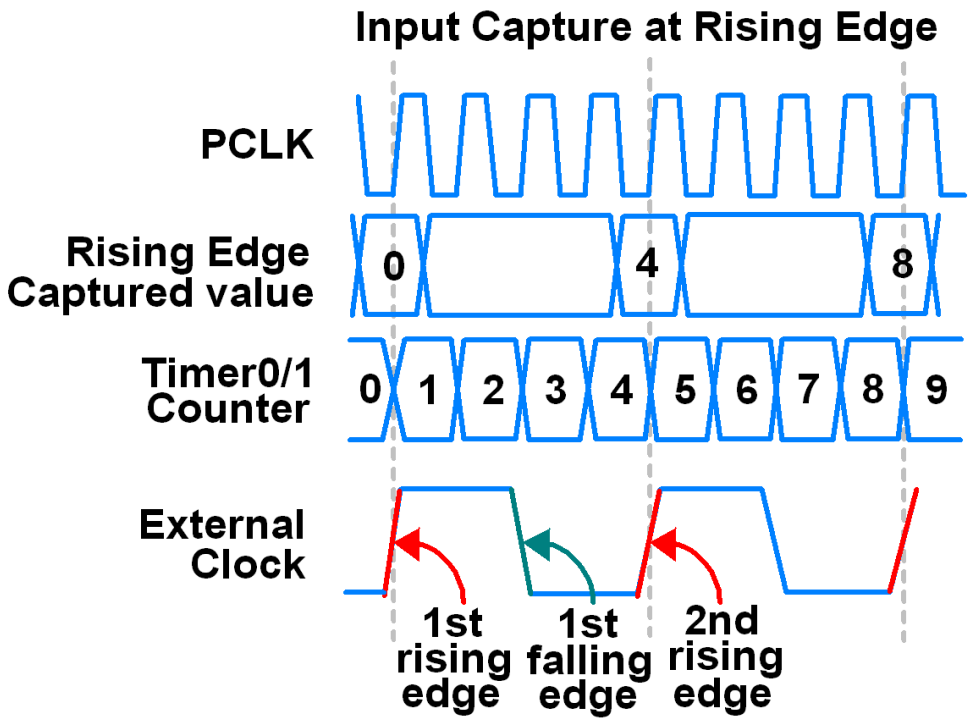
Input Capture Unit:

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.

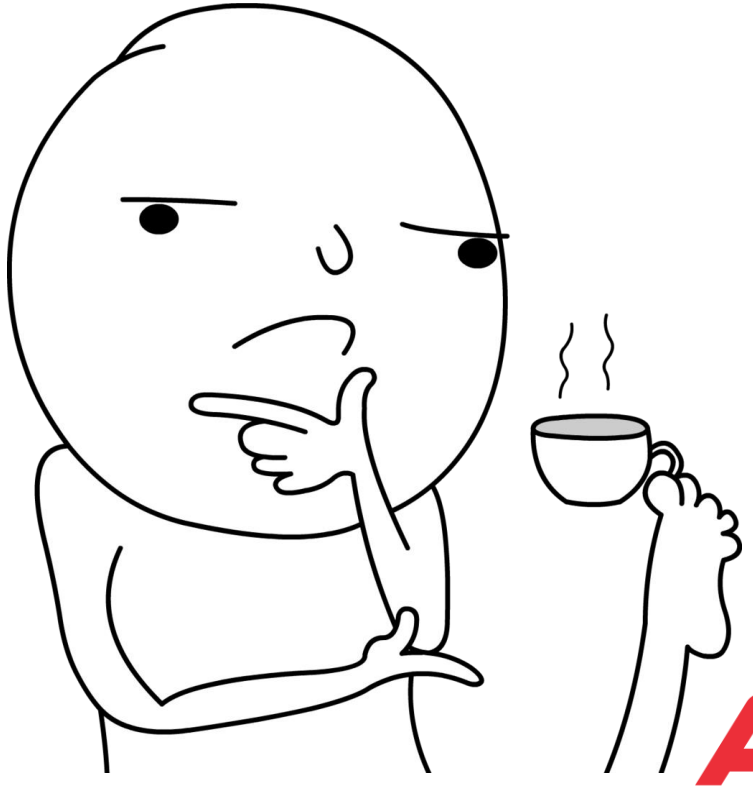
- At first falling edge, Timer must be read, this value indicates to the high-level period "**ON Time**", then change the detection to rising.

- 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 of not.

AMIT

Input Capture Unit:

Software Implementation:

- In this method, we will use EXTI0, and TIMER0:
  - EXTI0: 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.

AMIT

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)
        {
```

AMIT

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(EXTI0)
{
    if (state == 0)
    {
        //zeroize TCNT0;
        //zeroize counter;
        //set EXTI0 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 ++;
        }
}
```

AMIT

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.

AMIT

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.

AMIT

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 ++;
        }
}
```

AMIT

ICU Driver:

Time To Code

AMIT

Watchdog Timer:

Definition:

- A watchdog timer is a piece of hardware that can be used to automatically detect software anomalies and reset the processor if any occur. Generally speaking, a watchdog timer is based on a counter that counts down from some initial value to zero. The embedded software selects the counter's initial value and periodically restarts it. If the counter ever reaches zero before the software restarts it, the software is presumed to be malfunctioning and the processor's reset signal is asserted. The processor (and the embedded software it's running) will be restarted as if a human operator had cycled the power.

AMIT

Watchdog Timer:

Definition:

Watchdog Timer:

Definition:

- A watchdog timer is commonly used to calculate the approximately execution time of any function, by enabling the watchdog timer before calling and turning off after calling, if the system is reset, that means that the execution time of function exceeds the time that watchdog timer has been configured and if the system is not reset, that means that the time of function execution may be equaled or less than the time of watchdog timer.

- It is also used to avoid the system from dead lock, like when more than one task that needs the same semaphores, dead lock will be explained clearly later.

AMIT

Watchdog Timer:

Register Description:

- Watchdog Timer Control Register – WDTCR:
  - Watchdog Turn-off Enable: "**WDTOE**"

    This bit must be set when the WDE bit is written to logic zero. Otherwise, the Watchdog will not be disabled. Once written to one, hardware will clear this bit after four clock cycles. Refer to the description of the WDE bit for a Watchdog disable procedure.

Watchdog Timer Control Register – WDTCR

Bit	7	6	5	4	3	2	1	0
	–	–	–	WDTOE	WDE	WDP2	WDP1	WDP0
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

AMIT

Watchdog Timer:

Register Description:

- Watchdog Timer Control Register – WDTCR:
  - When the WDE is written to logic one, the Watchdog Timer is enabled, and if the WDE is written to logic zero, the Watchdog Timer function is disabled. WDE can only be cleared if the WDTOE bit has logic level one. To disable an enabled Watchdog Timer, the following procedure must be followed:
    - In the same operation, write a logic one to WDTOE and WDE. A logic one must be written to WDE even though it is set to one before the disable operation starts.
    - Within the next four clock cycles, write a logic 0 to WDE. This disables the Watchdog.

AMIT

Watchdog Timer:
Register Description:

- Watchdog Timer Control Register – WDTCR:
  - The WDP2, WDP1, and WDP0 bits determine the Watchdog Timer prescaling when the Watchdog Timer is enabled. The different prescaling values and their corresponding Timeout Periods are shown in Table 17.

Table 17. Watchdog Timer Prescale Select

WDP2	WDP1	WDP0	Number of WDT Oscillator Cycles	Typical Time-out at V_{CC} = 3.0V	Typical Time-out at V_{CC} = 5.0V
0	0	0	16K (16,384)	17.1 ms	16.3 ms
0	0	1	32K (32,768)	34.3 ms	32.5 ms
0	1	0	64K (65,536)	68.5 ms	65 ms
0	1	1	128K (131,072)	0.14 s	0.13 s
1	0	0	256K (262,144)	0.27 s	0.26 s
1	0	1	512K (524,288)	0.55 s	0.52 s
1	1	0	1,024K (1,048,576)	1.1 s	1.0 s
1	1	1	2,048K (2,097,152)	2.2 s	2.1 s

AMIT

Watchdog Timer:

Sudo Code of Watchdog Timer:

- Turn on Watchdog Timer:

```
WDTCR = (1<<WDE) | (Prescaler);
```

- Turn on Watchdog Timer:

```
WDTCR = (1<<WDE) | (1<<WDTOE);
WDTCR = 0x00;
```

AMIT

Watchdog Driver:

Time To Code

AMIT

THANK YOU!